;login:
THE MAGAZINE OF USENIX & SAGE
August 2001 • Volume 26 • Number 5

Special Focus Issue: Clustering

Guest Editor: Joseph L. Kaiser

inside:

USENIX & SAGE
The Advanced Computing Systems Association &
The System Administrators Guild

USENIX Upcoming Events

5th Annual Linux Showcase and Conference

Co-sponsored by USENIX & Atlanta Linux Showcase, in cooperation with Linux International

NOVEMBER 6–10, 2001
Oakland, California, USA

http://www.linuxshowcase.org

Registration materials available: August, 2001
Final Papers due: September 14, 2001

XFree86 Technical Conference

(Co-located with the 5th Annual Linux Showcase & Conference)

NOVEMBER 8-9, 2001
Oakland, California, USA

http://www.usenix.org/events/xfree86/
Registration materials available: August, 2001
Camera-ready final papers due: September 14, 2001

15th Systems Administration Conference (LISA 2001)

Sponsored by USENIX & SAGE

DECEMBER 2–7, 2001
San Diego, California, USA

http://www.usenix.org/events/lisa2001

Registration materials available: September, 2001
Camera-ready final papers due: October 1, 2001

Conference on File and Storage Technologies (FAST)

Co-sponsored by IEEE TCOS, in cooperation with ACM SIGOPS

JANUARY 28–29, 2002
Monterey, California, USA

http://www.usenix.org/events/fast/

Notification to authors: September 14, 2001
Registration materials available: October, 2001
Camera-ready final papers due: November 15, 2001

BSDCon 2002

FEBRUARY 11-14, 2002
San Francisco, California, USA

http://www.usenix.org/events/bsdcon02/

Notification to authors: October 1, 2001
Registration materials available: November, 2001
Camera-ready final papers due: December 4, 2001

The Fourth NordU/USENIX Conference (NordU/USENIX 2002)

Sponsored by EurOpen.SE and USENIX

FEBRUARY 18-22, 2002
Helsinki, Finland

http://www.nordu.org/NordU2002

Extended abstracts due: September 7, 2001
Notification of acceptance: October 12, 2001
Final papers due: December 7, 2001

2002 USENIX Annual Technical Conference

JUNE 9–14, 2002
Monterey, California, USA

http://www.usenix.org/events/usenix02/

Freenix Submissions due: November 12, 2001
General Session Submissions due: November 19, 2001
Notification of acceptance: January 22, 2002
Freenix Papers due for final approval: April 8, 2002
Camera-ready final papers due: April 16, 2002

2nd Java™ Virtual Machine Research and Technology Symposium

AUGUST 1-2, 2002
San Francisco, California, USA

11th USENIX Security Symposium

AUGUST 5-9, 2002
San Francisco, California, USA

Paper Submissions due: January 28, 2002
Notification of acceptance: March 25, 2002
Camera-ready final papers due: May 13, 2002

2nd Workshop on Industrial Experiences with Systems Software (WIESS)

Co-sponsored by USENIX , IEEE TCOS & ACM SIGOPS

DECEMBER 8, 2002
Boston, Massachusetts, USA

5th Symposium on Operating Systems Design and Implementation (OSDI)

Co-sponsored by USENIX , IEEE TCOS & ACM SIGOPS

December 9-11, 2002
Boston, Massachusetts, USA

For a complete list of future USENIX events, see **http://www.usenix.org/events**

contents

;login: Vol. 26 #5, August 2001

;login: is the official magazine of the USENIX Association and its Special Technical Group, SAGE.

;login: (ISSN 1044-6397) is published bimonthly, plus July and November, by the USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, CA 94710.

$50 of each member's annual dues is for an annual subscription to *;login:*. Subscriptions for nonmembers are $60 per year.

Periodicals postage paid at Berkeley, CA, and additional offices.

POSTMASTER: Send address changes to *;login:*, USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, CA 94710.

in this issue

by Joseph L. Kaiser

Guest Editor. In 1996 Joe Kaiser accidentally became a system administrator in Colorado. He is now intentionally practicing this craft at Fermi National Accelerator Lab in Illinois as the current primary lead sysadmin for the CMS collaboration. CMS is in the process of creating a 1000+ node multiuse analysis cluster for the LHC at CERN in Switzerland, so building clusters are his most pressing concern. His only other pressing concerns are keeping his wife and mother of his six children happy and trying to find whitewater and mountains in the Midwest.

jlkaiser@fnal.gov

When my wife and I were dating, we used to play free-association games, mostly for giggles, occasionally for insight. It usually went like this: she said a word, like "ham," and I would come back with whatever came to mind first, like "sandwich" or "pig" or "Jay Leno," whatever my id happened to come up with. (When you are dating, your id, as a general rule, is fixated on certain topics, and our free associations almost always went there – okay, my part of the free associations almost always went there.) This has absolutely nothing to do with this month's issue except that we are going to play this game right now. (No, not my wife and I: you and I.) I get to start, and the first word is "cluster."

What comes immediately to mind?

"VAX, only VAX, and nothing but VAX," "L.A. traffic at any time of day," "meetings," "Beowulf," "farms," "supercomputing," "teraflops," "gobs and gobs of machines that only I take care of." If none of this leaps to your mind and other things do, I might suggest that this is the incorrect reading material for you, and you should go find the media that makes your id happy. If the word VAX sprang from your unconsciousness, then you are significantly older than I am. Please remember that, as a youngster, I am not qualified to argue the merits of VAX clustering or whatever is The One True Clustering. If you thought "Oh, L.A.," I'm sorry, but you chose to live there. If you thought anything that has to do with computers grouped in some sort of logical domain in which CPU, disk, and network resources are used to accomplish a specific computational job other than running user Netscape processes and that will, someday, ideally, be managed as one machine image, then this media input is for you. It's about "clusters."

This word, as this issue hopefully demonstrates, means different things to different people depending on what exactly they are building. Some people are building Beowulf clusters, which are tightly coupled machines in both hardware and software which are designed to provide supercomputer-level, high-performance computing for problems that have a high degree of parallelism. Others are building server farms for Web, DNS, and database services. These sorts of clusters give high availability and increased performance by doing load sharing and dynamic resource allocation. Still other facilities are building compute and analysis clusters with machines that are loosely coupled and are designed to bring lots of CPU and disk resources into analyzing certain datasets. These are for problems which are not given to being easily parallelized and for which high performance computing is not as critical as high throughput computing.

My focus on clusters came about because of my current job. I am employed as a system administrator at Fermi National Accelerator Lab. I took this position almost a year ago and was assigned to be one of the liaisons for the CMS Computing Group. (CMS stands for Compact Muon Solenoid which is the name of a particle detector that will detect itty-bitty particles in the new Large Hadron Collider (LHC) when it is built in Switzerland at CERN.) As it stands right now, CMS expects that they will need in excess of 1000 compute nodes to analyze 600 terabytes per year of on-disk data when the LHC starts running in 2005. That is half of a petabyte for the U.S. facility alone. There will be more than a petabyte per year from the LHC for the CMS experiment distributed throughout the world. ATLAS is the other LHC experiment, and it has equivalent computing needs. Currently, as far as I am aware, there is no supercomputer, either single machine or cluster, that has been built with commodity hardware and open source software that can handle this amount of data. (Some may object to this statement, but most of the really, really massively mind-bogglingly large machines have some sort of

proprietary component to them. I also put in the "as far as I am aware" caveat, so please keep letters to the editor to a minimum.) Building clusters of this type and scale is what the high-energy physics field is faced with having to do.

The high priority concerns of building clusters center on the following:

- Affordability – how do we build something big enough and cheap enough to do the job we need it to do?
- Reliability – how can we use commodity hardware and software, either open source or homegrown, to provide a computing solution that provides reliable data in a reliable and available hardware framework?
- Scalability – how do we build our clusters to be able to grow from a few dozen or hundred nodes to a couple thousand nodes without exponentially increasing hardware and software costs and management issues?
- Manageability – how do we manage and maintain large-scale clusters with the smallest number of staff, the least amount of hassle, and in the smallest amount of space.

Between the covers of this magazine, you will see the beginnings of answers to these questions. We start out with reports from the chairpersons of three conferences dedicated to supercomputing and cluster building. Al Geist reports on the Fifth Workshop on Distributed Supercomputing: Scalable Cluster Software held in Cape Cod. Craig A. Lee reports on the CCGrid conference held recently in Brisbane, Australia, and Dane Skow and Alan Silverman give us the who, what, and when from the Large-Scale Cluster Computing Workshop held at Fermi National Accelerator Lab. While these summaries are short, they give pointers to where you can find out more about the goings on at these conferences. Also, Andreas Boklund kindly sent an article about his home machines just before leaving for the CCGrid conference in Australia.

After the conference reports, we have a long-view article by Dr. Thomas Sterling, one of the "fathers" of Beowulf clustering, talking about the next plateau in high-performance computing. As you will see, he was instrumental in inspiring the work on the PIRUN cluster in Thailand (among many others). It was his work that helped to bring us to the stage we are at now, and his thoughts for the future will have great impact if they are heeded.

For some practical insight, we next have three articles on facilities that have built or are building their clusters. Thomas Throwe contributes with an account of the current configuration of the RHIC farms at Brookhaven National Lab. Martin Krzywinski of the Genome Sequence Centre in Vancouver describes the ongoing creation of their cluster for bioinformatics work. He describes some interesting solutions to problems relating to the physical space they have available. One of the most impressive testimonies of the power of supercomputing with commodity clusters I received was Putchong Uthayopas' description of building the PIRUN Beowulf cluster at Kasetsart University in Thailand. It is a tribute to the human capacity to find a solution that is simple, beautiful, and has significant impact in the user community. Lastly, I have included in this section an article from Dave Lifka from the Cornell Theory Center on their Windows cluster. This was illuminating since I had not heard of or had any experience with anyone using Windows as a compute cluster.

Everyone needs tools, so now that you have an idea of what people are building and why they are building it, you should know what you can use to monitor and administrate it. Stephen Chan and friends from Lawrence Berkeley National Laboratory give an overview of some of the open source tools they have experimented with, and Don

Holmgren and Ron Rechenmacher of Fermi Lab give us a look at "rgang," a threaded application for cluster administration.

One of the things that impresses me most about the computing community is the willingness of many to experiment at home with software and hardware that they find compelling. It is this desire that has started great companies and really good bands in garages all over the world. With this in mind, I solicited home cluster configuration descriptions off of the Beowulf mailing list. You have here three of the most interesting ones I received, from Robert Brown, Andreas Boklund, and Richard Schilling. I hope that these experiences provoke others to try something new.

I decided to end with Ian Lumb of Platform Computing and his paper "Linux Clustering for High-Performance Computing." He gives us some practical considerations for achieving the next plateau that Dr. Sterling spoke about at the beginning.

This issue is not meant to be a definitive work, especially since it does not cover server clusters performing Web services – think especially Google and Akamai – but it is meant to show the different ways in which people are thinking about building, managing, and architecting clusters for serious computational use. In the last five to eight years it has become possible to build a computer out of commodity off-the-shelf (COTS) hardware and your favorite open source OS to create your own personal supercomputer. Universities, national labs, and some businesses are doing it now. If you don't already have one, you will probably be managing or building one sooner or later. The price-to-performance ratio is too tempting for the fiscally minded in your department/division/company to not notice. Hopefully, you will get a general idea of the current state of some of the issues that HEP, bioinformatics, and academia are dealing with and also some pointers in the direction of where to go next.

Thanks for reading.

motd

USA Computing Olympiad Wins One

The USA delegation to the International Programming Contest (known as the IOI – International Olympiad on Informatics) performed outstandingly well last week (July 16-20) in Tampere, Finland. Only recently chosen (see p. 92), the team tallied enough points to outscore every team but one.

by Rob Kolstad

Dr. Rob Kolstad has long served as editor of *;login:*. He is also head coach of the USENIX-sponsored USA Computing Olympiad.

kolstad@usenix.org

Best of all, gold-medal veteran Reid Barton topped all competitors with a whopping 580 points out of 600. He outpaced second place by almost 50 points and third place by about 50 more! He is the USA team's first overall winner.

Reid is an especially talented competitor. Just a week ago, he earned his fourth consecutive gold medal at the International Math Olympiad, posting a perfect score on his way to this world-record breaking achievement.

Reid is home-schooled, though he doesn't spend his time "at home." He attends classes at the local high school, MIT, Tufts, and other universities "local" to his Boston home. He plays the piano and speaks an assortment of foreign languages, including Swedish. He also works part-time at Akamai Technologies, a local startup company. Quiet and self-effacing, he is a pleasure to work with. Coaching him has been a challenge: what an awful thing it would be to coach him into a worse status than his current state! Reid will be attending MIT in a few weeks.

The rest of the team also earned medals.

Tom Widland, a private school student from Albuquerque, New Mexico almost topped the silver medalists. A couple more test cases (out of over 100) and he would have had a gold medal. Tom is planning to take a year off to work in Spain before attending Harvard.

Vladimir Novakovsky, a student at Thomas Jefferson High School of Science and Technology in Virginia, also earned a silver medal. Vlad came to Finland after a short break in Moscow subsequent to his other silver medal performance in Antalya, Turkey at the International Physics Olympiad. Vlad is foregoing his senior year at TJ (having taken all the courses there) to attend Harvard this fall.

Steven Sivek, a rising senior at TJHSST, earned a bronze medal. Steven will continue his extracurricular activities at TJ this fall, sharing many of them with his identical twin brother. They include science bowl, computer club (complete with training and lectures for IOI and American Computer Science League competitions), and other academic-style clubs.

All-in-all, a super group of young men who have performed better than any team we've had in the past. I hope that each of you encourages those young people you encounter who indicate an interest in computer careers. We always need a larger number of competent and talented professionals!

USENIX is the sole sponsor of the USACO, a completely volunteer organization. I head the staff of half a dozen coaches while Don Piele, a professor at the University of Wisconsin-Parkside, handles all the administration and operational tasks. I know that everyone on the team appreciates USENIX's commitment and works hard to make all members proud of the team's accomplishments.

apropos

by Tina Darmohray

Tina Darmohray, co-editor of *;login:*, is a computer security and networking consultant. She was a founding member of SAGE.

tmd@usenix.org

I had a rather heated discussion with another system administrator the other day. It was Monday morning after the cutover to a new mail server, and we were encountering the usual glitches that go along with such an upgrade. I tripped across the first in the list of things we'd need to fix when I sent some email to one of the managers who was out on vacation. Shortly thereafter I got an error message in my mailbox saying that smrsh couldn't run the vacation program. I was tracking down the fix for vacation when the other administrator showed up at my cube. When I shared with him what I was trouble-shooting he launched into a diatribe of derogatory comments about sendmail saying that it was antiquated, overly complicated and, just like BIND, needed to be replaced.

It was probably the culmination of the morning, which already felt like a full day, but I took issue with his comments about sendmail and BIND. I agreed that sendmail's configuration file wasn't for the faint of heart, but insisted that programs like sendmail and BIND, indeed the IP protocols themselves, were not antiquated. In fact, I argued, they are on the short list of programs and protocols that have scaled to the size of the Internet and remain viable and robust; obviously an accomplishment worthy of respect.

This isn't the first time I've heard disparaging comments about these programs. For some reason, people love to hate them, and often the more publicly, the better. It's as if by vocally calling attention to perceived flaws in these programs, they feel they boost their own technical stature. Many times, however, the people making the disparaging comments don't have first-hand knowledge of the programs they're criticizing. Such was the case with my fellow admin; in the course of his ranting he revealed, "Postfix is so much easier and can do everything I need; I never could figure out that .cf file, anyway!"

I happen to like sendmail.cf files. Maybe it's because I like a challenge? But I can appreciate that others may prefer something simpler to use. What I've become less tolerant of is folks randomly criticizing some of these great examples of networking applications. The fact that these programs and protocols scale and perform today as they have is a tribute to their design and garners my respect on that account alone. Experience shows that it's not that easy to get this "right." Why, oh why, do folks want to berate the few examples which have?

;login:

EDITORIAL STAFF

EDITORS:
Tina Darmohray <*tmd@usenix.org*>
Rob Kolstad <*kolstad@usenix.org*>

STANDARDS REPORT EDITOR:
David Blackwood <*dave@usenix.org*>

MANAGING EDITOR:
Alain Hénon <*ah@usenix.org*>

COPY EDITOR:
Steve Gilmartin

TYPESETTER:
Festina Lente

PROOFREADER:
Lesley Kay

MEMBERSHIP, PUBLICATIONS, AND CONFERENCES
USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710
Phone: +1 510 528 8649
FAX: +1 510 548 5738
Email: <*office@usenix.org*>
<*login@usenix.org*>
<*conference@usenix.org*>
WWW: <*http://www.usenix.org*>

letters to the editor

Question on Liability

From Fernando Montenegro
fsmontenegro@hotmail.com

To John Nicholson

I was recently going through your article about liability in the April issue of *;login:* and I have a question:

If I understood correctly (I am not a lawyer), every tort claim in the US has to have four elements:

- Duty – the defendant has to have a relationship where he had to "care" for the plaintiff.
- Breach of duty – there has been "negligence" by the defendant on providing that care.
- Damage – there must have been some harm to the plaintiff.
- "Proximity cause" – I understand this to be the assertion that the breach of duty above had to be "close enough" to whatever caused the damages.

I understand how the first three elements can be quickly established in an information security setting. However, I am having trouble with the fourth ("proximity").

The example you listed in the article, with the store owner, deals with a physical entity (bullets). However, using the identity theft example, how does one establish the "proximity" when the goods are just information? Worse, information that could *potentially* come from all sorts of different places (magazine subscriptions, banks, credit reports, club memberships, etc.). Do the same rigorous standards for burden of proof that we see in criminal cases apply to civil suits? Would it be reasonable to expect an individual to prove the "proximity" of the negligent act in a Web site attack?

Thanks for a great article! I thoroughly enjoy reading about "real-life" issues relating to information security.

From: John Nicholson
John.Nicholson@shawpittman.com

Your understanding of the four elements of a tort claim is correct. And you are also correct in thinking that proximate cause is frequently the most difficult part of any tort claim. Ultimately, it is the finder of fact (the judge or jury) who decides whether a specific action satisfies each of the four elements. In the US, the standard of proof for civil claims is what's called a "preponderance of the evidence" standard, and it is basically a "more likely than not" standard (i.e., more than 50% likely). Therefore, what a plaintiff must do is convince a judge or jury that information released from a poorly secured server was the "proximate cause" of some harm, and the defense must convince the judge or jury that, among other things, (1) the information was not the cause, or (2) even if the information was the cause, the server was reasonably secured.

Identity theft may not be as easy an example as release of medical data. If you had some medical condition that you did not wish to be disclosed, and, as a result of unreasonably poor security, that information became public knowledge, you could have grounds for a tort claim if you suffer some damage – whether actual, direct damage or what is called "emotional distress."

There could also be significant direct impacts from an identity theft. The thief could incur debts for which you end up being held responsible. Also, since your credit rating may be damaged, it might be difficult for you to get loans, and you might have to spend lots of time and effort trying to correct the situation, which could also be quite stressful. So, there could be actual, direct damages, and there could be "emotional distress" related to an identity theft case.

In addition to damages suffered by the victim, the US system allows for what are called "punitive damages," which are intended to penalize the party at fault so that the party takes steps to prevent the problem from happening in the future.

The point of my article was to raise awareness of the potential impact that the absence of a clear security policy could have. From a corporate point of view, the reasonable potential for a lawsuit should be enough to at least cause people to think about establishing a reasonable security policy. In addition to the prospect of being taken to court, there is the potential for a very bad public relations situation (the public knowing not only that you had been hacked but that people were filing lawsuits claiming your security procedures were insufficient), which might be very bad for business.

Charity

From Greg Black
gjb@gbch.net

To Andrew Hume

I am one of the small number of non-American members of USENIX. I believe very strongly that USENIX should continue to provide a (small) proportion of its funds to charitable ends such as those discussed in your article in the June 2001 *;login:*.

I'd also like to see more reporting on the destinations of these charitable contributions; and on the outcomes from them.

We do try to publish news about recipients of USENIX grants, and also publish some of the results: see page 92 about USACO.
The Editors

conference reports

This issue's reports are on the 3rd USENIX Symposium on Internet Technologies and Systems (USITS '01), the GNOME Users and Developers European Conference (GUADEC 2001), the first Java™ Virtual Machine Research and Technology Symposium (JVM '01) and three reports on events dealing with clusters: the 5th Workshop on Distributed Supercomputing Scalable Cluster Software, the 1st IEEE/ACM International Symposium on Cluster Computing and the Grid, and the Large-Scale Cluster Computing Workshop.

OUR THANKS TO THE SUMMARIZERS:

For USITS '01:

Hari Balakrishnan for organizing and editing the reports.
Stergios Anastasiadis
Allen Miu
Anupam Rastogi
Alex C. Snoeren

For GUADEC 2001:

Martin Wahlén

For JVM '01:

Johan Andersson
Chiasen (Charles) Chung
Okehee Goh
Hughes Hilton
V.N. Venkatakrishnan

For the cluster events:

Al Geist
Craig A. Lee
Dana Skow
Alan Silverman
Joe Kaiser

3rd USENIX Symposium on Internet Technologies and Systems (USITS '01)

San Francisco, California

March 26-28, 2001

KEYNOTE

Interfaces Are Forever, or It's the Hole, Stupid

Scott Guthery, Mobile-Mind, Inc.

Summarized by Allen Miu

IT engineers are the civil engineers of the Internet. IT engineers build systems by bridging very different technologies. Therefore, technology providers must be very clever about building usable interfaces. However, building and dealing with interfaces are notoriously difficult tasks.

Technology providers often have a hard time predicting what the "killer apps" are. For example, Apple II was anticipated as a popular appliance for gaming and keeping recipes at home. However, the killer apps turned out to be home publishing and spreadsheets. The surprise highlights the fact that technology providers are often not the best application providers. Therefore, it is in the technology provider's best interest to create flexible interfaces that lay a versatile platform on which unanticipated killer applications may be created and evolve.

Even when there is a clear direction to build interfaces for creating powerful development platforms, subtle details can cause the greatest successes and failures. One such example is WAP (Wireless Access Protocol), which is an interface designed to bridge mobile telephony and the Internet. Thus far, the development effort has been focused on exposing the Internet to mobile-phone applications. Unfortunately, people have found it very difficult to implement traditional Internet applications such as Web browsers on the mobile phone for various reasons, such as the form factor of the mobile phone and limited bandwidth. However,

Scott Guthery

the application model will become much more powerful and interesting if we flip the interface and try to expose mobile telephony to the Internet. For example, imagine the possibilities of embedding a lightweight HTTP server and a Smartcard chip on the mobile phone. Instantly, the mobile phone becomes a mobile authenticator for the user to conduct transactions on the Internet.

After showing various examples illustrating the importance of interfaces, the talk concluded with an outlook for building future "killer" applications on mobile devices. The speaker metaphorically described mobile computers as remote controls for reality. We should concentrate on building new interfaces that exploit the inherent real-time, interactive capabilities provided by any mobile platform. In fact, many existing applications can already take advantage of the mobile platform. For example, instant text messaging is a long-dominant Internet application. It has been adopted by the GSM network and became one of the most popular applications running on today's cell phones. Online auctioning is another highly successful mobile extension of a popular Internet application. NTT has recently launched a system that under-

writes mobile online auctioning transactions for the cell phone users connected to their DoCoMo network. Like the Internet counterpart, the DoCoMo online auctioning system became a hugely popular application among cell phone users.

We are just beginning to provide "killer apps" for mobile devices. There are still numerous desktop applications that do not have mobile counterparts because we still lack the appropriate "interfaces" that allow engineers to create mobile extensions of these applications.

SESSION: FLEA MARKET

Summarized by Anupam Rastogi

The Age Penalty and Its Effect on Cache Performance

Edith Cohen, AT&T Labs – Research; Haim Kaplan, Tel-Aviv University

This paper addresses the issue of the cache-age penalty in wide area networks. This issue arises when there is a hierarchy of caches between the server and end users of content. Such hierarchies are becoming more common today, with proxy caching, reverse proxies, and Content Delivery Networks (CDNs) being increasingly deployed. Caches determine an expiration time for a cached copy by computing its freshness lifetime and its age. A copy becomes stale when its age exceeds its freshness lifetime, and it must be refreshed, even though it may not have been modified at the higher level. When we have hierarchies of caches, lower levels get data with positive age and, thus, a shorter time-to-live compared to what it would have been if the data had come directly from the origin server. This imposes a penalty, since the cached data would now become stale sooner. This is termed "the age penalty" in the paper.

The age penalty is measured by comparing the performance of a low-level cache that gets its data from another cache with the performance of the same cache if it

Tom Anderson, USITS '01Program Chair

received its data from the origin server. Simulations have been carried out to measure the impact of cache penalty on performance. Trace-based simulations are also used to measure the extent of age penalty for content served by content delivery networks and large caches. The age penalty has been shown to be significant in some cases.

The age penalty can be avoided by maintaining strong consistency between high-level caches and the origin server. But this is expensive and difficult to implement. The future work includes two possible approaches: source selection, where low-level caches can select where they forward a request on a miss, and rejuvenation, where pre-term validation of selected copies is used to decrease age.

Online Markets for Distributed Object Services: The MAJIC System

Lior Levy, Liad Blumrosen, and Noam Nisan, The Hebrew University of Jerusalem

Blumrosen described an infrastructure that performs online auctions for computer services over distributed-object systems. An implementation of such a system over Jini was also presented.

The motivation for the need for such services is that there are many distributed resources on the Internet which belong to different organizations. These organizations must have a motivation to share these resources. This is realized by having an infrastructure where services are paid for (economic paradigm). Examples of some such existing systems are Spawn, Popcorn, and SuperWeb.

The distributed-object paradigm entails that systems on the network encapsulate sharable resources in well-defined interfaces, which can be accessed using Remote Procedure Calls (RPC) / Remote Method Invocation (RMI). The distributed-object paradigm and the economic paradigm are combined to get the new infrastructure, where services are offered for a price, and "customers" can "buy" the service with the best combination of service parameters and price. A service marketplace functions as the object-request broker. For this, each service type has parameters defining the parameter space. Sellers provide quote functions for parameters, which give the price for providing a service for the given parameters. Buyers, or service users, employ a utility function, based on service parameters, which measures the utility of a service for the buyer. Buyers also provide a parameters search engine, which attempts to find the optimal parameters, given the quote functions.

The system functions as follows: the market holds current quotes from all sellers; when it receives a request from a buyer, it attempts to match the request with the best seller and choose the best parameters using the utility function and parameters search engine provided by the buyer.

It is claimed that such economic systems can also provide load balancing automatically if designed correctly. Also, the system architecture allows avoidance of inefficient allocations caused by untruthful sellers.

The MAJIC (Multi-Parameter Auctions for Jini Components) system was presented. MAJIC is built on top of Sun's Jini platform and implements the basic architecture of the system described above. Performance studies showed 15%

overhead per request due to the MAJIC system in a high-load scenario.

More information is available at : *http://www.cs.huji.ac.il/~majic*.

INVITED TALK

Search Engine Experience and Internals

Mike Burrows, Compaq Computer Corporation Systems Research Center

Summarized by Alex C. Snoeren

Mike Burrows gave two related short talks. The first described the internals of a general library he implemented for indexing text, which formed the basis of the search engine known as AltaVista. The second talk was a humorous retrospective on the issues encountered while deploying AltaVista.

He began the first talk by enumerating a set of goals he had in mind for a general purpose indexing library. He made special note that he did not originally have AltaVista in mind when designing the library. One of the key features of the library was its ability, unlike previous similar libraries, to support online updates, that is to continue to support queries during updates, but still provide reasonable update performance.

He described the flat-file storage mechanisms employed by the library, detailing the tricks necessary for rapid processing. In particular, he showed that by supporting a small number of basic operations, the library is able to support an arbitrary set of conjuncts and disjuncts while processing sequentially through the data file. Hence his library provides stream abstractions called Indexed Stream Readers (ISRs) which are powerful enough to provide full-search functionality. Avoiding random access provides enormous implementation efficiency, enabling him to fully utilize the deeply pipelined multi-stage Alpha architecture. To prove his point, he presented a single slide of a highly optimized Alpha assembly language which provided all the basic searching functionality.

After delving into the gory details of how online updates are supported by dividing the dataset into tiers of hash buckets, Burrows presented some (slightly outdated) performance metrics that showed that the search library was entirely CPU bound, and did not fully utilize the memory bus of the Alpha in use, hence additional performance gains could be achieved by adding processors.

Redundancy was the theme of the second talk, in which he described the actual implementation of AltaVista (as of a few years ago), which utilized multiple Alpha workstations as front ends, a few 4-10 CPU Alpha servers as back ends (the system was purposefully designed to showcase DEC's flagship big iron hardware, each of which was configured with 8GB of RAM and 150GB of disk space), and connected them with a FDDI switch. The back-end machines were clustered in groups of 4 to 10 machines, each of which shared their own copy of the index.

Burrows described the great pains taken to ensure failure-free operation of every major subsystem. The hard drives were managed by RAID controllers with spare disks; hot-spare workstations could automatically take over the IP addresses of downed front ends; a cold-spare FDDI switch was kept on-site at all times; each front end could dynamically select alternate back ends; and the entire site could failover to the backup site with a manual DNS change.

Burrows was quick to point out, however, that the seemingly impressive amount of replication was far from sufficient in practice. He wound through a comical tale of disasters large and small, ranging from hardware issues such as the sad truth about self-repairing RAID controllers (performance drops by a factor of two during reconstruction), unexpectedly high file-system corruption rates, and poorly designed interface cards whose pins were all too easy to bend, to software issues such as poor testing, design flaws, and operator error caused by poor interfaces. In addition, he pointed out that spammers and denial of service attacks became quite common as AltaVista grew, and he eventually began spending a great deal of time simply dealing with users abusing the system.

Burrows concluded by calling AltaVista a "success disaster." Generally never staffed by more than two people at a time, the search-engine load grew at a rate of 10-15% per week for over a year, a rate which they found extremely difficult to support. In retrospect, Burrows suggested that if he were to design AltaVista again from scratch, he would prefer to use lots of smaller machines as a back end instead of the small clusters of larger ones.

SESSION: ADAPTATION

Summarized by Stergios Anastasiadis

CANS: Composable Adaptive Network Services Infrastructure

Xiaodong Fu, Weisong Shi, Anatoly Akkerman, and Vijay Karamcheti, New York University

The CANS architecture injects application-specific components in the data path between applications and services. This allows seamless integration of services and devices in diverse networking environments.

The data path notion is extended to include application-specific functionality in the form of different components: drivers and service. The drivers are soft-state mobile code modules that apply operations to data streams passively. Besides the type model used, their efficient composition and reconfiguration requires adherence to restricted interfaces. Services, on the other hand, could be legacy components which use any standard Internet protocol. They could maintain persistent state and do not have to adhere to standard interfaces. Legacy applications can be integrated through an interception layer.

Dynamic changes in system characteristics are handled by three different modes of adaptation. Intracomponent adaptation occurs when services or drivers detect and adapt to environment change by themselves. Data path reconfiguration and error recovery include localized changes to the data path involving insertion, deletion, and reordering of drivers. Replanning is the response to large-scale system variations that require tearing down existing data paths and constructing new ones. The runtime overhead of the system is shown to be negligible.

Related information is available at the project Web site: *http://www.cs.nyu.edu/pdsg*.

Dynamic Host Configuration for Managing Mobility between Public and Private Networks

Allen Miu, MIT Laboratory for Computer Science; Paramvir Bahl, Microsoft Research

The CHOICE network provides authenticated users with high-speed wired or wireless access to the Internet. It supports secure, customized, and accountable services to possibly unknown customers, and it operates seamlessly as mobile clients move across different public and private networks. The underlying protocol is called Protocol for Authorization and Negotiation of Services (PANS).

In a CHOICE network, IP addresses are leased to potential clients through a standard DHCP server, while an authentication database is globally accessible through the Internet. The PANS Authorizer provides controlled access to the authentication service and determines a customized service policy based on the user's credentials. Once the user has been authenticated, the PANS Authorizer generates a session key that is distributed securely to both the PANS Client and the PANS Verifier. From then on, the PANS Client cryptographically tags every transmitted packet with the given session key and sets the PANS Verifier as the default gateway to access the Internet. The PANS Verifier enforces service policy by checking the tag of every transmitted packet and accounts for the client's resource usage by keeping a log of traffic generated by each user.

Mobility between public and private networks is managed by the PANS Autoconfiguration module, which offers service discovery, bootstrapping, protocol configuration, and key management. Configuration parameters are transmitted to the client modules using a beaconing technique that is based on lightweight periodic broadcasting. Multiple verifiers can be used for managing the active key set in order to provide fail-over operation and load balancing. Scalable key distribution is achieved by migrating keys on demand as clients roam between different subnets. Finally, several techniques are described for making denial of service (DoS) and hijacking attacks difficult and detectable.

Details can be found at the project Web site: *http://www.mschoice.com*.

SESSION: COOL HACKS

Summarized by Anupam Rastogi

Alpine: A User-Level Infrastructure for Network Protocol Development

David Ely, Stefan Savage, and David Wetherall, University of Washington

This work addresses the issue of making the task of modifying the network protocol code simpler and less tedious by moving the network stack to user space for development. The premise is that kernel development is a pain, the main factor being the time required for recompiling and rebooting. The networking protocols are currently in the kernel, thus making changes to the network code a pain, too.

There are previous applications like Entrapid, OSKIT, and X-Kernel which address similar issues, but these require changes to the kernel, the applications, or the networking stack. Alpine requires no changes to any of these.

In Alpine, the socket, TCP, and IP layers are moved into a library. A "faux Ethernet" layer is inserted below the IP layer. To the IP layer, it appears like a normal Ethernet driver, but it sends packets to the actual interface using raw sockets and receives using a packet capture library.

Ely also described various challenges in the implementation, involving virtualizing kernel services and virtualizing the system-call interface.

Alpine is shown to perform almost as well as the kernel in terms of throughput up to a bandwidth of around 10Mbps, after which the performance gap starts widening.

The major benefits of this infrastructure are easy source-level debugging of code and quick turnaround between revisions. Alpine can be a useful environment for class projects and application-specific protocol extensions.

Some of the limitations of the current version of Alpine are: it only works for TCP and UDP; the number of sockets usable by Alpine is limited to 100; fork() calls in the application code are not currently supported; and it requires root privileges.

More information about Alpine is available at *http://alpine.cs.washington.edu/*.

Measuring Client-Perceived Response Times on the WWW

Ramakrishnan Rajamony and Mootaz Elnozahy, IBM Austin Research Lab

An important factor that affects the success of a WWW service is the Client-Perceived Response Time (CPRT). If this time is high, the user may get bored and go away. CPRT is the gap between click time and view time, and is the sum of the network delay, the server processing time, and the rendering time taken by the browser. Quantitative information about response times can be important to businesses and Web sites, and may be used to determine whether an improved server

or network infrastructure is required. This paper presents a framework for measuring the actual response time perceived by customers as they access a Web service. The scheme uses HTML and JavaScript, which are supported by most browsers and load fast. An "instrumented" entry to a Web page causes embedded JavaScript within the downloaded page to execute on the client. The client-side script then notes the time at which a subsequent request is made and records it locally, permitting JavaScript downloaded along with the Web page response to compute the delta between the "click" time and the "fully loaded" time. The response time is then sent by the script to a predetermined record-keeping Web site, which can collect the data and process it.

The system has been implemented for the "Wondering Minstrels: Poem of the Day" Web site (*http://www.cs.rice.edu/~rrk/minstrels.html*). Assuming the attention time, i.e., the time after which the user gets bored, to be four seconds, around a sixth of the response times were found to be more than the attention time. The response times have further been studied with respect to accessing top level domains and time of the day.

The limitations of the outlined scheme are that response time can only be measured for an instrumented entry to a Web page — response times to MIME types other than objects embedded within HTML cannot be measured – and that it works only if JavaScript is enabled.

The overhead of instrumentation of Web pages is about 200 bytes per page and 2KBytes per site. The script itself is downloaded only once per site, and it never expires. It is claimed that the extra code has minimal effect on load time.

The advantages of the scheme over other related ones are that no changes to the server, browser, or proxies are required (only JavaScript support in the browser is required) and that the CPRT can be sent to any third party.

GUADEC 2001

Summarized by Martin Wahlén

GUADEC (the GNOME Users and Developers European Conference) is an annual conference/workshop whose purpose is to focus the effort of developers on users' needs. GNOME is the GNU Network Model Environment, a free desktop and component model for X.

The main goals for this year's GUADEC, held in April in Copenhagen, Denmark, were to prepare and set the expectations for what should be in GNOME 2.0, which is scheduled for release at some later time. GNOME 1.4 was released just before GUADEC 2001, and an effort was made to inform the developers and users of how to make the most of the features in that release at this year's conference.

To make that happen several of the representatives of other desktop projects were invited to GUADEC. Matthias Ettrich from the KDE project gave a very good keynote address. Everyone was impressed by what KDE was able to do with kdevelop. The general consensus was that we share some of the basic technologies with the other desktops, but we should be able to share much more. We also need to make clear that both KDE and GNOME applications can work together; this is particularly important to independent software vendors as they observe the current fragmentation of the X desktop market.

Havoc Pennington representing the GNOME Foundation gave the wrap-up speech. He concluded that meeting in real life had been productive and that GNOME had made progress during GUADEC. The inter-operability BoF had been productive according to Havoc, and many of the issues were looked at and addressed. The KDE people were very helpful, and it was decided that the GNOME project should use the same techniques as KDE does.

Two groups of the GNOME project were particularly successful at this year's conference. The internationalization developers and localization teams were able to decide on the core technologies to be used in GNOME 2.0. The localization teams developed some new techniques for quality management, and the documentation team, together with the internationalization developers, focused on using XML (tools) for structured information.

GNOME has 1.5 million users, but where do we go from here? We wish to expand beyond the technical desktop market, which is estimated to be just 5% of the global desktop market. In order to expand, GNOME needs to be more usable, so there were three presentations on how to make better user interfaces. Shockingly, the people talking about user interfaces came to the conclusion that users don't really want five clock applets, especially not one that presents the time in binary.

Some MPEGs from the sessions can be obtained from *ftp://ftp.dkuug.dk/pub/GUADEC2001/*.

First Java™ Virtual Machine Research and Technology Symposium (JVM '01)

Monterey, California
April 23-24 2001

Keynote: Virtual Machines, Real Time

Greg Bollella, Sun Microsystems; David Hardin, aJile Systems

Summarized by V.N. Venkatakrishnan

The invited talk of the conference was the presentation on real-time virtual machines. Greg Bollella introduced the topic by presenting the scenario in embedded systems today. The presence of networking everywhere and the demand

for building large, complex systems are two of the reasons for the inevitability of increasing software complexity. In this scenario, the two paradigms of system development – the one for business, personal, and Web computing and the other for device, scientific, and industrial computing – are moving toward a collision in this era of computing. Greg pointed out function migration from large devices to the hand held is the emerging trend. The real-time factor in hand-held devices is important because customers are used to system response in real time (e.g., a phone). Greg then presented a case example of JPL's mission data system and explained the role of real-time software in such an example.

Having motivated the listeners on the subject, Greg explained the technical aspects of a real-time system, using a

l to r: Saul Wold, David Hardin, Greg Bollella

car's electrical components as an example. He clarified the popular myth that a real-time system is not a fast system, but a system which includes time as an integral part of its computation.

There are several reasons why the Java language is an ideal choice for implementing such systems. As an advanced OO language, Java has a large set of libraries, a common set of APIs, an automatic memory management, and belongs to all the layers in software abstraction. A real-time JVM would thus support building various embedded system software, not just applications. But there are numerous barriers to achieving this: application-level unpredictability, hardware latencies, x86 context switch latencies, and inherent unpredictability due to various functions in the JVM such as scheduling and garbage collection.

In David Hardin's presentation, the approach taken by aJile is to implement JVM directly with simple, low-cost, low-power hardware. JVM bytecodes are native instructions and this supports real-time threads in hardware using Java thread primitives as instructions. This enables the entire system to be written in Java, with no C code or assembly required. Such an implementation has provided the fastest real-time Java performance.

Greg continued the discussion with the Java real-time specification, emphasizing such issues as scheduling, memory management, concurrency, and physical memory access. The implementation of JSR was scheduled for presentation to the expert group by April 30, and final release of the specification is in progress. The discussion culminated with a spectacular demo-presentation of piano playing robot hands controlled by two different real-time virtual machines.

After attending the talk, one was convinced that despite the various problems posed by hardware, OS, and JVM, real-time applications can be successfully built using Java.

Further information on this project can be obtained by contacting Greg at *greg.bollella@east.sun.com.*

SESSION: CODE GENERATORS

Summarized by V.N. Venkatakrishnan

The Java HotSpot Server Compiler

Michael Paleczny, Christopher Vick, and Cliff Click, Sun Microsystems

How can the performance of JVM improve through optimization of frequently executed application code? Michael Paleczny's talk addressed this research question through the presentation of the Java HotSpot Virtual Machine. The client version provides very fast compilation times and a small footprint with modest levels of optimization. The server version applies more aggressive optimizations to achieve improved asymptotic performance. These optimizations include class-hierarchy-aware inlining, fast-path/slow-path idioms, global value-numbering, optimistic constant propagation, optimal instruction selection, graph-coloring register allocation, and peephole optimization.

Michael described the runtime environment that both the compiler and generated code execute within, followed by the structure of the server compiler. Then he described some of the phases of compilation, discussing solutions for specific language and runtime issues. Finally, he outlined the directions for future work on the compiler which include range checks, loop unrolling, instruction scheduling, and a new inline policy.

Further information about this work can be obtained from *michael.paleczny@eng.sun.com*

Can a Shape Analysis Work at Runtime?

Jeff Bogda, Ambuj Singh, UC Santa Barbara

A shape analysis is a program analysis that can identify runtime objects that do not need to be placed in the global heap and do not require any locking. It has been shown through previous research that these two optimizations speed up some applications significantly. Since the shape analysis requires a complete call graph, it has not been implemented in the JVM.

After illustrating the purpose and some history of shape analysis, Jeff Bogda's talk went on with the description of his approach to build an incremental shape analysis to analyze an executing program. The analysis is done through an experimental framework to which the execut-

ing application is instrumented so that the analysis is performed at key points in the program execution. Jeff then described three approaches to performing shape analysis: immediate propagation, where the analysis is done before the method execution; delayed propagation, which delays the analysis untill an appropriate time; persistent propagation, which utilizes results from previous executions.

Jeff discussed the various trade-offs in these approaches. The experiments suggest a strategy which consults the results of the previous executions and delays the initial analysis untill the end of the first execution.

For more information on this work, the reader may visit
http://www.cs.ucsb.edu/~bogda
or contact Jeff at *bogda@cs.ucsb.edu*.

SableVM: A Research Framework for the Efficient Execution of Java Bytecode

Étienne M. Gagnon, Laurie J. Hendren, McGill University

SableVM is an open-source virtual machine for Java intended as a research framework for efficient execution of Java bytecode. The framework is essentially composed of an extensible bytecode interpreter using state-of-the-art and innovative techniques. Written in the C programming language and assuming minimal system dependencies, the interpreter emphasizes high-level techniques to support efficient execution.

Saul Wold presenting Best Student Paper Award to Étienne Gagnon

Sable VM introduces several innovative ideas: a bidirectional layout for object instances that groups reference fields sequentially; this allows efficient garbage collection. It also introduces a sparse interface virtual table layout that reduces the cost of interface method calls to that of normal virtual calls. Another important feature is the inclusion of a technique to improve thin locks by eliminating busy-wait in the presence of contention. In his talk, Gagnon presented SPEC benchmarks that demonstrated the efficiency of this research framework.

This paper won the best student paper award at the conference. Further details on this work can be obtained from the author (*egagnon@j-meg.com*) and at the Web site (*http://www.sablevm.org/*).

SESSION: JVM INTEGRITY

Summarized by V.N. Venkatakrishnan

Dynamic Type Checking in Jalapeño

Bowen Alpern, Anthony Cocchi, and David Grove, IBM T.J. Watson Research Center

Jalapeño is a JVM for servers. In any JVM, one must sometimes check whether a value of one type can be can be treated as a value of another type. The overhead for such dynamic type checking can be a significant factor in the running time of some Java programs. Bowen Alpern's talk presented a variety of techniques for performing these checks, each of these tailored to a particular restricted case that commonly arises in Java programs. By exploiting compile-time information to select the most applicable technique to implement each dynamic type check, the run-time overhead of dynamic type checking can be significantly reduced.

Bowen introduced the topic by going over the Java type system and the basic types. He then presented the main contributions of this research. This work suggests maintaining three data structures operationally close to every Java object. The most important of these is a display of superclass identifiers of the object's class. With this array, most dynamic type checks can be performed in four instructions. It also suggests that an equality test of the runtime type of an array and the declared type of the variable that contains it can be an important short-circuit check for object array stores. Together, these techniques result in significant performance improvements on some benchmarks.

This code that implements these techniques is not available in the public domain. The system is available for academic purposes; one may contact the author at *alpern@watson.ibm.com*. More information about the project is available at *http://www.research.ibm.com/jalapeno.*

Proof Linking: Distributed Verification of Java Classfiles in the Presence of Multiple Class Loaders

Philip W.L. Fong, Robert D. Cameron, Simon Fraser University

Computations involving bytecode verification can be expensive. To offload this burden within Java Virtual Machines (JVM), distributed verification systems may be created. This can be done using any one of a number of verification protocols, based on such techniques as proof-carrying code and signed verification by trusted authorities. Fong's research advocates the adoption of a previously proposed mobile code verification architecture, proof linking, as a standard infrastructure for performing distributed verification in the JVM. Proof linking supports various distributed verification protocols. Fong also presented an extension of this work to handle multiple class loaders.

Further details on this work can be obtained from the author at *pwfong@cs.sfu.ca*.

JVM Susceptibility to Memory Errors

Deqing Chen, University of Rochester; Alan Messer, Philippe Bernadat, and Guangrui Fu, HP Labs; Zoran Dimitrijevic, University of California, Santa Barbara; David Jeun Fung Lie, Stanford University; Durga Mannaru, Georgia Institute of Technology; Alma Riska, William and Mary College; and Dejan Milojicic, HP Labs

Deqing Chen presented a series of experiments to investigate memory error susceptibility using a JVM and four Java benchmark applications. Chen's work was woven around the fact that except for very high-end systems, little attention is being paid to high availability. This is particularly true for transient memory errors, which typically cause the entire system to fail. To bring systems closer to mainframe class availability, addressing memory errors at all levels of the system is important.

The experiments were done using the technique of fault injection. To increase detection of silent data corruption, JVM data structure checksums were examined. The results that were presented indicated that the JVM's heap area has a higher memory error susceptibility than its static data area and that up to 39% of all memory errors in the JVM and application could be detected. Such techniques will allow commodity systems to be made much more robust and less prone to transient errors.

For further information on this work, the author can be contacted by email at *lukechen@cs.rochester.edu.*

WORK-IN-PROGRESS REPORTS

Summarized by Chiasen (Charles) Chung

Implementing JNI in Java for Jalapeño

Ton Ngo, Steve Smith, IBM T.J. Watson Research Center

This talk addressed the advantages and implication of JNI implementation in Jalapeño, which is a JVM written in Java developed at the IBM T.J. Watson Research Center.

In order for the JNI functions to reuse the same internal reflection interface in Jalapeño, it is written in Java rather than in C as might be expected. This approach has two benefits: 1) changes in Jalapeño are transparent to the JNI implementation; 2) despite being a native interface, the JNI functions are portable to any platform where Jalapeño is installed.

When a native method is invoked in Jalapeño, a special static method is called to resolve the native method with the corresponding native procedure. JVM then generates the prologue and epilogue to establish the transition frames from Java to C code. The code entry from C to Java is through JNI functions defined in the specification. In Jalapeño, these are methods collected in a special Java class, and they are compiled dynamically, with special prologue and epilogue to handle the transition.

To resolve references in Jalapeño JNI, each Java object to be passed to a native code will be assigned an ID and then stored in a side stack. The native code accesses these objects based on their IDs. In a garbage collection cycle, Jalapeño JNI checks for live references in the native stack frames against the side stack.

The implementation of JNI on the PowerPC/AIX platform has been completed, while the Intel/Linux platform is still under development. The group is currently researching threading for long executions of native methods and issues concerning interaction between Java and native programs. More information can obtained at *http://www.research.ibm.com/jalapeno.*

JaRec: Record/Replay for Multi-threaded Java Programs

Mark Christiaens, Stijn Fonck, Dries Naudts, Michiel Ronsse, Koen De Bosschere, Ghent University

Debugging multi-threaded programs is difficult because thread races are hard to reenact, thus introducing non-determinism into the debugging. To solve this problem, Mark Christiaens suggested a two-phase "record/replay" technique.

JaRec is a program that records and replays the interaction sequence between threads in Java programs using two (enter and exit) monitors. Every thread has a Lamport clock which is incremented when the thread leaves or enters a monitor. During the record phase, a trace for the interaction between the threads based on this clock value is generated.

These Lamport clock values are recorded in the trace file as a timestamp. By forcing the order in which threads enter the monitors base on this timestamp, the thread execution and interaction sequence can be reproduced exactly. Synchronization is forced by waiting for a thread to report.

Both the record and replay phase in JaRec are implemented using the Java Virtual Machine Profiler Interface. The record phase is near completion and the group is currently implementing the replay phase of the system.

Kaffemik – A Distributed JVM Featuring a Single Address Space

Johan Andersson, Trinity College

Kaffemik is a scalable distributed JVM based on Kaffe VM. It is designed to run large-scale Java server applications by using clustered workstations. The goal of this project is to investigate scalability issues in a distributed JVM and to improve performance in large-scale Java applications.

Kaffemik is designed as a single JVM abstraction over the cluster by imple-

menting a single address space architecture across all the nodes based on the global memory management protocol. On top of the common local thread operations, Kaffemik supports internode synchronization and remote-node thread creation.

Preliminary benchmark results show that Kaffemik starts local threads significantly faster than remote threads, but is much slower starting local threads compared to Kaffe. Remote threads are even more expensive due to the overhead induced by page-faults.

The current Kaffemik prototype shows that it is costly to implement distributed applications over high-speed clusters on single address space architectures. The next step in the project is to implement a two-level (global and local) memory allocator. A garbage collector for the global memory is also needed, but it is not addressed in this paper.

A Java Compiler for Many Memory Models

Sam Midkiff, IBM T.J. Watson Research Center

The Java memory model is heavily coupled into the programming language.In hopes of overcoming its various flaws, a new memory model has been proposed. Instead of fixing the memory model, this talk focused on defining the memory model as part of a property of the code being compiled.

Sam Midkiff proposes a Java compiler that accepts a ".class" file annotated with a memory-model specification. The compiler first represents the program using the Concurrent Static Single Assignment (CSSA) form. Escape analysis is applied to determine the order in which variables should be accessed according to the memory model. Next, the program represented in the CSSA graph is optimized. Finally, the compiler produces an executable that maps the program onto the underlying hardware consistency model.

This work explores the development that supports programmable memory models. Relative efficiency of different memory models running on a common hardware can be investigated. More information can be obtained from *http://www.research.ibm.com/people/m/midkiff/*.

State Capture and Resource Control for Java: The Design and Implementation of the Aroma Virtual Machine

Niranjan Suri, University of West Florida

Aroma VM is a research VM designed to address some of the limitations of current Java VMs. The capabilities for Aroma were motivated by the needs to mobilize agent systems and distributed systems.

Aroma provides two key capabilities: the ability to capture the execution state (of either the complete VM or individual threads) and the ability to control the resources used by Java programs running within the VM. The state capture capabilities are useful for load-balancing and survivable systems. The resource-control capabilities are useful for protecting against denial of service attacks, accounting for resource usage, and as a foundation for quality of service. Aroma currently provides both rate and quantity controls for CPU, disk, and network resources.

There is no Just-in-Time compiler for Aroma currently, but there are plans to integrate freely available JIT compilers (such as OpenJIT) in the future. More information on Aroma VM can be obtained from *http://nomads.coginst.uwf.edu/*.

OpenJIT2: The Design and Implementation of Application Framework for JIT Compilers

Fuyuhiko Maruyama, Satoshi Matsuoka, Hirotaka Ogawa, Naoya Maruyama, Tokyo Institute of Technology; Kouya Shimura, Fujitsu Laboratories

OpenJIT2 is a JIT compiler for Java written in Java that is based on "open compilers" construction technique. It not only serves as a JIT compiler but also as an application framework for JIT compilers. This framework allows multiple coexisting JITs to compile different parts of a program.

In the OpenJIT system, each instantiated compiler is a set of Java objects that compiles at least one method. The selection of methods to be compiled is determined through an interface that is based on method attributes. If the attribute does not specify a particular compiler (a set of compilet objects) to be used, the default baseline compiler will be selected.

Both baseline compiler and compilets are constructed using the OpenJIT2 framework and class library. Without the limitations of OpenJIT1's relatively simple internal structure, OpenJIT2 uses complex compiler modules to carry out analysis, program transformation, and optimization during compilation. The preliminary result shows that the baseline compiler will have reasonable compilation speed as an optimizing compiler compared with IBM's jitc and Jalapeño's optimizing compiler.

The first version of OpenJIT2 is expected to be completed by the second quarter of 2001. Once OpenJIT2 is complete, a more comprehensive runtime performance will be evaluated.

SESSION: THREADING

Summarized by Okehee Goh

An Executable Formal Java Virtual Machine Thread Model

J. Strother Moore and George M. Porter, University of Texas at Austin

This presentation describes a research project in which formal methods are applied to Java Virtual Machine (JVM). "Formal methods" is the idea of using mathematics to model and prove things about computing systems. Certain aspects of the JVM are modeled, including classes, objects, dynamic method resolution, and threads. A benefit of

modeling software in a mathematical notation is that theorems can be proved about the model. These proofs can be checked mechanically via a theorem prover. This paper discusses several such theorems about the JVM and byte-code programs for it. The theorems were proven with the ACL2 theorem prover.

ACL2 (A Computational Logic for Application Common Lisp) is a theorem prover for a functional programming language based on Common LISP. The JVM is modeled in ACL2 by defining a simulator for it. The state of the JVM consists of three components, including a collection of threads, a heap, and a class table. The semantics of each bytecode is represented as a function that transforms the state.

There are certain differences between this model and the JVM. For example, the model does not support bounded arithmetic or exceptions. Many such features were omitted to make it easier to explore alternative modeling and proof techniques. There is ample evidence from other ACL2 case studies that such features can be added without unduly complicating the analysis.

Complicated features of JVM bytecode programs, such as thread synchronization, can be analyzed using this mathematical model. Eventually, it should be possible to prove properties about the JVM itself, such as that the bytecode verifier is correct. Because the JVM is a very good abstraction of Java, models such as this will eventually permit mechanically checked correctness proofs about Java software.

More details about ACL2 are available at *http://www.cs.utexas.edu/users/moore/acl2* The case studies using ACL2 are at *http://www.cs.utexas.edu/users/moore/publications.*

TRaDe: A Topological Approach to On-the-Fly Race Detection in Java Programs

Mark Christiaens and Koen De Bosschere, ELIS, Ghent University, Belgium

The worst type of bug occurring in multi-threaded programs is a data race, which occurs when multiple threads execute while they modify a common variable in an unordered fashion. Normally it is hard to find a data race because they are non-deterministic and non-local.

TRaDe models the ordering of instructions performed by threads through the use of vector clocks. To detect data races, an access history for every object is constructed. When a new read or write operation occurs, it is compared to the previous operations to uncover data-race conditions. However, because the size of each vector clock is proportional to the number of threads, the memory and time consumption is very costly. One way to minimize this cost is to reduce the number of objects for which an access history must be maintained. Objects are distinguished into two types: local objects accessible to one thread and global objects accessible to several threads. Because "global objects" have the potential to be involved in a race, access to those objects must be checked, and the JVM instructions that can change the topology of the object interconnection graph must be observed.

Relative to the benchmark created by using an implemented TRaDe method in the Sun JVM1.2.1, TRaDe is 1.62 times faster than existing commercial products with comparable memory requirements.

The overhead of data-race detection is still large when compared to normal execution. The authors plan to reduce this gap, applying static analysis techniques such as "escape analysis."

SESSION: JVM POTPOURRI

Summarized by Johan Andersson

The HotSpot Serviceability Agent: An Out-of-Process High-Level Debugger for a Java Virtual Machine

Kenneth Russell, Lars Bak, Sun Microsystems

This talk demonstrated a really useful Java debugging tool, built with the HotSpot Serviceability Agent (SA). This is a set of APIs for the Java programming language, developed to help developers recover to a high-level state from a HotSpot JVM or core file, to make it possible to examine high-level abstract data types. When examining a JVM with a traditional C/C++ debugger, all this high-level information is gone, since these debuggers only deal with raw bits.

The SA can attach a remote process or a core file, read remote-process memory, and symbols lookup in remote processes. In principle, the Solaris version of SA launches a native debugger called dbx to actually interface with a remote process. It then loads a core file or attaches to a running HotSpot JVM process. This allows transparent examination of either live processes or core files, which makes it suitable to debug the JVM itself or Java applications. In order to examine the high-level data types in Java, the APIs in the SA mirrors the C++ structures found in the HotSpot JVM.

Kenneth Russell demonstrated the features found in the SA's APIs, which seemed to be very useful. It was very easy to traverse the heap and the stack, get histograms of allocated objects, and look up symbols.

In the future, the SA APIs, which are currently available for Solaris and Windows, will be ported to Linux. Russell said the APIs haven't been included in the JDK yet, but they are working on making this technology available for end users. The SA sources are currently available to licensees in the HotSpot source bundles.

More Efficient Network Class Loading through Bundling

David Hovemeyer, William Pugh, University of Maryland

David Hovemeyer presented bundling, a technique for transferring files over a network. Files that tend to be needed in the same program execution and that are loaded close together are placed together into groups called bundles. Hovemeyer presented an algorithm to divide a collection of files into bundles based on profiles for file-loading behavior. The main motivation for bundling is to improve the performance of network class loading in Java, by transferring as few bytes as possible to make best use of available bandwidth. This is very useful in areas of wireless computing, where bandwidth is a scarce resource.

Before Hovemeyer introduced the bundling algorithm, he discussed the alternatives. The first alternative involves downloading individual files: no unneeded files are transferred, but for each file that is, the cost is high in terms of network latency. The other alternatives are to use monolithic archives such as JAR, thus risking transfer of unwanted files, or to use individual-class loading with on-the-fly compression, which can be time-consuming.

Hovemeyer and Pugh's bundling approach is a hybrid of the above alternatives, combining the advantages of each. The collection of files making up the application is divided into bundles, which are then compressed. The basic idea is to avoid files that are not used and to transfer files to match the order of request by the client. The problem is to divide the collection of files into bundles. To solve this, Hovemeyer talked about establishing class-loading profiles, which can be determined by using training sets of applications to record the order and time at which each class was loaded during execution. The bundling algorithm then uses this information to group the files into bundles, according to the average use in the class-loading profiles.

The experimental results indicated that bundling is a good compromise between on-demand loading and monolithic archives. The results also showed that bundling is no worse than the JAR format, when used on an application not included in the training set.

The bundling algorithm is described in detail in the paper. Links to related research done at the University of Maryland can be found at *http://www.cs.umd.edu/~pugh/java/*.

Deterministic Execution of Java's Primitive Bytecode Operations

Fridtjof Siebert, University of Karlsruhe; Andy Walter, Forschungszentrum Informatik (FZI)

Siebert started his talk by presenting the problems with real-time Java and gave a brief definition of Java real-time. To provide Java with real-time support, all operations must be carried out in constant time, or at least the upper bounds for the execution times of Java bytecode operations must be known. Essentially, the worst-case execution time for object allocations, dynamic calls, class initialization, type checking, and monitors must be determined.

The talk presented a JVM called Jamaica, which implements a deterministic JVM and a hard real-time garbage collector (GC). First, Siebert discussed the typical mark-and-sweep GC, followed by a presentation on how garbage collection and memory allocation are implemented in Jamaica to guarantee a hard upper bound for an allocation. To avoid memory fragmentation, compacting or moving garbage collection techniques are usually employed. However, Jamaica takes a new turn on this issue in order to avoid fragmentation altogether. The heap is divided into small, fixed-sized blocks (32 bytes). An object, depending on the size, is assembled as a linear list of possibly non-contiguous objects. With this model there is no need to defragment memory and move objects. When a block is allocated, the GC scans a certain number of blocks. This approach can guarantee that the system does not run out of memory, as well as guaranteeing an upper bound for the garbage collection work for the allocation of one block of memory.

The rest of the talk focused on how to obtain deterministic bytecode execution. Most bytecode operations can be implemented directly as a short sequence of machine instructions that executes in constant time. These operations include access to local variables and the Java stack, arithmetic instructions, comparisons, and branches. Siebert briefly discussed this but focused more on the bytecodes where deterministic implementation is not straightforward: for example, class initialization, type checking, and method invocation. The details of this can be found in the paper.

Finally, Jamaica's performance was compared to Sun's JDK implementation using SPECjvm98. The results suggested that performance comparable with Sun's non-deterministic implementations can be reached, by tuning the compiler, for example, and by direct generation of machine code instead of using C as the current intermediate representation.

For more information, contact the authors or visit *http://www.aicas.com*.

SESSION: GARBAGE COLLECTION

Summarized by Hughes Hilton

Mostly Accurate Stack Scanning

Katherine Barabash, Niv Buchbinder, Tamar Domani, Elliot K. Kolodner, Yoav Ossia, Shlomit S. Pinter, Ron Sivan, and Victor Umansky, IBM Haifa Research Laboratory; Janice Shepherd, IBM T.J. Watson Research Laboratory

A garbage collector must scan registers and the stacks in order to find objects which can be collected. Typically, there are three types of garbage collector: con-

servative, type-accurate, or conservative with respect to roots. All three have advantages and disadvantages.

A conservative collector is very simple to implement and has a low performance penalty. However, it must retain some garbage because it is not absolutely positive about what is garbage and what is not. This uncertainty also prohibits object relocation, which means that the stack cannot be compacted, degrading performance over time.

A type-accurate collector is much more complex to implement and is very expensive in terms of performance. However, all object-references are known with certainty and therefore all garbage is collected. Objects can also be moved so that memory may be compacted.

Type-accuracy also adds the factor that threads may be stopped only where type maps exist. Creating maps at every instruction can be very voluminous (although maps may be compressed somewhat). Certain algorithms, such as polling and patching, allow for better performance but are still comparatively expensive.

Lastly, a conservative approach with respect to roots scans the stack conservatively, but uses object type information to scan objects accurately. This is a compromise of the other two types of garbage collectors and works well. It allows object relocation and is used widely in Java Virtual Machines. However, compaction and some other GC algorithms are still difficult with this method of scanning.

The contribution of this paper is to propose another type of stack scanning: mostly accurate with respect to roots. In this method, the stack is only scanned accurately where it is easy to do so (most stack frames) and scanned conservatively otherwise. Therefore most objects can be relocated (allowing compaction), and the performance hit is minimal. Also, threads can be stopped anywhere. Further information about projects of IBM's Haifa research group is available at

http://www.haifa.il.ibm.com/projects/systems/Runtime_Subsystems.html.

Hot-Swapping Between a Mark&Sweep and a Mark&Compact Garbage Collector in a Generational Environment

Tony Printezis, University of Glasgow

Two algorithms for generational garbage collection that are often implemented in JVMs are Mark&Sweep and Mark&Compact. The main difference between the two is that Mark&Compact compacts the remaining objects to consolidate free space after garbage collection. These two algorithms are being considered when they are applied to the old generation of the system; they share the same algorithm for young garbage collections (that is, copying).

The Mark&Sweep algorithm is slightly faster than Mark&Compact, in most cases, because Mark&Sweep provides 200-300% faster collection for old objects, although old objects are usually not garbage collected as often as young objects. However, memory fragmentation can occur in a Mark&Sweep system, which can affect long-term performance.

The Mark&Compact algorithm is 10-20% faster in collecting the younger generation of objects because it provides faster allocation of objects to old space (which occurs during young garbage collection). Young garbage collection can occur up to 1000 times more often than old garbage collection, and it must also be taken into account that Mark&Compact defragments memory.

The performance difference between these two types of generational collection is fairly minimal and depends on the behavior of the application involved. However, what if a garbage collector could hot swap between the two types and get the best of both worlds? That was the question that Tony Printezis asked, and the subject of his paper.

The requirements set forth by Printezis for a hot-swapping garbage collector are fairly rigid. It must swap back and forth in constant time, incur a minimal performance penalty from swapping, be time flexible, and make minimal changes to the Mark&Sweep and Mark&Compact algorithms.

In order to develop the switching algorithm, Printezis had to use a fake byte array class to make a free chunk of memory look like garbage to the Mark&Compact collector, while still looking like a free chunk to the Mark&Sweep collector. He used a simple heuristic for when to swap. Mark&Sweep was used mostly for old garbage collections, but if linear allocation of objects from the young generation to the old generation failed a lot, one pass was made with Mark&Compact to defragment the memory.

In benchmarks, the hot-swapping algorithm fared well. It was the fastest of the three garbage collectors in two of the six benchmarks, and those benchmarks it did not win were very close. Also, the fact that the algorithm prevents memory fragmentation must be taken into account when considering the results. In the future, Printezis wants to develop more complex swapping heuristics, but preliminary results look very promising.

Parallel Garbage Collection for Shared Memory Multiprocessors

Christine H. Flood, David Detlefs, Sun Microsystems Laboratories; Nir Shavit, Tel-Aviv University; Xiolan Zhang, Harvard University

Since Java is being used increasingly with shared-memory multiprocessor systems, it makes sense that those systems should employ garbage collection algorithms that can take advantage of multiple processors to increase performance. This paper describes how Christine Flood and her fellow researchers parallelized two sequential, stop-the-world garbage collection algorithms: a two-space copying algorithm (semispaces) and a

Mark&Sweep algorithm with sliding compaction (Mark&Compact).

Load balancing is a big problem for parallel garbage collection. The key to load balancing is correctly and efficiently partitioning the task of tracing the object graph. This task does not lend itself to static partitioning, which is too expensive. Another solution might be over-partitioning by making more chunks than needed and having each processor get a chunk and come back for more. The problem with this algorithm is that the size of the problem is not necessarily known. The solution is a work-stealing algorithm. In work stealing, threads that have work copy some of it to auxiliary queues, where it is available to be stolen by other threads that do not have work to do.

In parallelizing the semispaces algorithm, Flood and her team used work-stealing queues to represent the set of objects to be scanned, rather than Cheney's copy and scan pointers (used traditionally). To avoid contention when many threads were allocating objects into space at the same time, they had each thread allocate relatively large regions called local allocation buffers (LABs).

Mark&Compact consists of four phases that must be parallelized: marking, forward-pointer installation (sweeping), reference redirection, and compaction. The researchers did the mark phase in parallel using work-stealing queues. They handled the forward-pointer installation by over-partitioning the heap. They implemented the reference redirection phase by treating the scanning of the young generation as a single task and reusing the previous partitioning done in the forward-pointer installation phase for the old generation. Finally, they parallelized the compaction phase by using larger-grained region partitioning.

In benchmarks it was found that with the teams' algorithms, the more processors working, the greater the advantage in garbage collection. With eight processors, there was as much as a 5.5x performance gain. The team concluded that parallel garbage collection must be used to avoid bottlenecks in large, multi-threaded applications. The contents of this paper and other works appear on Sun's site at: *http://www.sun.com/research/jtech/*.

SESSION: SMALL DEVICES

Summarized by Chiasen (Charles) Chung

Automatic Persistent Memory Management for the Spotless Java Virtual Machine on the Palm Connected Organizer

Daniel Schneider, Bernd Mathiske, Matthias Ernst, and Matthew Seidl, Sun Microsystems, Inc.

PalmOS does not support automatic multi-tasking capabilities. To achieve that, programmers have to implement low-level event callbacks using the OS database API to suspend and reload their applications. The talk proposes an alternative approach to allow transparent multi-tasking support for Java programs running on Spotless VM, a predecessor of KVM.

To restrict open memory access, the OS provided a simple database API. The API not only accesses a small subset of RAM for the application program but is also costly. Thus, the database API is bypassed by calling an undocumented system call to disable memory protection. The byte-code interpreter in the persistent Spotless VM still resides in the dynamic memory, but all the Java data (including the byte-codes and thread data) are stored in the static memory.

A program is first started by creating a new Store in the resource database tag of the type "appl." When the program is suspended, the VM automatically saves the current state of the application by closing the persistent Store in a controlled manner. To resume the suspended Spotless VM, it will be retrieved from the Store database. Next the VM will restore each heap record. Since the OS can move Store records in the heap segments, VM needs to update the pointers. After all the pointers have been updated, each module of the VM restores their state from the content in the Store header field before execution of the application continues. When a program finally terminates, the VM will remove the Store data from the database.

A program often needs external states or data that are not under the control of the program runtime system. Spotless VM supports persistence in these states through the implementation of an interface "External." External data have to synchronize with the internal data when the program is suspended or resumed. To achieve this, Spotless uses a protocol adopted from the Tycoon-2 system.

Disabling write protection creates a new dimension of safety issues for PalmOS. It is arguable whether a well-implemented VM will not cross its boundary, but hardware restriction is suggested. More information on Spotless Java Virtual Machine is available at *http://www.research.sun.com/spotless/*.

Energy Behavior of Java Applications from the Memory Perspective

N. Vijaykrishnan, M. Kandemir, S. Kim, S. Tomar, A. Sivasubramaniam, and M. J. Irwin, Pennsylvania State University

With mobile and wireless computing gaining popular ground, battery lifespan has become a growing concern. N. Vijaykrishnan's presentation addressed the energy behavior of the memory system during the execution of Java programs. It has been observed that memory systems consume a large fraction of the overall memory energy. Load/store are the instructions that access the most memory, consuming more than 50% of the total energy in both interpreted and JIT-compiled programs. As data themselves, byte-codes need to be fetched from memory, and so interpreters are

more memory-intensive than JIT-compiled code.

ExactVM (EVM) is the JVM from Sun Labs Virtual Machine for Research used in experiments. The experiment is based on the seven applications from the SPEC JVM98 benchmark suite, with emphasis on "javac" and "db." Beside the actual execution of Java applications, EVM utilizes memory heavily in three areas: class-loading, dynamic method compilation, and garbage collection. Other than the frequency of memory accesses, energy consumption is also dependent on frequency of cache misses since off-chip memory accesses are more expensive than on-chip accesses; thus data locality is an issue. It was found in their experiment that energy consumption is inversely proportional to the cache size

To improve energy consumption, it was recommended that class files should be reused across different applications and that heap allocators and garbage collectors be energy aware. Although energy consumed by dynamic compilation in JIT mode is quite significant, a well-designed compiler will produce native code that actually reduces energy consumption. More information on the talk can be found at

http://www.cse.psu.edu/~mdl/.

On the Software Virtual Machine for the Real Hardware Stack Machine

Takashi Aoki, Takeshi Eto, Fujitsu Laboratories Ltd.

This talk focused on using picoJava-II as a software virtual machine running on a real hardware stack machine. picoJava-II is a Java chip developed at Fujitsu. Unlike traditional JVM, which uses a straightforward memory area as a Java stack, picoJava-II takes advantage of the hardware cache for the stack to improve the bytecode execution performance. Sun's PersonalJava 3.02 is ported onto picoJava-II, which is running on REALOS.

picoJava-II has a different engine architecture from traditional JVMs. Numerous modifications have to be made in order to port PersonalJava onto the direct bytecode execution engine of picoJava-II. picoJava-II has a 64-word stack cache to improve bytecode execution performance. Since there is no coherency between the stack and the data cache, the former has to be flushed frequently before accessing the stack frame. Another issue is that the stack grows in the opposite direction (downward), requiring additional computation to resolve the start of the next frame. JavaCodeCompact (JCC) is a tool available on PersonalJava to improve class-loading performance and reduce code size. The internal data structure of JCC has to be modified before the hardware can accept it.

The testing indicates that the Java microprocessor is significantly better than the conventional C interpreter. It is also competitive with JIT-compiled code. However, there are a number of open problems encountered in the research. First, the lack of coherency between stack and data caches complicates software design. Next, the JNI implementation can be more efficient if the C compiler of picoJava-II follows the calling convention of the Java method. Lastly, the presence of aggregate stacks for solving the stack cache incoherency problem complicates system programming.

Saul Wold & Étienne Gagnon

Fifth Workshop on Distributed Supercomputing Scalable Cluster Software

Cape Cod, Massachusetts

May 22-24, 2001

Summarized by Al Geist, Chair

The invitation-only Scalable Cluster Software workshop was attended by 50 system administrators and managers from the DOE ASCI Labs, DOE Science Labs, and the NSF , representing the largest computer centers in the nation.

The tone of the conference was set by the opening talk by Bill Camp, director of computing at Sandia National Lab, "What Are the Roadblocks to Terascale Computing with Commodity Clusters?" He described Sandia's experiences building and running CPlant, which is a collection of clusters totaling more than 1,600 nodes. He emphasized the need to focus on software reliability, scalability, and usability for terascale clusters.

Following Camp's talk was a session where each of the represented computer centers had 15 minutes to describe the systems they have in place and what new systems they expect to acquire in the next year. This helped the audience understand the scale of systems that must be addressed by cluster software. The largest-scale system, ASCI Red at Sandia, has over 9,000 nodes, and the most powerful system, ASCI White at Livermore, gets over 12Tflops from 8,200 processors.

The afternoon session had administrators from Sandia, Pittsburgh Supercomputer Center, and Oak Ridge National Lab (ORNL) describing the key system software needed for managing and running terascale computer centers. Their talks were followed by discussions of what software was needed to increase scalability and reliability for the systems we heard about in the earlier session.

The first day ended with a vendor session where IBM, Compaq, Scyld (Linux clusters), and Unlimited Scale had a chance to talk about their efforts to produce scalable cluster software.

So, the first day set the scale of the systems that exist, the system software needs, and what vendors are and are not going to solve for us.

The keynote talk on the second day of the workshop was given by Al Geist, leader of computer science research at ORNL. His talk, "Scalable System Software Enabling Technology Center," described a new five-year DOE effort to address the gap forming between the size of the hardware being put in place and the scalability of the systems software presently available. He described the two-year history in putting this center together and the four major goals of the center:

1. To collectively (with industry) agree on and specify standardized interfaces between system components in order to promote inter-operability, portability, and long-term usability. This process would follow the model used in the MPI specification effort, with an open invitation for any groups who want to participate in that effort.
2. To produce a fully integrated suite of systems software and tools (based on the interfaces defined in 1) for the effective management and utilization of terascale computational resources, particularly those at DOE facilities. Initially, the suite would adapt existing system software tools, later producing more scalable versions.
3. To research and develop more advanced versions of the suite components as well as OS modifications required to support the scalability and performance requirements of science applications.
4. To carry out a software life-cycle plan for the long-term support and maintenance of the resulting systems software suite. Again, this would be modeled after MPI, where vendors began supporting their own compliant versions of the specification.

The resulting discussion revolved around the ambitiousness of the effort, much harder than MPI, and how just getting the first goal done would be a big step forward in the future development of systems software. The discussion also brought up the need for a portable set of regression tests to go with the systems software suite.

The conference ended with a "Future Vision Panel," where Mark Seager, Art Hale, and Martin Frey reflected on the discussions at the workshop and projected these forward to 100Tflops systems that are coming in the next five years.

For more details on the conference, including copies of most of the talks, visit the conference Web site: *http://www.csm.ornl.gov/meetings/CapeCod*

The First IEEE/ACM International Symposium on Cluster Computing and the Grid

Brisbane, Australia

May 15–18, 2001

Summarized by Craig A. Lee

The First IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid) focused on the combined areas of clusters and Grid computing, which share many related technical issues and are both areas of intense interest and rapid growth. Cluster computing has enabled low-cost entry into supercomputing performance by using clusters based on commodity components, such as processors and network infrastructure. Grid computing borrows its name from the analogy with the electrical power grid. The electrical power grid made electricity widely available and easy to use. The "information power Grid" endeavors to make the discovery and sharing of information and resources widely avail-

able and easy to use. Clusters and Grids share many communication, scheduling, monitoring, and application development issues, with Grids being the most general case since they can be heterogeneous and open-ended.

Following a traditional structure, the symposium consisted of six keynote addresses and invited talks, three tutorials, seven workshops, 48 technical papers, a poster session, an industry track, and a panel. The six keynotes covered the spectrum of important cluster and Grid computing issues: Ian Foster of Argonne National Lab spoke on Grid architecture, Andrzej Goscinski of Deakin University spoke on cluster organization and management, Satoshi Matsuoka of Tokyo Institute of Technology and Domenico Laforenza of CNUCE both spoke on programming, Greg Pfister of IBM spoke on a new communication technology, and Bruce Maggs of Akamai spoke on content delivery.

The keynotes set the tone for the rest of the symposium. The main symposium technical tracks covered component and agent approaches; Grid computing; scheduling and load balancing; message passing and communication; I/O and databases; performance evaluation; distributed shared memory; and tools for management, monitoring, and debugging. The seven workshops presented more recent "work-in-progress" in areas closely related to the technical tracks: agent-based cluster and Grid computing, object and component technology for cluster computing, quality of service for global computing, scheduling and load-balancing on clusters, global computing on personal devices, distributed shared memory, and cluster computing education.

The symposium concluded with a panel: "The Grid: Moving It to Prime Time" that was moderated by David Abramson. Panelists included Satoshi Matsuoka, Craig Lee, Gul Agha, and Bruce Maggs. Besides discussing the myriad technical issues surrounding the development of effective Grid computing in general, the panel discussed the even more problematic issues of moving Grids from the scientific and engineering communities to be part of the mainstream-computing infrastructure that is enveloping the world.

Grid computing has emerged as the predominant approach for wide-area, high-performance computing, but other approaches, such as peer-to-peer computing and CORBA, are also emerging, and these technologies are motivated more by the business-to-business and business-to-consumer markets. However, these application domains are faced with the same fundamental problems (e.g., resource discovery, scheduling, security), but the solution spaces and potential implementations could be quite different and determined by the commercial marketplace. Hence, the future of cluster and Grid computing will be heavily influenced by how they co-evolve with these other global computing paradigms.

CCGrid 2001 was highly successful by any standards and especially for a new symposium. It attracted world-renowned computer scientists from 28 countries with a high-quality program. It has already been announced that CCGrid 2002 will take place in Berlin, Germany, May 21–24, 2002. Full details are available at *http://www.ccgrid.org*.

Large-Scale Cluster Computing Workshop

Batavia, Illinois

May 22–25, 2001

Summarized by Dane Skow and Alan Silverman, with Joe Kaiser

Invitations to the Large-Scale Cluster Computing Workshop (LCCW), held at Fermi National Accelerator Laboratory, were sent not only to HEP sites but also to sites from other sciences, including biophysics. Participation by representatives from commercial firms with technical backgrounds was also welcomed. Invitations were extended to those institutions with a minimum cluster size of 100–200 nodes.

The workshop was jointly chaired by Dane Skow of Fermi Lab and Alan Silverman of CERN and was held as a response to the formation of a Large Clusters SIG at HEPiX at JLab last year. The mandate of the SIG was to promote appropriate sessions at HEPiX meetings and to hold special meetings outside of the realm of HEPiX to discuss ongoing work and future needs. The LCCW was a response to the special meeting idea, and it was conceived to give sites the opportunity to share relevant work-in-progress, promote collaboration, and share projects.

The primary goal of the workshop was to gather practical experience in building, managing, and designing clusters. Another goal was to take the practical experiences gained and write the definitive guide to running and building a cluster — hardware selection and testing; software installations and tool upgrades; performance testing, logging, and management; accounting issues; and security concerns. This documentation must include what currently exists and what might scale to clusters in the 1,000+ node range. The "Cluster Builder's Guide" was expected to be a work-in-progress by the end of the workshop.

The format of the workshop was half-days devoted to talks and panel discussions and half-day working-discussions in parallel sessions on two tracks:

Topic categories included:

Track A: Facility Operations

- Monitoring
- Fault-Tolerance Design
- Configuration Management

Track B: Usage Cases/Applications

- CPU Allocation
- Data Access/Movement
- Security/Access Control

The plan was to have morning plenary sessions to set the stage and stimulate general themes, while the parallel sessions discussed details toward a full outline of needed work enlightened by experience.

THE FIRST DAY

The opening session was chaired by Dane Skow, OSS department head at Fermi Lab. Dane and Alan Silverman (CERN) discussed the opening goals of the workshop and then turned the time over to Matthias Kasemann (head of the Computing Division at Fermi) for the official welcome. Matthias outlined the work that is being done at Fermi to support a myriad of experiments being conducted there (CDF, DO, BTeV) and throughout the world (the Compact Muon Solenoid [CMS] collaboration for the CMS experiment at the Large Hadron Collider at CERN, NUMI/MINOS, MiniBooNE, and Pierre Auger).

Matthias laid out the challenges to conducting meaningful computing when communication, collaboration, and computing resources are widely distributed, and software development and physics analysis has to be done at such great distances. Matthias posed some critical questions to the participants with regard to the clusters they are currently building and designing. He asked that they consider whether or not clusters should or can emulate a mainframe with regard to resource allocation, accounting, monitoring, and system administration. Is this even possible in a heterogeneous environment? Other questions were: How much can the compute models be adjusted to make the most efficient use of cluster computing? Where and when is it more cost-efficient to not use compute clusters? What is the total cost of ownership for clusters? How can a cluster be built based on the incidental use of desktop resources, e.g., Condor and seti@home? The bottom-line concern is how to get the most cost-efficient use of compute resources while still undertaking global computing for global experiments. He ended by welcoming the participants with the weather report, stating that the weather was perfect for this workshop: it would be raining.

Wolfgang von Rueden of CERN then gave an outline of the Large Hadron Collider (LHC) computing needs. The LHC is being built in Switzerland and is expected to come online in 2005. It is the next generation of supercollider and will be almost 10 times more powerful than Fermi Lab's Tevatron. The computing structure required to handle the data acquisition and data analysis requires worldwide collaboration because no one institution has the fiscal or physical resources to do this alone.

Bill Gropp of Argonne National Laboratory and the IEEE Task Force on Cluster Computing (CSTF) discussed the current activities of his group. The CSTF was created in 1999 as a focal point for discussion of cluster activities. Their goals are to set up standards and be involved with issues related to the design, analysis, and development of cluster systems as well as the applications that use them. Bill pointed out that the task force had a short-term lifetime (two to three years) and considered cluster computing as NOT just parallel, distributed, or the Internet. It is a mix of them all. Bill included several URLs that are really noteworthy:

http://www.ieee.org
http://www.clustercomp.org
http://www.TopClusters.org

After a brief break, a panel conducted by Dane Skow, and including a number of HEP and non-HEP facilities, gave presentations on their current clusters. The assignment was to provide a brief description of each cluster, its size, its architecture, its purpose, its special features, what the decisions/accommodations were made because of the special nature of applications being run, and any optimizations made. Contributions were made by Tom Yanuklis (RHIC — Brookhaven National Lab), Charles Young (BaBar), Steve Wolbers (Fermi Lab), James Cuff (The Sanger Centre), Ralf Gerhards (H1 at DESY), Atsushi Manabe (Kek), and Jim Simone (TH QCD at Fermi Lab). These presentations are online and can be accessed at *http://conferences.fnal.gov/lccws/*.

THE SECOND DAY

This day began with a plenary session on "Clusters at Large Sites." This was chaired by Steve Wolbers and featured Tim Smith presenting "CERN Clusters of Today," "BNL and Other Large Clusters" by Steve Duchene of VA Linux, and "The SLAC Computer Center" by Chuck Boeheim. The most pressing issues for BNL and CERN were floor space, cooling, and power. Clusters need a lot of all of these and they are in tight demand in many lab computer centers. NERSC is not facing this issue as they recently had a large computing facility built for them. Chuck Boeheim noted that "A cluster is a large error amplifier." Management of hardware and software are issues for all of these clusters, and these activities are usually performed with a combination of open source and homegrown tools.

The next panel was on "Hardware Issues, Selection Criteria, Life Cycles, and Cluster Heterogeneity," chaired by Lisa Giachetti, group leader for Fermi's Scientific Support Group. SCS handles the offline CDF and DO farms and the CMS farms at Fermi. The panelists were Tom Yanuklis of BNL speaking about the RHIC farms at Brookhaven and Thomas Davis from Lawrence Livermore talking about the PDSF Operational Model. The panelists were given the following seed questions:

- What criteria are used to select hardware: price, price performance, compatibility with another site, in-house expertise, future evolution of the architecture, network interconnec-

tivity, etc.? Obviously, all of these may play a role – which are the three most important in order of significance?
- Do you perform your own benchmarking of equipment?
- How do you handle life cycles of the hardware (e.g., the evolution of Pentium processors where later configurations and generations may need a new system image)?
- Do you have experience, positive or negative, with heterogeneous clusters?

The criteria for hardware selection ranged in complexity for all the facilities. Some have a qualification process by which vendors supply sample machines with specific configurations that are then tested. The vendors who pass the qualification are then sent bids whenever a purchase needs to be made. Other facilities have nothing so complicated and simply give out test machines from vendors to their users to evaluate. For all hardware purchases in most facilities, there is an acceptance test or period of time in which the systems are given a workout before final acceptance. Most facilities assume the criterion of three years until obsolete. Tom Yanuklis ended his short presentation with questions for the audience: What drives software upgrades and changes? Who proposes and approves the changes, users or administrators? Can you retool your cluster quickly when your users' environment and needs change? How will vendor changes to a product affect your farm? Will Moore's law cease to be accurate in the future? This is a fairly critical question since HEP and clustering depends upon this for future needs.

After lunch, a pair of parallel sessions was held.

Parallel Session A1 was on "Cluster Design, Configuration Management," with Thomas Davis of LBNL as chair. Panelists were John-Paul Navarro of Argonne National Lab speaking about the Chiba City cluster, Shane Cannon of LBNL with the PDSF and Alvarex clusters, and Joshua Harr of Linux NetworX. The seed questions for this session were: Do you use modeling tools to design the cluster? (Most do not.) Do you use a formal database for configuration management? (Some do, and as clusters get bigger it will become more necessary.)

Parallel Session B1 was on "Data Access, Data Movement" and was chaired by Don Petravick of Fermi Lab. Panelists were James Cuff of Sanger Centre, a biophysics company, Doug Thain (Wisconsin) presenting on Condor, Kors Bos of NIKHEF, and Chris Dwan from The Center for Computational Genomics and Bioinformatics at the University of Minnesota. The seed question was, What is the size of the data store, and what tools are in use? A lengthy discussion then ensued ranging from SANS to the performance of tape disk vaults.

There was a blissful half-hour break before the next two parallel sessions.

Parallel session A2 was on "Installation, Upgrading, and Testing of Clusters." This was chaired by Steven Timm, a member of the Fermi SCS group and one of the chief system administrators for the offline farms there. Panelists included Atsushi Manabe (KEK) speaking about the newest incarnation of Dolly++; Philip Papadopoulos (San Diego) speaking about NPACI Rocks, a most interesting cluster install took, and Jarek Polok of CERN discussing his work-in-progress for DataGrid. The seed questions were: Do you buy installation services from the supplier or a third-party vendor? Do you buy pre-configured systems or build your own configuration? Do you upgrade the full cluster at one time or in rolling mode? Do you perform formal acceptance or burn-in tests? All of these questions were bandied about. Most facilities are moving to or have moved solely to network installs. NPACI Rocks is RedHat-based and is a definite must see if you are installing a cluster; for more information, go to *http://rocks.npaci.edu*.

The second parallel session, B2, was on "CPU and Resource Allocation," chaired by Jim Amundson of Fermi Lab. Panelists were David Bigagli and Charles Young (BaBar). Seed questions were: What batch queueing system is in use? Do you have turnaround guarantees? Do you have pre-allocation of resources? This turned into a discussion on the merits of various batch systems, mostly LSF.

THE THIRD DAY

The morning began with a plenary session where the clusters of smaller sites were featured. The presiding chair was Wolfgang von Rueden of CERN, and the panelists were Kors Bos of NIKHEF presenting D0 clusters in the Netherlands, Ian Bird from JLAB speaking about the "Design and Management of the JLAB Farms," and Wojciech Wojcik (CCIN2P3) on "Running the Multi-Platform, Multi-Experiment Cluster at CCIN2P3." The session emphasized the multi-experiment configurations that smaller sites must maintain. They frequently play host to smaller but necessary amounts of compute resources for large experiments and represent a significant resource for smaller experiments worldwide. Their main issues are floor space and adequate network connectivity to the larger experiments' host sites.

A panel on software issues specifically concerning tool selection criteria, tool evaluation, etc. was chaired by Ian Bird of JLAB. Panelists were Derek Wright of Wisconsin discussing how to install, configure, and monitor a Condor pool; Metaprocess Platform presenting the software that launched a million seti@homes; and Ruth Pordes of Fermi Lab speaking on the Grid. Questions the presenters were expected to address were: How do you select software tools — by reputation, from conference reports, after in-house evaluation, or by personal experience? Since all of these may play a role,

which are the three most important in order of significance? Do you trade off personnel costs against the cost of acquiring commercial tools? The biggest issues with software for clusters are scalability and affordability.

Another lunch and then a launch into parallel sessions.

The A3 parallel session was chaired by Olof Barring of CERN, and panelists were Tony Chan who maintains and extends the software monitoring tools at BNL; Tanya Levshina of Fermi Lab, part of the software team that is developing NGOP (next generation operations), and Olof Barring of CERN. Discussions centered on the tools currently used, open source, NGOP, and some other homegrown scripts; the scalability of these tools; and the practicality of building versus buying.

The B3 parallel session concerned user issues and security. Mark Kaletka, former FCIRT team head and Chris Dwan were the panelists, with chairing by Ruth Pordes. Questions presented to the panelists were: Do you have written policies for users regarding non-abuse of the system, the right to check email, and the right to enforce password rules? Do you have a dedicated security team? Do you permit and enforce rules for off-site access? Mark Kaletka presented the current state of affairs in terms of security at Fermi and detailed the Kerberos rollout. It was pointed out that the security of the data is not the worry at Fermi so much as the use of the systems themselves to launch attacks.

Another break, two more parallel sessions. The pace was exhausting, but it was raining, and the food at Fermi isn't very good, so what else was there to do?

Parallel session A4 was on Grid computing. This was chaired by Chuck Boeheim of SLAC with panelists Olof Barring (CERN) (European DataGrid, Fabric Mgmt) and Ruth Pordes (Fermi Lab) (GriPhyN/PPDG). Grid issues and how they relate to current experiments and upcoming experiments were discussed.

Parallel session B4, chaired by Tim Smith of (CERN), was on application environment, load balancing, and job and queue management. Panelists were David Bigagli of Platform; Jeff Tseng (MIT), one of the makers of the Run 2 CDF Level-3 Trigger Online Cluster; and Tim Smith of CERN. Questions initially put to the panelists were: What kind of applications run on the cluster? Does the cluster support both interactive and batch jobs? Is load balancing automatic or manual?

And it came to pass that the sun set, and the participants ate once again. They looked upon what they had done, saw that it was good, and they called it the end of day three.

THE LAST DAY

The plenary session this morning consisted of 10-minute presentations summarizing the eight panels (A1–4, B1–4). Alan Silverman supervised the speakers. The goal was to present to the general body what had been discussed so that everyone could have the benefit of the discussions.

After a brief break Greg Lindahl and Neil Pundit gave a summary of the 5th Workshop on Distributed Supercomputing, with Dane Skow chairing. (see Al Geist's summary of that conference in this issue.)

The final panel was on the future. This also was chaired by Alan Silverman, with panelists Neil Pundit of Sandia speaking about the CPlant initiative and Jan Lindheim of Cal Tech. Questions put to the panelists were: What is the most potentially useful future trend that might affect your cluster? What is the largest single bottleneck to future expansion or development of your cluster?

Conclusions and thanks were made by Alan Silverman.

beowulf cluster computing at the third plateau

by Dr. Thomas Sterling

Thomas Sterling holds a joint appointment at the NASA Jet Propulsion Laboratory's High Performance Computing Group, where he is a principal scientist, and the California Institute of Technology's Center for Advanced Computing Research, where he is a faculty associate.
For the past 20 years, Sterling has carried out research on parallel-processing hardware and software systems for high-performance computing. Since 1994, he has been a leader in the national petaflops initiative. He heads the hybrid technology multithreaded architecture research project.

tron@cacr.caltech.edu

Introduction

It was only seven or eight years ago that the early NOW (Network Of Workstations) and Beowulf projects launched their exploration of the opportunity to harness clusters of low-cost workstations and PCs, respectively, to achieve significant speedups on real-world applications at superior price-performance. Prior to that, and even to this day, throughput (also referred to as "capacity") computing was pursued by means of workstation and PC farms: desktop and server computers on shared local area networks whose first obligation was to serve dedicated users but whose available idle cycles might be applied in concert to some additional workload, albeit largely decoupled.

While such capacity processing leverages existing resources, increases efficiency of system utilization, and yields advantage in cost of computation, workstation clusters and Beowulf-class systems employ parallel processing to increase the size and speed of individual applications. Even the inchoate Beowulf systems of the mid-1990s were competitive in performance with respect to their MPP counterparts (of equal numbers of processors) while exhibiting price-performance advantage of an order of magnitude or more for some (but not all) technical and scientific problems.

By 1996, Beowulf systems were delivering supercomputer performance at high-end scientific workstation prices, earning the 1997 Gordon Bell Prize for price-performance and being dubbed "do-it-yourself supercomputing" by *Science* magazine. Beowulf cluster computing had reached the first plateau. Beowulf was useful, distinct, and attracted many practitioners. As a subdiscipline of parallel computing, it was self-sustaining.

In the intervening years, Beowulfs have experienced an explosive growth in the scale of capability and capacity as well as their installed base and range of application domains. On a per processor basis, clock rate has increased by a factor of more than an order of magnitude and peak floating-point performance by more than two orders of magnitude. Network bandwidth has increased as well by two orders of magnitude while latency has dropped by more than a factor of 10. Storage capacity of both main memory and hard disks has been expanded by more than an order of magnitude on a per node basis. Overall, price-performance has improved by better than 200 and total system performance for the largest Beowulf-class systems, taking into consideration system scale (number of processors) as well as per node performance, has exploded by 10,000 – truly a revolutionary capability.

Today, many of the Top 500 computing systems are commodity clusters that include Beowulf-class systems. In the near future, such clusters are likely to be at the top of the list with important new systems under development by NSF and DOE. But the dramatic evolution and impact of Beowulf-class clusters have been enabled as much by software as hardware, and their future is as dependent on next generation software technology as their hardware technologies. Today, commodity clusters are at the second plateau, defined as much by their supporting software as their assembly of hardware. Slowly and incrementally, elements making up the software environment have matured and been enhanced in scope to enrich the tools with which clusters are managed and applied.

Beowulf clusters are at a pivotal stage in their evolution

But now Beowulf clusters are at a pivotal stage in their evolution. They are poised to dominate the realm of high-performance computing, but only if they can provide the level of services and robustness demanded of early generations of vector supercomputers, MPPs, DSMs, and SMPs. The promise of commodity clusters is their potential ability to ratchet up the performance by creating ensembles of these and other classes of computing elements including simple PCs. But they will fail if they prove too difficult to use. The question is: What are the key attributes that must be achieved to bring Beowulf clusters to the level needed to dominate high-end parallel systems, to reach the third plateau?

The First Plateau

Though there was more than one choice, it was Linux combined with MPI that made Beowulf-class systems both possible and practical. Initially, PVM was the message-passing library employed for the first Beowulf systems. With the emergence of the community-wide standard across platforms and the potential for true portability, the use of Beowulfs took off, but these were primitive environments at best. Linux provided the virtual memory multi-tasking node environment needed to build a distributed capability. And MPI provided the logical inter-process data transfer and coordination mechanisms necessary for cooperative computing.

However, these simple systems were usually modest in scale, rarely more than 64 processors and often only 16 processors or less. They usually ran only one parallel application at a time, often administered by a single individual or small group where scheduling was coordinated by word of mouth. Frequently, a simple set of tools for monitoring the low-level status of the cluster nodes was developed in-house. In some cases, not even MPI was used, the programmers preferring to optimize their application communications using the sockets layer protocol. Communication performance could be improved by a factor of two to three when custom crafted compared to early implementations of MPI.

Although basic, these simple systems were responsible for a strong grassroots community effort to establish PC clusters as a viable low-cost alternative to expensive more tightly coupled vendor-specified parallel computers. More than cost, these early ensembles had other attributes that a portion of the user community found desirable. One was flexibility of configuration and easy upgrades. Many aspects of the system structure could be determined by the end user without permission from some vendor. Another was the low vulnerability to vendor market and product decisions. If one vendor stopped producing the elemental components of a Beowulf, it was easy to acquire comparable units from any one of several other distributors. This sense of confidence and empowerment led to the view that Beowulf-class systems represented a convergent architecture – one for which application software could be guaranteed many generations of compatible hardware.

The Second Plateau

By 1997 it was clear that commodity clusters were having an impact on high-end computing and would become a major force in parallel processing. Vendors began to support commodity clusters and, ultimately, even Linux-based Beowulf systems. Both hardware and software products were developed to directly support a burgeoning commodity clusters market.

At the first plateau, Beowulf systems leveraged existing pieces of hardware and software developed for other purposes with only small additions such as network drivers contributed by the community where essential. But by 1999, if not before, Beowulf-class

cluster computing had reached the second plateau, where hardware and software were being developed and, in some cases, marketed explicitly for clusters. At the second plateau, node packaging and system area networks were implemented to facilitate commodity clusters. Rack-mounted 1U packages are now available that permit 40 or more nodes to be assembled in a single rack where less than half that many could be contained in the same footpad using desk-side towers.

Rack-mounted 1U packages are now available that permit 40 or more nodes to be assembled in a single rack

SCI and Myrinet were devised initially for workstation clusters, but as their costs were reduced per node and the scale of topologies they could implement was increased including the degree per switch, the target system was increasingly large PC clusters. The virtual interface architecture (VIA) was devised by a consortium of industrial partners for the express purpose of reducing the latency of communication between nodes within a cluster, with example implementations including Servernet II and cLAN. Second-plateau cluster scale grew from moderate-sized systems to those comprising more than a thousand processors.

However, the most significant advance marking the transition to the second plateau is the improvement in software environments and tools. Linux, once a hobbyist's plaything, emerged as the foremost UNIX-like operating system, providing serious competition to NT in certain markets. Linux now has myriad distributions, some of which are sophisticated environments including some support for Beowulf clusters.

The Extreme Linux consortium reworked the Linux kernel internals to eliminate bottlenecks to scalability and to reduce inefficient mechanisms. Equally important was the development of distributed resource-management software compatible with Linux. Several schedulers were developed that have wide distribution, including PBS and the Maui schedulers. PVFS from Clemson University provides one example of a parallel file system developed explicitly for commodity clusters. A number of tools exist for monitoring the status, operation, and behavior of the system nodes. Etnus provides a parallel debugger. Oscar, a consortium led by Oak Ridge National Laboratory, is one of several efforts to provide an inter-operable collection of the basic cluster software derived from a number of existing tools. Today, commodity clusters of the second plateau are challenging all other forms of high-performance computing for preeminence.

The Third Plateau

Commodity clusters may remain, as they are, ensembles of conveniently packaged nodes, interconnected by means of quasi-independent networks and running a collection of separate but mutually friendly software packages that provide various services to allow users to get by. In such a scenario, clusters will continue to contribute to the technical computing arena and aspects of the transaction-processing commercial domain. However, for commodity clusters to rise above their limited condition and forge a new path, then significant advances in both hardware and software will be required. A new generation of Beowulf clusters will be required, ones that achieve the third plateau.

Hardware for Beowulf clusters at the third plateau is being pursued aggressively by industry independently and collectively. The transition to 64-bit architecture for high-end PCs, now currently dominated by Compaq's Alpha family, will be accelerated with the market emergence of the long-awaited Intel IA-64 processor. Meanwhile, IBM describes future plans of incorporating multiple processors on a single chip, thus ensuring continued increase in performance per chip through at least the end of the decade made possible by such new techniques as EUV lithography.

We can expect processor chips delivering as much as 10Gflops peak performance in the next few years.

We can expect processor chips delivering as much as 10Gflops peak performance in the next few years. The industry consortium developing the Infiniband network architecture will push bandwidths up beyond 10Gbps per channel and reduce network latency to near microsecond levels. With continued rapid increase in DRAM capacity and innovative structures for achieving greater effective memory bandwidth, hardware systems will be capable of petaflops-scale performance by the end of the decade. Market pressures for laptops, PDAs, and cellular phones will force power consumption down while significantly improving the size, weight, and power of micro-disks, further enhancing the practicality of large-scale Beowulf-class systems and their availability to a broad range of institutions and users. With these significant advances, commodity cluster hardware is well positioned to reach the third plateau in a few years.

The obstacle to this next quantum step is in enabling software. There are three main challenges to next generation commodity clusters that may limit their long-term impact: resource management, fault recovery, and programming methodology. Resource management involves all aspects of system initialization, maintenance, administration, and job allocation. Today, while some strides have been made in each of these areas for second plateau systems, they represent only partial and incomplete solutions. For example, system administration tools are not on a par with conventional uniprocessor systems. Also, launching new processes is often not nearly as efficient on remote cluster nodes as on the same local processor. Managing copies of software across a large number of nodes can be surprisingly difficult, and it is easy to have nodes within a cluster be inconsistent.

A major thrust in the commodity-cluster community is achieving what is called "single-system image," or SSI. In any system, whether uniprocessor, multiprocessor, or cluster, there are multiple namespaces. These include the variable-address space, file-name space, process-id space, and others. On a uniprocessor, each of these namespaces is single: that is, there is only one such space for each class of name. But on a cluster, in the worst case, there can be as many separate namespaces for each class of name as there are nodes in the system. Commodity clusters at the third plateau will present a single-system image to the user and administrator for most if not all name classes. The exception may be the user-variable namespace. Even here, hardware solutions such as SCI and software solutions such as HPF provide ways to let the user think about a single variable namespace. The Scyld tool set manages all remote-process calls through a single master node providing a single process-id namespace. PVFS provides a single parallel-file namespace. It is possible that a synthesis of these methods may ultimately provide the SSI operation that is key to effective control of large clusters of the third plateau.

Efficiency of resource management is also critical to the successful deployment of clusters. Again, Scyld's process migration mechanisms can achieve speedups of as much as an order of magnitude in starting a process compared to rsh. Another efficiency improvement may come from Linux BIOS being developed by Los Alamos National Laboratory which optimizes the onboard BIOS of each node for use with Linux. This can permit a new node to be brought up as part of a cluster system in less than a minute.

The challenge of reliability requires new fault-response techniques that will prevent an entire cluster system from crashing each time a failure occurs with a single node. For the largest scale clusters with as many as 10,000 processors, MTBF measured in hours or less is possible (numbers vary significantly depending on whether you include infant mortality as part of the life cycle). When a hard fault occurs, future management tools must allow the rest of the system to continue to operate. Ideally, the application running on

the failed node could be restarted at an earlier stage in its computation through automatic checkpointing, thus avoiding having to restart the application from the beginning. While progress in related areas has been made, no fully satisfactory solution exists, and one will be required.

Thus commodity clusters of the third plateau will present the user with a single-system image, manage its resources to varying demands of users and administrators, and provide high availability even in the presence of faults. It is unclear how third-plateau systems of the future will be programmed. New models are slow to achieve acceptance, and the heritage of legacy codes causes older languages to remain for many years. But in the long term, users are likely to be separated from the arduous task of directly managing the computing resources from within the application code and are more likely to rely on advanced operating system and runtime system software to allocate tasks to physical components.

Conclusion

Beowulf-class systems and other forms of commodity clusters have evolved over less than a decade from primitive small piles of PCs to among the largest systems in the world. However, their future dominance and impact on both technical and commercial computing will depend on future advances that will allow them to attain the third plateau. This is a stage of robust, manageable, and effective computing systems that will permit their use in almost any domain of workload currently serviced by other system forms. The third plateau will represent the final stage in this evolution and will offer a stable, cost-effective convergent form of parallel architecture that will promise users dependable and scalable computing platforms for generations to come.

Beowulf-class systems and other forms of commodity clusters have evolved over less than a decade from primitive small piles of PCs to among the largest systems in the world

the RHIC computing facility linux farms

by Tom Throwe

Throwe is the Deputy Head of RHIC Computing at Brookhaven National Laboratory (BNL) on Long Island New York. He received a Ph.D. in Nuclear Physics from Indiana University in 1984 and has been at BNL since 1987.

throwe@bnl.gov

The Relativistic Heavy Ion Collider (RHIC) at Brookhaven National Laboratory began colliding gold ions in June of 2000. The four experimental groups situated around the collider ring came online with the accelerator and collectively recorded 22 terabytes of data during that first run of the machine. RHIC began the second run on the machine in June of 2001.

The data from the RHIC experiments is sent via a gigabit Ethernet link to the RHIC Computing Facility (RCF), where it is stored, reconstructed, and analyzed. Currently, the computing facility consists of four StorageTek tape silos with a capacity of 1.2 petabytes of data, twelve Sun E450 servers serving 50 terabytes of RAID storage, and 422 dual and quad CPU farm machines with a total computing power of approximately 18,000 SPECint95. The facility is expected to grow to between 6 and 10 StorageTek silos, 100 terabytes of disk, and a steady state of approximately 600 dual CPU farm machines.

The Linux farms at RCF started out in 1996 as 10 4U quad CPU rack-mounted systems. Trying to exploit the "commodity" market, the farm was augmented with 24 "desktop" machines arranged on shelving in four racks in 1997. Another 24 "desktop" machines were purchased in 1998, but these had to be laid down on their sides to fit eight in a rack. As we approached our first major purchase, it was clear that we could not use "desktop" machines on shelves, so we only looked at rack-mounted systems and purchased 148 2U rack-mounted machines in 1999. The last purchase was made in late 2000 and consisted of 160 2U dual CPU machines. We are presently preparing for our next purchase, which was originally expected to be of comparable size to our 2000 purchase, but in a 1U form factor. The 1U size is attractive since we will soon be short of space, but our user base is looking toward a locally attached disk, which is limited in a 1U form factor. Purchases beyond our next purchase will involve the retiring of old machines to make space for the new higher performance machines. We are planning on a three- to four-year life cycle of the farm machines.

Figure 1: A picture of part of the RCF Linux computing farm at BNL. Two of the networking cabinets can be seen in the background.

The Linux machines at RCF form a "cluster" only in that they are configured and managed together. The machines do not share resources (other than the network), and no parallel jobs run on the machines. The code running on the machines may be the same, but the running jobs do not communicate with other jobs on the same or different nodes. Whether or not this style of computing continues into the future will depend on the needs and manpower of the RHIC experiments.

The machines are divided into two farms, namely a Central Reconstruction Server (CRS) farm and a Central Analysis Server (CAS) farm. Each of these farms is further divided by experimental group. The division by experiment is at the network and machine level such that one experiment cannot have an impact another experiment's resources. Each of the experiments has a large packet engine switch with a gigabit input link and 100 megabit switch ports to the individual machines. Each farm machine is on a separate switch port. Thus an experiment's server machines are connected to the switch via gigabit, and the server traffic is fanned out to the farm nodes on the 100 megabit network. Our division of resources by experiment leads to inefficiencies in usage of the network and machines, but it eliminates any impact of one experiment's possible poor usage of their resources on the other experiments' resources. So, in the end, the Linux machines are logically divided into eight farms where physically there is only one.

Data first flows into the CRS. The purpose of the reconstruction farm is to take the raw data collected by an experiment and convert it to "physics" quantities that the individual

physicists can then subject to analysis. General users have no direct access to the reconstruction farm machines. One or two members of each experimental collaboration have access to a job-submission account and software. The job-submission software for the reconstruction machines was written in-house mainly due to the need for a custom interface into the HPSS mass-storage system, which manages the data in the StorageTek robots. A job-control file is produced by the authorized user (since a very large number of jobs will be processed, the job-control file is not produced by hand, but by an automated method coupled with the sinking of the raw data) and submitted to the system. The input files necessary for the job are staged from tape and put into the HPSS disk cache. When all input files for a particular job are available, the system finds an available farm node and sends instructions to that node to initiate the transfer of the input files from the HPSS cache to the local disk of the farm node. When the transfer is complete the system sends a message to the farm machine's daemon, which determines if the job is allowed to start. Since the machines have dual CPUs, in the steady-state situation, each node will have two running jobs and one job waiting to run with the input files already resident on the disk. When a job is finished, depending on its exit status, the system will either transfer the output files back into HPSS and clean up the local disk, or on failure, it will just clean up the local disk. The user is then informed of the final status of the job. Database entries are made as the job passes through the system so that statistics on time spent in each stage of the process can be gathered as well as statistics on the final status of each job.

During the running of reconstruction jobs, the user has access to machine and job status through a Graphical User Interface (GUI) written in Perl/Tk. The user can query the status of an individual machine or groups of machines, and he or she can disable and enable machines to stop or allow jobs to be queued to those machines. Similarly, the user can query the status of jobs running on the system and can kill, suspend, or resume those jobs. Nonprivileged users can query information about machines and jobs in the reconstruction farms, but they cannot affect the running of the system.

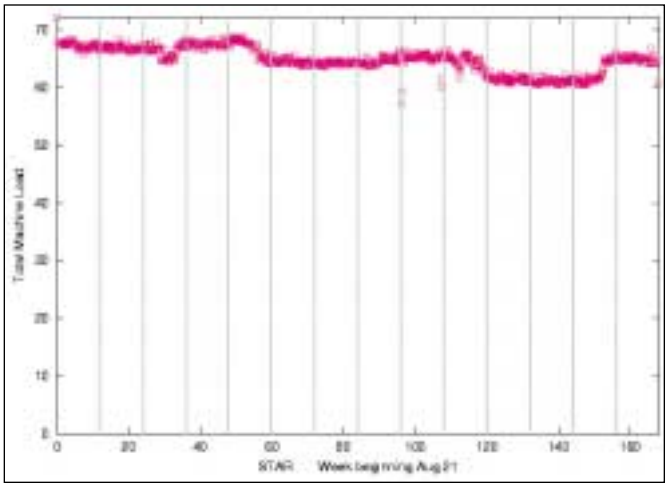


Figure 2: A plot showing the type of steady-state load achieved on the Central Reconstruction Server (CRS) farm at RCF. The STAR experiment's portion of the farm at the time of the plot was 33 machines, so a fully loaded system would have a load of 66 plus the load due to pre-staging a job on each node.

The experiment's code that is run on the reconstruction machines is written by a number of collaborators within the experiment. After being certified by the experiment, the code is integrated into the experiment's reconstruction package and a schedule for reconstructing the data is agreed upon within the collaboration. Due to the amount of time needed to do a reconstruction pass of the current data set, the experiment is unlikely to be able to make more than one such pass on the bulk of the data, so each experiment has a quality-control mechanism in place to try to ensure that the code is both robust and producing accurate results.

Since the RCF reconstruction farm has such restricted access, the load on the machines is fairly steady, as can be seen in Figure 2, and the average uptime of the machines in the CRS farm is rather high and measured in "hundreds" of days rather than "tens" of days. In general, the reconstruction farm machines are only brought down for deliberate maintenance.

The Central Analysis Server farms at the RCF are used somewhat differently than the reconstruction farms. The analysis machines allow batch-only access or batch and interactive access. Since the hardware is segregated by experiment, each experiment group is free to decide on a usage policy. Most groups have decided to allow both batch and

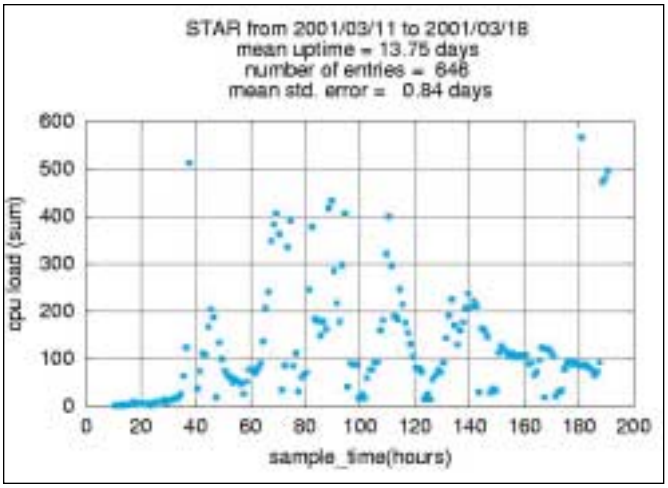


Figure 3: A plot showing the load on the STAR experiment's portion of the Central Analysis Server (CAS) farm at RCF. The load is produced by interactive and batch jobs and shows much more variation than the load on the CRS farms.

interactive access to their CAS machines, but some have designated a subset of their CAS machines to be batch only. This arrangement requires somewhat more work on the part of the RCF staff, but the configuration is fairly static and usually only has to be done once.

The batch system used in the RCF on the CAS farms is the Load Sharing Facility (LSF) from Platform Computing (*http://www.platform.com/platform/platform.nfs/webpage/lsf/*). The system has been found to be quite flexible and allows for the experiments' batch queues to be kept separate and individually configured. The LSF system is quite powerful and allows for the management of a number of system resources, but we essentially only use its batch queueing system.

Unlike the CRS farms, where access is restricted and the jobs are strongly controlled, the CAS farm machines can often get overloaded with users. Figure 3 shows a fairly typical fluctuation of the load on an experiment's part of the CAS farm with time. These large fluctuation loads make for a much less robust farm. The average uptime of these machines then varies by usage pattern among the experiments and by allowed access methods. The average uptimes range from days or weeks for some interactive nodes to months for some of the batch only nodes. In general, the average uptime of the CAS nodes is much less than the average uptime of the CRS nodes, and the downtime is more likely to be caused by user activity on the CAS machine than on the CRS machines.

Monitoring of the RCF Linux farms exists at a number of levels. General connectivity monitoring, farm-specific monitoring, service monitoring, and hardware monitoring are all done. The Mon program (*http://www.kernel.org/software/mon/*), because of its ease of use and extensibility, is presently used for connectivity monitoring, with any detected machine failures going to the Linux farm group's pagers and monitoring Web pages. A heartbeat from the server daemons on the CRS machines is coupled to the CRS control software to keep jobs from trying to be queued to machines with problems. Load, NFS, AFS, and LSF monitors on the CAS machines also send alarms to the Linux farm group's pagers and another monitoring Web page. The CRS and CAS-specific monitoring software was written in-house, mainly in Perl and Python. Hardware status, including temperature, voltages, fan status, and disk errors, is monitored with the VACM software from VA Linux (*http://www.valinux.com/software/vacm/*). The VACM software also allows for the remote reboot and power cycle of any of the farm machines and has greatly enhanced the manageability of the machines.

Up to now, the operating system on the farm machines has been installed using a BOOTP network boot. In this process, the OS is first built on a development machine and transferred to an installation directory tree. A compressed tar image is made of the directory tree, and this image is copied to the BOOTP servers. In the case of a new machine, the network boot procedure is initiated from a floppy disk, while a rebuild of a system can be initiated from a swap of the kernel by replacing `lilo.conf` followed by a reboot. The initial BOOTP connection provided the network parameters for the booting machine and the NFS file system from which to mount the machine's initial root file system in memory. The local disk of the machine is then partitioned, and the compressed tar file of the OS image is untarred onto the local disk. The local network files are customized with the network parameters obtained during the boot process, LILO is run on the local disk, and the machine is rebooted. This first local boot of the machine

runs a customization script that configures the machine as either a CRS or CAS node and then does the final reboot of the machine. When the machine comes up this third time, it either automatically registers itself as a CRS node, or comes up as a CAS node with LSF running. The entire process from initial BOOTP start to final configuration takes about 10 minutes plus the time to build the AFS cache. With the distributed BOOTP servers in place, we have brought up an entire rack of machines at the same time without a problem.

Major upgrades to the system are currently done by producing a new system image and redoing the installation. Minor upgrades are done by distributing the file or rpm via scp, NFS, or AFS and initiating the upgrade by distributing the appropriate command to the nodes via SSH. However, both the initial installation method and the upgrade methods currently used are proving to be inadequate. The installation method has evolved over time and is much more flexible now than when we started, but the hardware is becoming more diverse and the users are demanding more customization, so we are looking at other methods. The "KickStart" system is actively being tested and the "SystemImager" of VA Linux is also being investigated. The goal is to move to a more customizable installation and upgrade system than we currently have.

As noted previously, we are not at the full capacity of the facility. It is expected that we will reach a steady state of approximately 600 dual CPU machines with a replacement cycle of three to four years. All farm CPU purchases to date were of machines with minimal local storage (enough for two running jobs plus a staged waiting job on the CRS machines and a comparable or somewhat larger amount on the CAS machines). Unfortunately, the experiments' need for disk space is outstripping the budget we have for centralized RAID. The experiment groups therefore want to embark on a study of using disks local to the farm nodes to augment the centralized disk. Such investigations of distributed disks are also going on at other institutions. The argument is that by diverting a relatively small amount of the centralized disk budget, a larger amount of non-centralized IDE disks can be attached to the farm CPU than the amount of centralized disks the diverted funds would have purchased. The problem is then shifted from managing and serving data from a large centralized disk to one of managing and tracking the data on a large number of distributed disks. Everyone agrees that keeping track of where the data is in a distributed system and managing the loss and replacement of data files due to disk failures is a difficult problem, but enough people see this as the only way even to approach the amount of disk space they feel they need.

The Linux farms at the RCF have performed well to date. We have had a total of one disk failure, one CPU failure, three motherboard failures, and eight power supply failures in the entire farm. The rack-mounted machines have proven to be quite reliable and maintainable. There is some apprehension – as we add IDE disk drives to some of the existing machines and anticipate future purchases of machines fully loaded with IDE disks — that the increased number of disks and the accompanying increased heat load will increase the failure rates of the machines. Everyone agrees that the machines should work at their maximum configuration, but since we have not run any machines in that configuration, we will reserve judgment until we do have such experience. Our goal is to provide a robust and manageable facility that meets the needs of our user community, and we will work with our users as we evolve the facility to meet that goal.

Keeping track of where the data is in a distributed system and managing the loss and replacement of data files due to disk failures is a difficult problem

picking cluster parts: cluster construction at the genome sequence centre

by Martin Krzywinski

Martin Krzywinski's background is in biochemistry and physics. He has been involved in medical imaging (M.Sc. PET imaging, UBC), stochastic simulation and web programming and design. He is currently the bioinformatics coordinator at the Genome Sequence Centre in Vancouver.

martink@bcgsc.bc.ca

When we decided to purchase our cluster, it seemed that everyone had a different idea of what one should look like. Although the hardware budget largely influences the specifications of a cluster, even those built with similar budgets, there were many solutions. Whether or not to go with a turn-key solution from a vendor was a difficult choice – again largely motivated by the total cost. We had enough expertise to build the nodes from parts and assemble the cluster, but the idea of a pre-assembled solution held its own attraction. If we could have a system shipped to us, already in a rack, much of the tedious assembly time would be saved. That is, if the pre-assembly did not cost too much. On the other hand, better cluster price-performance was probably found in commodity off-the-shelf (COTS) units.

Our cluster had an additional important restriction: physical space. The lab does not have a spacious computer room, and the network room which would house the cluster is approximately 100 sq. ft. – basically large enough to fit three computer racks and a small chair and table. We knew right away that cooling would be of significant concern in such a small space. We had to purchase slim components, since every 1U in the rack counted.

This article describes steps and decisions we made in choosing the hardware for the cluster.

It should be noted that the mention of vendor-specific information is done purely for added information and reference. At the outset, we had no particular vendors in mind and arrived at our final solution based largely on the functionality and value of components rather than the available service options.

Overview and Computing Facilities

The Genome Sequence Centre is a high-throughput facility for carrying out research in bioinformatics and genomics, as well as mapping mammalian genomes (mouse, rat, cow). The Centre is also involved in sequencing projects such as the full-length human cDNA project. The Centre was founded in early 1999 by the late Nobel laureate Dr. Michael Smith. We moved into our new facility at the British Columbia Cancer Agency in December of 1999.

The Centre's computing platform was built around a dual Xeon VA Linux server with 200GB of RAID5 storage provided by a redundant, external Raidion controller. The philosophy behind designing the computing environment was based on the concept of a powerful workstation which did not have to rely on a central server for CPU power, but only for files through NFS. So, while CPU resources are decentralized, much like in a logically managed set of computers, the disk storage and other resources, such as mail and Web service are centralized. All workstations are dual Pentium III platforms (500MHz or higher) with 512MB of RAM and a local system disk. Currently, there are

LINKS:
Genome Sequence Centre:
http://www.bcgsc.bc.ca
Raidion storage:
http://www.raidionsystems.com/
VA Linux:
http://www.valinux.com
Linux:
http://www.linux.com

several file servers with total online storage of nearly 3TB. The network is 100Mbps, fully switched, with a gigabit backbone.

With our wide use of Linux in the lab, in-house expertise and widely available binaries, it was natural to pick Linux as the operating system for the cluster.

Motivation for a Cluster

The field of bioinformatics is famous for its large data sets and diverse, numerous databases. There are hundreds of public databases of biological information. Attempts have been made to centralize some of this information. To be sure, not only do such data sets require large and relatively fast storage solutions, analysis and indexing of this data is a time-consuming task – ideal for clusters.

One example of cluster application is in the building of a physical map of a genome. The process requires *N* x *N* comparisons of objects, and the comparison function is very expensive. With *N* being on the order of 3 million, the process is exhausting on a single computer. In early 2000, the physical map of the human genome took about two weeks to assemble on a 900MHz K7 system (Kryotech's super-cooled 900MHz system). Luckily the process is amenable to being parallelized, since the comparisons are independent. By using existing workstations and parallelizing the software used to build the map (FPC), we were able to reduce the time of a build to less than 24 hours.

The drastic speed increase meant that we could not only assemble the map very quickly, but we could also assemble many maps with different build parameters within a reasonable amount of time. Distributing the code over the workstations in the lab allowed us to gain the expertise to develop similar approaches to other computing problems in bioinformatics. As CPUs very quickly became cheaper and faster, the 500–600MHz workstations no longer represented the fastest white-box computing platforms. In addition, ensuring the constant availability of workstations was a challenge – the owners of the boxes often stressed the machine's capacity, at times to the point of requiring a reboot. Interactive user sessions were being affected by distributed tasks, slowing personnel efficiency.

It became clear that the Centre would greatly benefit from a central source of CPU power.

LINKS:

Kryotech:
http://www.kryotech.com/index.html
FPC:
http://www.sanger.ac.uk/Software/fpc/faq.shtml
Human Genome Project:
http://www.ornl.gov/TechResources/Human_Genome/home.html
Human map build:
http://carbon.wi.mit.edu:8000/cgi-bin/contig/phys_map
Mouse map build:
http://www.bcgsc.bc.ca/projects/mouse_mapping
Genomic databases:
http://ihg.gsf.de/ihg/databases.html
SRS:
http://srs.ebi.ac.uk
SGI bioinformatics cluster resource:
http://www.sgi.com/solutions/sciences/chembio/linux_clusters.html
YAC bioinformatics cluster:
http://bioinfo.mshri.on.ca/yac/
The Collective cluster:
http://www.cs.uidaho.edu/~beowulf/

Cluster Construction

Initial Considerations

As soon as we knew that we needed a cluster, typical issues presented themselves: to build or to buy? Gigabit connectivity to the nodes or not? Do we buy the fastest CPU available or hunt for the best price-performance, the so-called price sweet-spot.

The cluster had to be made flexible to address all types of computing problems: those included CPU, memory, and I/O bound. We could not list all the problems that the cluster was going to help us solve. Ideally, answers we obtained with the cluster would give rise to new and interesting problems.

The Centre was awarded a grant from CFI to construct the cluster. Our budget was large enough to entertain sourcing the hardware and construction to vendors such as VA or IBM, both supporting Linux. After communicating with these companies, it became clear that we could build our own hardware significantly cheaper. Some of the management features of the VA and IBM solutions appeared very attractive, although it was not clear exactly whether this additional monitoring and management hardware provided good value.

LINKS:
IBM high-availability clusters:
http://www1.ibm.com/servers/eserver/iseries/ha/
VA clusters:
http://www-unix.mcs.anl.gov/chiba/
CFI:
http://www.innovation.ca/index.cfm

Broadly, the questions that faced us fell into these categories:

1. Profile of nodes (1U, 2U, or ATX/micro)
2. Node components (CPU, memory, storage, power supply)
3. Cluster networking and topology
4. Electrical and UPS
5. Node management
6. Cooling
7. Vendor-bought or assembled

Profile of Nodes

Being heavily restricted by space, the 1U node solution was the obvious one. At the time, good quality dual CPU 1U cases were becoming available. Internal cooling was a very important factor – the 1U server market was just now opening and it seemed that each vendor had a unique solution. At the time, in the first quarter of 2000, even VA Linux proposed that clusters be built with 2U nodes out of cooling considerations. Later, they designed a small case and handled the cooling effectively and are now proponents of building clusters using 1U nodes exclusively.

LINKS:
IBM 4000R:
http://www.pc.ibm.com/us/netfinity/4000r.html
IBM x330:
http://www.pc.ibm.com/us/eserver/xseries/x330.html
VA 1221:
http://www.valinux.com/systems/productinfo.html?product=1221
Rack mount equipment RC0101 1U case:
http://www.rackmountequipment.com/products/rackchassis/RC0101/RC0101.htm
Rack mount RM0120 1U case:
http://www.rackmount.com/Rackmt/RM-01-20TEST7.htm

If the use of the cluster helped us to solve complicated computational challenges, additional funding for more nodes could be expected. The 1U solution was attractive because it allowed us to expand the cluster with minimal use of additional space. With our space restrictions, it was important to minimize empty space within a node as much as possible, without sacrificing cooling or the quality of the power supply of the case.

Node Components

STORAGE

To keep the cluster flexible, we wanted each node to have its own local storage space. Installs, and possibly booting, would be done over the network, but each node would house its own copy of the required system files and libraries. Anticipating at least 512MB in each node, the memory overhead in caching these files would be insignificant. Local storage would also remove the stress on the cluster NFS server placed by nodes having to retrieve any information from a centralized disk. Local storage would also allow nodes to write large temporary files without causing an NFS bottleneck. Finally, if some jobs required very large input files, these files could be copied to each of the nodes ahead of time. SCSI was felt to be too expensive, especially for large, fast disks, and therefore, we decided at the time to stay with the EIDE interface. We could easily give each node 20 or 30GB of local storage at a fraction of the price of SCSI disks. With the cost savings, we could always purchase additional disks for the cluster RAID stack.

MEMORY

Many of the bioinformatics tasks require large amounts of memory, upwards of 1GB or more. We decided that each node would therefore have 1GB of fast memory, PC133 ECC Registered. If the motherboard was chosen to hold up to 4GB, we could always add memory later. Deciding to skip ahead from 512MB to 1GB for the nodes meant that the cost would be higher, but we felt that we could address a broader range of problems this way.

POWER SUPPLY

The power supply was a significant concern. In the management of about 50 computers in the Centre, power supply faults are probably the most common hardware failure.

Low-quality power supplies lose capacitance over time and put undue stress on node components, decreasing their life span. As much as internal node cooling, choosing a case with a powerful power supply was important. If we were to go to a vendor to purchase the cluster, the power supply choice would be made for us.

CPU

The last and important key to node performance is the CPU. We quickly discovered that measuring the value of CPU speed in cluster nodes did not necessarily follow the same rules as doing so for a single workstation. For example, at the time of writing this article, local CPU prices were about C$200 for 700MHz Pentium III and C$360 for 1GHz Pentium III. The latter price per MHz is 1.3 times higher. When we were building the specification list for the cluster, however, the difference between the fastest processor and the best-valued processor was much larger – typically five times or more.

When building a single workstation, one could argue that paying a premium for the fastest processor does not equate to good price-performance. This simple consideration motivates most workstation purchasers, who typically like to buy the second or third-fastest CPU. However, once one factors in the price of the memory and other components in the system, the CPU represents only a small fraction of the total cost of the computer. Consequently, the relative price difference between, for example, a 700MHz or 1GHz processor is small. At this point one might think that when already spending many times the cost of a single processor on a workstation, the cost of an additional half-processor can be justified.

This can be illustrated by the following example. Suppose that the cost of the node without the CPUs is $4000. Let the price of 700MHz CPUs be $200. Using the 700MHz CPUs, with a budget of $100,000 for nodes alone, one can purchase 23 nodes. The speed of such a cluster would be 32Gflops. Going to the 1GHz CPUs at $360 each, we can afford 21 nodes but the cluster speed is 42Gflops. We have purchased two fewer nodes and the cost of the node components was transferred into the additional CPU cost. Interestingly, the GHz CPUs can cost as much as $1140 and we would still be well advised to use them in the cluster. In this extreme case, we could afford only 16 nodes but the speed would be 32Gflops, just like for the "value-priced" 700MHz-based cluster. The benefit of this scenario is that there are now seven fewer nodes to maintain and support with UPS and networking components. Thus, while the node costs are the same, we can probably save money on the supporting hardware.

One could argue that by buying fewer nodes, you lose proportionately more cluster cycles when a node goes down. This may be important in environments where node availability is low and for highly parallelized, robust tasks which can suffer a node loss without a break in the computation.

For us, taking advantage of the fastest CPU meant that we could use our space more effectively. Specifically, we would not have to purchase as many UPS units, saving room in the rack for more nodes.

These ideas are illustrated in the figures. In Figure 1 we let ks be the ratio of speeds of the most expensive CPU to the best-value CPU. For example, if the most expensive CPU is 1GHz and the best-value CPU is 700MHz, ks = 1.4. With a fixed budget we buy as many of the best-value nodes as possible in one case and as many nodes with the most expensive CPU in another. Let the ratio of the speeds of these clusters be S. The figure uses the arbitrary value of $200 for the cost of the best-value CPU. Notice that when S = 1 the speeds of

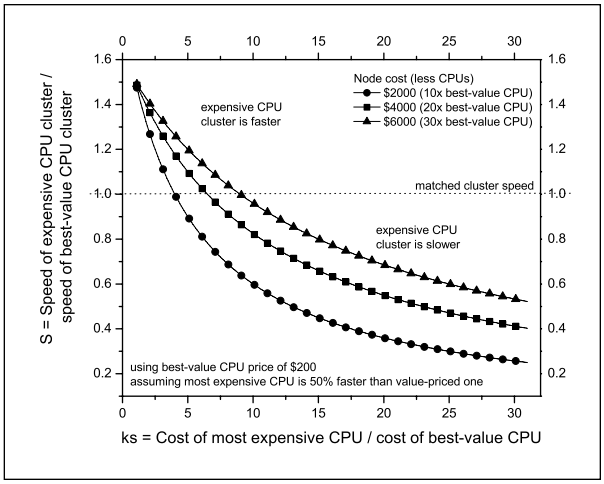


Figure 1

LINKS:
PC magazine price-performance Java viewer:
http://www.zdnet.com/pcmag/features/cpu/

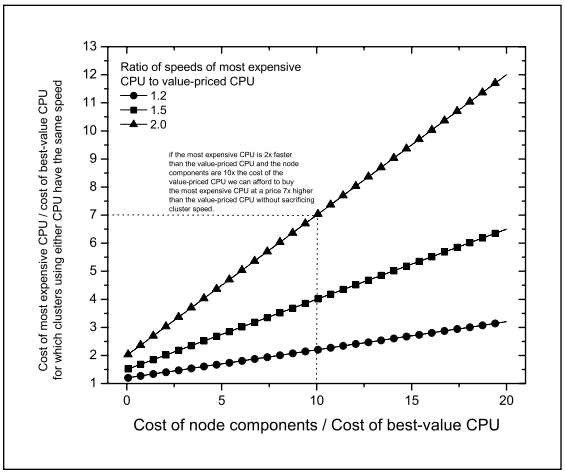


Figure 2

LINKS:
Myrinet:
http://www.myri.com/myrinet/performance/index.html

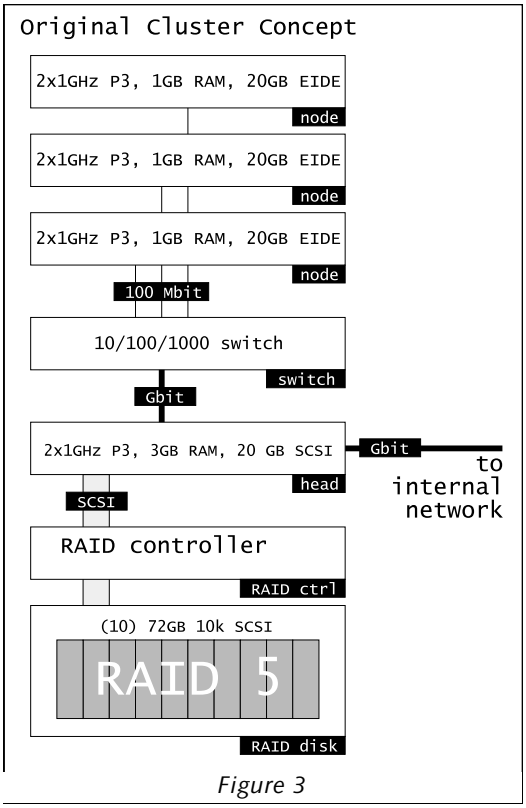


Figure 3

the two types of cluster configurations are the same. In other words, we can afford fewer nodes, but their CPUs are faster and the overall speed is the same. If our node components (memory, case, hard drive, etc.) cost 20x the price of the value-priced CPU ($4000) then we can afford to pay as much as 6.5 times more for the most expensive CPU, assumed to be only 50% faster.

Figure 2 shows the relative price of the fastest CPU vs. relative node component cost at which the two clusters have the same speed.

Even if one cannot afford as many nodes as would make the fast-CPU cluster have the same overall speed, the fact that each node is faster can be of significant advantage, particularly when the cluster is never fully utilized. These calculations were much more meaningful in the early parts of 2000, when the cost ratio between CPUs was much larger than their speed ratio. During May 2000, a 650MHz CPU was $130 and a 866MHz CPU was $1200.

Cluster Networking and Topology

Topology

Our internal network is a private class C subnet. To avoid filling the IP space, the cluster will be relegated to its own class C subnet. The nodes will be switched within the subnet. The main cluster management node will serve as an NFS server as well as a router for the nodes, connecting the cluster switch and the Centre's subnet switches using gigabit Ethernet. The topology is shown in Figure 3.

To keep the cost down and the cluster flexible, we decided to go with an Ethernet interconnect. Vendors such as IBM use Myrinet technology to connect nodes. This fabric provides lower latency than TCP/IP. Anticipating that we will not have many applications which will require numerous small messages to be passed between nodes, the cost of this technology, in our case, was better transferred into additional CPUs.

Networking

The choice of the switch for the cluster was motivated by our prior experience with managing a stack of 3Com Superstack II 3300 units which we use for the Centre's internal network. The drawback of these units is the four-unit limit on a logically managed stack. Furthermore, the stack is controlled by a central management unit whose failure brings down the entire stack. More sophisticated switches are circularly daisy chained in a stack so the failure of any one switch does not affect another. Secondly, the Superstack II units only have a single expansion slot. The stack master requires a matrix module to control the stack, and therefore one is left with only three expansion slots per stack. Using these for gigabit expansion modules, the maximum number of gigabit ports is three ports per 96 100Mb ports.

We wanted a switch which would stack more flexibly and contain more expansion modules. Not anticipating very heavy network loads, a switch with a full wire-speed backbone is not necessary, although the requirement for multiple gigabit connections to a single switch, to support multiple centralized servers and reduce the network bottleneck at these machines, exists. A chassis switch, which has multiple slots for various modules, is too expensive and bulky for our needs. However, the HP Procurve 4000M switch is a well-priced, 5U switch with 10 expansion slots which can be filled by a number of modules. It can receive up to 10x8 Mb ports, making it a very dense 80-port 100Mb switch. It can also receive multiple gigabit expansion modules, making the network topology flexible. We can always add additional giga-

bit-connected servers without the worry of running out of expansion slots on the switch. This particular unit is a good solution when the traffic on the network does not saturate the backbone of the switch and when there is a need for multiple gigabit connections.

LINKS:
HP Procurve 4000M:
http://www.hp.com/rnd/products/switches/switch4000/summary.htm
3Com Superstack II 3300:
http://www.3com.com/products/en_US/detail.jsp?tab=features&pathtype=purchase&sku=3C16981A-US

ELECTRICAL SYSTEMS AND UPS

Powering the cluster using UPS units was probably one of the most challenging aspects of constructing the specs. We cannot afford the space for a centralized, hard-wired UPS solution. All UPS units beyond 5000 VA must be hardwired. Multiple UPS units would have to be bought. Supplying power drops in the network room was a project for the building electricians. At the time, we had only four 15A/120V circuits in the network room.

When considering non-hardwired UPS units to power a large number of computers, the main requirement is that the number of UPS units be kept small and as much of the available electrical power as possible be transferred to the nodes.

For example, APC's 5000RM UPS requires a 30A/208V circuit. While the circuit can deliver 6000VA of power, the UPS only uses 5000VA. Furthermore, to deliver power to the cluster nodes, which require 15A/120V input, a transformer would have to be purchased to step-down the UPS voltage. The transformer, however, can deliver only 4500VA through its NEMA 5-15 outlets. This particular UPS and transformer together take up 7U in the rack. This is a comparatively bulky solution and does not optimally deliver all power coming into the UPS to the nodes.

Our requirements to deliver as large a fraction as possible of the incoming power into the nodes is due to the management of generator power in the building. All plugs for computers have generator backups, and it is important to maximize the use of all circuits to facilitate the plant operations staff in load balancing and expansion planning. The Centre has its own distribution panel within the building. The number of drops on the distribution panel is limited. Considering that a 208V circuit requires two drops and a 120V circuit requires a single drop, we could more efficiently use the drops if a 120V UPS unit could be found. Ideally, the Tripplite SMART3000RM2U is a 2U unit which uses a 30A/120V circuit. It can deliver 3000VA of power to the workstations and has nine NEMA 5–15 outlets. Tripplite is currently unique in being able to provide a slim, powerful UPS system designed specifically to power 15A/120V components. We anticipate that other vendors will follow in marketing similar products.

We estimate the average load of our nodes to be 130W (220VA). Each UPS could therefore support about 9–10 nodes at 70% load. Ideally, no power strips or transformers would have to be purchased. Looking ahead, in a 42U rack in which 5U are used for the switch, 2U for the head node and 3U for a disk tray, we have room for 32U of UPS/nodes. Given that a 2U UPS can support 10 1U nodes, the effective node width is 1.2U and the rack capacity is 26 nodes with three 2U UPS units. Using the APC UPS, for example, would require 2 UPS units which would take 14U, leaving only 18U for the nodes!

As most vendors do, Tripplite offers an environmental sensor which can be used to monitor the temperature in the room – a very useful backup thermometer for our enclosed space. Tripplite offers Linux drives and is considered one of the more Linux-friendly vendors.

The nodes were placed in an HP Rack System/e. This rack has the advantage of an available vertical extension kit which expands the rack from 41U to 49U. This rack system is

LINKS:
Tripplite SMART3000RM2U:
http://www.tripplite.com/products/family/ups/smart_pro/2u/index.cfm
APC 5000 RM:
http://www.apc.com/resource/include/techspec_index.cfm?base_sku=SU5000RMT5U&language=en&LOCAL.APCCountryCode=US
Powerware 9125:
http://www.powerware.com/Products/9125/product.asp
HP rack systems:
http://www.hp.com/racksolutions/p_erack.html
Great Lakes rack equipment:
http://www.greatcabinets.com/Standard_Cabinets/gl720-32.htm

extremely sturdy, supporting 2000-pound static loads, and is also available in a very minimalist style. Panels and doors may be added. A fan tray of nine 75 cfm fans was bought to facilitate air circulation through the rack. Our cooling requirements call for a high air flow in the room, probably making the fan tray unnecessary.

NODE MANAGEMENT

One could say that any number of networked computers is a cluster. With added management tools and software, a group of computers can be managed and used as a single entity. Many of the vendor solutions package some cluster-management software.

Cluster management can be broadly divided into two areas: system maintenance and administration; job queueing and monitoring. Often two separate packages or sets of programs would be used for these distinct tasks. Chances are, regardless of what the purpose of the cluster is, given an operating system there will be some cluster administration software available for it. There are a number of open source programs that will mirror, maintain, and logically administer multiple machines.

Because our cluster is a research cluster, we expect that its use will range from job distribution by simple constructs like rsh to more complicated job queueing using dedicated packages. Very likely, a number of in-house parallel applications will be developed using PVM- or MPI-style libraries. Currently, we see the cluster job management challenge as an ongoing effort to logically address the nodes in a transparent and efficient fashion.

LINKS:
Cluster management software:
http://www.npac.syr.edu/techreports/hypertext/sccs-748/cms-table.html; http://www.nhse.org/NHSEreview/CMS/index.html
Beowulf:
http://www.beowulf.org/software/software.html
Systemimager:
http://www.systemimager.org
TORC:
http://www.epm.ornl.gov/torc

It will be up to the system-administration scripts and applications to ensure node homogeneity and availability and to keep monitoring cluster statistics and utilization. One option is to boot from a floppy. During this process, nodes would be instructed to fetch specific system files and libraries from the head node. Multiple floppies could be used to load various node configurations. Once the nodes are appropriately configured, it will be important that they form a coherent collective which is not affected by the loss of one or more nodes. Both KickStart, which comes with RedHat distributions, or SystemImager can do this well.

One of our initial attempts to create some low-level structure in the cluster is through the use of hostnames. Each node is named XofY where X,Y are integers in the range 0–9. We split the cluster up into groups of 10 nodes (e.g., 0of0, 1of0 ... 9of0). This simple partitioning, in which hostnames depend on incremental group index and node index within the group, allows users to address nodes in an automated way through their scripts — they can automatically construct hostnames rather than remembering IP ranges. We expect that while we are deploying and testing higher-level management packages, users will be able to take advantage of this. Developing tools for user management will be an ongoing and evolving effort.

COOLING

By our estimation of a load of 130W per node, and using a general load of 130W per 1U of rack equipment, a 42U rack will produce about 19,000Btu/hr of heat. It is important to provide fully redundant cooling for a space as small as ours. In the case when cooling fails, a small room such as ours will significantly heat up, leading to damage to the equipment.

We chose to contract the cooling out to an engineering consulting firm to assist us in choosing the correct solution. This work is ongoing. Some of the considerations were as follows:

First, it is important that the temperature be as constant as possible. Some cooling systems cycle (turn on and off) in an effort to keep the temperature constant. When the load is much smaller than the capacity of the system (which can typically cool at a fixed rate) the system can cycle frequently, leading to increased wear. Some units have a minimum cycle delay, which can cause the temperature to rise significantly between cooling cycles. Consequently, choosing a cooling system which can adapt to varying loads is crucial. A typical computer room fills up with heat generating equipment slowly over time but cooling is purchased at the outset. It is wise to purchase sufficient cooling right away to handle the anticipated future loads. For example, our small room can at most house three computer racks. Using the numbers above and adding in 3 KW of power drawn by switches and other equipment, we find that the entire room will never require more than 66,000 Btu/h of cooling, assuming that 130W per 1U can be used.

Choosing a cooling system which can adapt to varying loads is crucial

Generally the controls of the cooling unit can be adjusted to match the requirements of the environment. Ideally, one should monitor the temperature and humidity in the room with an external, possibly redundant, probe. Units can be bought which attach to UPS devices, or to serial ports of nodes. Linux kernels can have LMSensors compiled into them which gives access to motherboard and CPU temperatures for some motherboards.

Secondly, the volume of air flow in the room should be considered. If the room is very small, the air circulation can make working in the room uncomfortable. Our cooling requirements call for some 4000 cfm of air flow. For a 100 sq. ft. space this makes for a breezy room.

Finally, it is important to plan the air flow to facilitate the existing cooling dynamics of the nodes. The cases will generally draw air from the front and exhaust out the back. Therefore, cool air should be delivered in front of the racks, and warm air should exhaust from behind the racks.

Monitoring the temperature of each node in the cluster can provide not only valuable environmental information but also give some measure of how temperature varies from node to node in a rack. Possibly, if a node's temperature reached some critical limit an automated script could shut down the node or send an email alert.

VENDOR-BOUGHT OR ASSEMBLED

Our original concept was to build our cluster using the rackmountequipment.com RC0101 cases and locally purchased components. We chose the Tyan Thunder LE server board for its compatibility with LMSensors, the ability to house 4GB of RAM, and the two 64-bit slots. Integrated Ethernet is useful for us, especially considering that the networking loads will be low. Alternatively, to minimize packet loss and maximize throughput reliability of a 100Mb connection, a server-class network card such as the 3Com 980 TXM can be used.

Very soon after finalizing the cluster specifications, we were approached by IBM to partner with them in building a cluster. We were offered their x330 nodes at a significant educational discount, paralleling our own costs to construct the nodes in-house.

The x330 node embodies many of the qualities that we were seeking through various vendors. The case is ultra-cooled with six internal fans, giving redundant cooling if any one fan should fail.

The problem of connecting all the nodes to a monitor is addressed by daisy chaining keyboard, video, and mouse for as many as 42 nodes to a single keyboard, mouse, and

LINKS:

IBM x330 review:
http://techWeb.techreviews.com/sections/topics/article/TT20001114S0002/1

IBM ASM guide:
ftp://ftp.pc.ibm.com/pub/pccbbs/pc_servers/25p2574.pdf

Tyan Thunder LE:
http://www.tyan.com/products/html/thunderle_p.html 3com 980

TXM:
http://buydirect.3com.com/iom_dcms/b2c/catalog/detail.html?SKU=3C980C%2DTXM&SM=

monitor. This is done via a special console port which combines the three signals, requiring a single cable between the nodes. Each node has a selection button on the front which connects it to the input devices and monitor. We are already finding that this is an invaluable tool. Additionally, it saves the space of having to manage cables and KVM switches.

The built-in SCSI 160 and hot-swappable 18GB 10,000 RPM disks make for crisp I/O. The facility to add another disk means that the disks can be mirrored with RAID to enhance availability.

The power consumption of an x330 node is very close to 130W when CPUs, memory, and disk are fully utilized. The draw spikes to about 160W during booting.

We have received nearly all of the parts for our cluster and are in the process of building and testing the components. We expect that the final construction will be complete in the next month or so.

the PIRUN cluster at Kasetsart University: the most powerful supercomputer in Thailand

by Putchong Uthayopas

pu@ku.ac.th

Putchong Uthayopas has been involved in Beowulf clustering technology for the past 5 years. He is the head of the SCE (Scalable Cluster Environment) project. Currently, he is an assistant professor at the Department of Computer Engineering, Faculty of Engineering, Kasetsart University, Thailand.

and Somsak Sriprayoonsakul

Somsak Sriprayoonsakul is in the Master Degree Program ofthe Department of Computer Engineering at Kasetsart University.He has worked as a system administrator for numerous Linux Servers and has helped build many clusters. He is the author of a scheduling system on Beowulf Cluster called SQMS (Simple Queueing Management System) which is a part of the SCE integrate software environment for Beowulf Clusters.

In the past, high-performance computing was considered by many as too costly to be widely practiced in Thailand. Most of the high-performance computing research in Thailand has been done by a small group of people who used small commercial supercomputers operated by government agencies and universities. These machines had only about four to six processors per machine with a computing speed aggregate of only about 2–3Gflops. As Beowulf clusters become widely used by organizations around the world, the cost problem mentioned earlier is resolving itself. Due to the high performance and low cost of these types of systems, Beowulf technology seems to be the solution for developing countries like Thailand.

Beowulf Cluster Development at Kasetsart University

Kasetsart University is the second oldest university in Thailand, established on February 2, 1943. Current enrollment is around 25,000. Our research laboratory (the Parallel Research Group) was founded in 1996 to explore Beowulf clustering technology and its possible applications. We have since built many clusters.

We started our cluster adventure by building a small five-node 486SX Beowulf cluster that ran MPICH. The first machine's performance was around 2–3Mflops. It was nevertheless a real thrill to finally have a machine for which we could write a parallel program and then actually run it. The next several machines that we built grew rapidly in terms of size, complexity, and performance. Finally, the moment that we had long been waiting for arrived when, in 1999, the university's computing center had a budget to build a 72-node Beowulf cluster. With this size, the machine is definitely the most powerful supercomputer in Thailand.

The PIRUN Beowulf Cluster

One of the most important tasks was to find a good name for the machine. After several serious discussions among the group, we decided to call it PIRUN (Pile of Inexpensive and Redundant Universal Nodes) Beowulf cluster. Coincidentally, PIRUN is also the name of the Thai god of rain and the god image that our university uses in all of its logos.

The goals in building this machine were to:

- Build a system to serve as a centralized Internet super-server for 15,000–20,000 Internet users at Kasetsart University (KU).

We decided to reduce some cost by having this system consist mostly of diskless nodes.

- Provide a world-class supercomputing facility for researchers at KU in areas such as computational chemistry, computational fluid dynamics, bioinformatics, and computer science.
- Build a large PC cluster to be used as a test-bed for cluster computing technology.

PIRUN Beowulf Cluster Hardware

The PIRUN cluster system comprises:

- Seventy-two compute nodes built with Pentium III 500MHz, 128MB RAM per node, ASUStek P2B series motherboard, that support wake on LAN and hardware monitoring chips.
- Three file server nodes using a dual Pentium Xeon 500MHz server with hardware RAID. Each server node has about six Ultra SCSI disks of 9GB for a total of about 54GB per node.
- KVM (Keyboard/Video/Mouse) switch with daisy-chained capability. A total of 10 KVMs have been chained to centralize the console access to a single node.
- Four 24-port, stackable, Fast 3Com Superstack II Ethernet switches are used to link the system together.

We decided to reduce some cost by having this system consist mostly of diskless nodes. We developed an in-house tool to aid in installing the system ourselves. These tools and much more are available for download at the main Web site: *http://smile.cpe.ku.ac.th*. The whole configuration cost around $200,000 in 1999.

Planning the System Configuration

We decided to divide the 72 nodes into three sets of 24 nodes each. Each node is named CPIRUNxx, where xx is the corresponding IP number of that node. For example, CPIRUN11 has IP number 192.168.10.11. (IP addresses have been changed to protect the innocent.) The file servers are named FPIRUN1, FPIRUN2, and FPIRUN3. Each set of 24 nodes uses the diskless file system (/, /usr, /lib, /var, etc...) from each FPIRUN to lessen the file server load. The organization is:

- CPIRUN11–CPIRUN34 use file on FPIRUN1
- CPIRUN41–CPIRUN64 use file on FPIRUN2
- CPIRUN71–CPIRUN97 use file on FPIRUN3

The tool we used to build the diskless file systems is called the "Diskless Cluster Suite" (version 1.2) and was built by us for this purpose. This tool is also available for download from our site and from the Tucows site (*http://www.tucows.com* and search for the name under the Linux category).

For our setup, FPIRUN1 also provides the common space for the mail spool directory /var/spool/mail for all nodes. All 72 nodes mount the users' working directory (/home) from each FPIRUN. However, there are special partitions for many particular purposes. For example: FPIRUN1 exports /home directory for staff members. FPIRUN2 holds /home2, which contains temporary space provided for research projects that use a very large disk space. Finally, FPIRUN3 holds /home3 for student users. (This is not guaranteed to be as reliable as the staff volume.)

Building and Installing the System

Around December 1999, the equipment finally arrived. The first task was to connect all the hardware together, so we rounded up a group of volunteer students and faculty members and had fun getting things out of boxes, making cable connections, and put-

ting lots of systems on racks. Things went very smoothly, and we were able to finish the installation within a few days. The first problem we encountered was the power system; there were some mistakes in balancing the power phase, and things needed to be rearranged. So, first lesson: be careful, these clusters really need power and a large floor space.

Be careful, these clusters really need power and a large floor space.

The software installation was our next consideration. First, the three file servers FPIRUN1, FPIRUN2, and FPIRUN3 were installed using Linux. We chose Linux RedHat 6.2 (the best and newest at that time). Second, the Diskless Cluster Suite tool was used to generate the initial configuration. Due to the complexity of installation, some configuration had to be performed manually to meet our requirements. For convenience, we allowed each FPIRUN to rsh to each other without a password prompt by creating the file /root/.rhosts that contained the fpirun1, fpirun2, fpirun3 hostnames. Of course we also added fpirun1, fpirun2, fpirun3 hostnames to /etc/hosts. For better security, this has now been replaced by SSH.

To create the diskless configuration on the nodes, we used Diskless Cluster Suite three times on each of the file servers, each time creating a configuration of 24 nodes. All we needed to do was to merge all configurations into one and distribute it to each diskless space (each directory under /tftpboot of each FPIRUN). For the DHCP configuration, the Diskless Cluster Suite software already did this part for us. We only needed to boot each diskless sequentially one-by-one the first time so that the DHCP server could assign a correct IP for each node. DHCP always remembers the client's MAC address, so next time, all the nodes could be booted at once. (We set the DHCP lease time to infinite to get this effect.)

For the hosts' configuration, we had to fix the /etc/hosts to contains all hostnames and IPs of all nodes. The Diskless Cluster Suite had already done this for each node set. Hence, the only task we needed to do manually was to merge the /etc/hosts from all hosts into one and distribute it back to each FPIRUN file server.

File System Configuration

Diskless Cluster Suite generated all our file system configuration needs for each 24-node cluster as well. First, we needed to fix /etc/exports for each FPIRUN to allow mounting of the /home, /home2, /home3 partitions by adding these lines to /etc/exports on FPIRUN1:

```
/home 192.168.3.41(rw,no_root_squash)
/home 192.168.3.42(rw,no_root_squash)
/home 192.168.3.43(rw,no_root_squash)
 .....
/home 192.168.3.71(rw,no_root_squash)
 /home192.168.3.71(rw,no_root_squash)
.....
```

We did the same thing for FPIRUN2, FPIRUN3 but changed from /home to /home2 and /home3, respectively. This can be done easily with regular UNIX commands. On FPIRUN1 we typed:

```
% rsh fpirun2 cat /etc/exports | grep home | \
     sed "s/home2/home/g" >> /etc/exports
% rsh fpirun3 cat /etc/exports | grep home | \
     sed "s/home3/home/g" >> /etc/exports
```

This command uses cat to print out all contents of /etc/exports in FPIRUN2, then we grep only the home2 entry, and change it to home. The same goes for FPIRUN3, but

home3 changes to home. Of course, we need to do this on FPIRUN2 and FPIRUN3 as well. In total, we needed to issue these commands six times.

Next, we modified the /etc/fstab file on each diskless node to mount /home, /home2, and /home3 from each FPIRUN properly. This was also done by adding the entry (CPIRUN11 /etc/fstab (192.168.3.10, 192.168.3.40, 192.168.3.70 are FPIRUN1, FPIRUN2, FPIRUN3, respectively)) as follows:

```
192.168.3.10:/dev/nfsroot   /              nfs      defaults      0 0
none                        /proc          proc     defaults      0 0
none                        /dev/pts       devpts   defaults      0 0
192.168.3.10:/tftpboot/usr  /exportusr     nfs      defaults      0 0
192.168.3.10:/usr/local     /usr/local     nfs      defaults      0 0
192.168.3.10:/usr/software  /usr/software  nfs      defaults      0 0
192.168.3.10:/home          /home          nfs      defaults      0 0
192.168.3.40:/home2         /home2         nfs      defaults      0 0
192.168.3.70:/home3         /home3         nfs      defaults      0 0
192.168.3.10:/var/spool/mail /var/spool/mail nfs     defaults      0 0
```

The last four entries were added by us, and the first six lines were created automatically by the Diskless Cluster Suite. The same goes in CPIRUN41, and CPIRUN71.

In order to be able to run the MPICH program, all diskless nodes must be able to rsh to each other without a password prompt. The Diskless Cluster Suite already does this for each 24-node set; our task was again to merge them into one. This can be done easily by issuing the command:

```
% rsh fpirun2 cat /etc/hosts.equiv >> /etc/hosts.equiv
% rsh fpirun3 cat /etc/hosts.equiv >> /etc/hosts.equiv
```

Then, we needed to delete the fpirun1, fpirun2, and fpirun3 entries from /etc/hosts.equiv since we prohibit user access to FPIRUN. After finishing all that, we could easily copy all these files back in place.

```
% rcp /etc/hosts.equiv fpirun2:/etc/hosts.equiv
% rcp /etc/hosts.equiv fpirun3:/etc/hosts.equiv
```

Booting the Nodes

We used the Diskless Cluster Suite to create a boot disk for each node. This was time-consuming since we had to create 72 disks for 72 nodes. This process can be sped up using a smart parallel technique. First, create the first 24-disk set. Then, dump the disk image to a file using the following command:

```
% dd if=/dev/fd0 of=/home/diskless_image
```

Boot the first 24 nodes. Then, insert a new set of disks to these 24 nodes and have all nodes dump the disk image to the disk using the same command.

```
% dd if=/home/diskless_image of=/dev/fd0
```

These new disks, created simultaneously, are produced 24 times faster.

User Account Synchronization

We synchronized the user accounts by distributing /etc/passwd, /etc/shadow, and /etc/group to each diskless space. Using this replication instead of an NIS-based system will generate less traffic and will be faster since all user logging and verification is done on the local node. We centralized the account information at FPIRUN1, so any changes must be done there first. The later file distribution is done using rcp between file server nodes and plain cp in the diskless configuration under /tftpboot, using an hourly

crontab to copy each file to each diskless space. For example, at FPIRUN1 we add this script:

```
#!/bin/bash
#/etc/cron.hourly/copypassword
for i in /tftpboot/*.*.*.*; do
      cp -f /etc/passwd /etc/group /etc/shadow $i/etc/
done
```

At FPIRUN2 and FPIRUN3, we needed to add rcp to copy the files from FPIRUN1 to their own space.

```
#!/bin/bash
#FPIRUN2 and FPIRUN3 /etc/cron.hourly/copypassword
rcp fpirun1:/etc/passwd fpirun1:/etc/shadow fpirun1:/etc/group /etc/
for i in /tftpboot/*.*.*.*; do
      cp -f /etc/passwd /etc/group /etc/shadow $i/etc/
done
```

For configuration files like /etc/bashrc, /etc/profile, and /etc/csh.cshrc we created a /tftpboot/config directory in each FPIRUN to hold these files and used hard links from each of these files in diskless space to the file in /tftpboot/config. This will ease configuration updates by changing the file in /tftpboot/config and then distributing the file to all diskless nodes. We cannot use a soft link since the directory structure on the diskless nodes and servers are different.

Changing Passwords

Users can change passwords at FPIRUN1 only; /etc/passwd will be updated every hour. There is a catch-22, however: we prohibit user access to FPIRUN1. This can be done by creating /etc/nologin on FPIRUN2 and FPIRUN3. FPIRUN1 already has /etc/profile to deny user login. So we modified /etc/profile so that if a user logs in, it will automatically run the command passwd to change the password. The script is below and can be handy.

```
if [ "$USER" != 'root' ];then
      /usr/bin/passwd
      /usr/bin/logout
fi
```

We prohibit all access via Telnet, rlogin, and rsh from outside the cluster by using tcp-wrappers (/etc/hosts.allow, /etc/hosts.deny). We use an SSH 2.4.0 server for remote login.

Finally, we use DNS round-robin to distribute the hostname to the client. All CPIRUN nodes will have the name cpirun.ku.ac.th associated with all CPIRUN IPs. Below is the output from the nslookup command:

```
Name: cpirun.ku.ac.th
Addresses: 192.168.3.47, 192.168.3.48, 192.168.3.49, 192.168.3.50,
192.168.3.51, 192.168.3.11, 192.168.3.12, 192.168.3.13, 192.168.3.14,
192.168.3.15, 192.168.3.16, 192.168.3.17,  192.168.3.18, 192.168.3.19,
192.168.3.20, 192.168.3.21, 192.168.3.22, 192.168.3.23, 192.168.3.24,
192.168.3.25, 192.168.3.26, 192.168.3.27,  192.168.3.28, 192.168.3.29,
192.168.3.30, 192.168.3.31, 192.168.3.32, 192.168.3.33, 192.168.3.34,
192.168.3.41, 192.168.3.42, 192.168.3.43, 192.168.3.44, 192.168.3.45,
192.168.3.46
```

At this point, the machine is running, so it's time to do something useful.

REFERENCES

T. Sterling, D. Becker, D. Savarese, J. Dorband, U.A. Ranawake, and C.E. Packer, "Beowulf: A Parallel Workstation for Scientific Computation," in *Proceedings of the International Conference on Parallel Processing*, 1995.

Barry Wilkinson and Michael Allen, *Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel Computers* (Prentice Hall, 1999).

William Gropp, Ewing Lusk, and Anthony Skjellum, *Using MPI: Portable Parallel Programming with the Message-Passing Interface* (Cambridge, MA: MIT Press, 1999).

Putchong Uthayopas, Surasak Sanguanpong, and Yuen Poovaravan, "Building a Large-Scalable Internet Super Server for Academic Services with Linux Clusters Technology," in *Proceedings of Internet Workshop2000*, Tsukuba, Japan, February 15–17, 2000.

How We Use the PIRUN Beowulf Cluster

One of the problems that formerly prevented us from offering any course in parallel and distributed computing was the lack of a real parallel computing platform. In Thailand, we expect that manpower need will be greatest in the areas of parallel scientific computing and system engineering. Therefore, the courses emphasize parallel programming and algorithms more than parallel architecture. Also, the interest in cluster computing and the building of cluster infrastructure has created many new kinds of research programs and new areas of research, some of which are listed below:

- Parallel software tools and environment, cluster administrator tools, cluster integration tools, cluster middleware
- Parallel and distributed applications
- Internet search engine, parallel text search engine, Web infrastructure
- Pollution modeling, fluidized bed simulation in chemical engineering, molecular dynamics simulation, computational fluid dynamics application in vehicle design, and heat analysis in electronics industry

Conclusion

We have gained invaluable experience from the Beowulf cluster system at KU and have learned that careful planning is essential in setting up a large Beowulf system. If everything has been carefully planned, this kind of system can be put in place within one or two days.

Because of our expertise in cluster software tools, initial software installation was done quickly. Without this kind of automated scripting, the installation would be very tedious and long. However, since there is always something misconfigured or totally missing, it takes awhile for the system to become smoothly operational for users. User feedback is also important to improve the operation of the system.

Overall, we have had a very positive experience; the Beowulf cluster system has brought many benefits to our organization in terms of cost reduction, producing new expertise, and creating new projects. It was almost impossible a few years ago for our university to possess the most powerful supercomputing system in the country. Moreover, it was hard to imagine that building such a machine could be done easily using just a bunch of PCs on rack. Linux and the free software movement have actually created a miracle here. We expect use of this class of system in Thailand to increase in the coming years. There is currently an effort to form a communication network among researchers in Thailand. More information about cluster computing activities in Thailand can be found at *http://tfcc.cpe.ku.ac.th*.

Acknowledgments

The PIRUN system has been maintained by many students and staff members. First, we would like to acknowledge the contribution of Professor Yuen Povarawan, director of the KU computing center who helped push until this project became a reality; my friend, Assistant Professor Surasak Sa-nguanpong, the co-project technical leader, who spent a lot of his time selecting the most effective hardware combination for the project; Thara Angskun and Jakchai Sonakul, who first installed the system; Sugree Phatanapherom and Theevara Vorakosit, who helped tweak the machine and catch some bugs. Also, we appreciate the many, many students who helped set up the machine in the first phase. Without their kind help, the setup would have been hard and painful. Finally, we would like to thank Dr. Thomas Sterling, father of the Beowulf cluster, who inspired us all in doing this. In fact, he kindly visited us in Thailand and gave an inspiring talk at Kasetsart University. He also visited the PIRUN cluster site when we had our first 16 nodes set up and made many useful comments about the future of this project.

special purpose clusters at the Cornell Theory Center

by Dave Lifka

Chief Technology Officer, Cornell Theory Center. Lifka has played a major role in the formation of the Advanced Cluster Computing Consortium, AC3, formed in conjunction with Dell, Microsoft, and Intel. CTC is focused on using Windows 2000 clusters for production quality computing.

lifka@tc.cornell.edu

Why CTC Moved to Windows Clusters

In August 1999, the Cornell Theory Center (CTC) moved from traditional UNIX HPC to Windows-based computing with the installation of a 256-processor Dell PowerEdge Windows cluster. CTC's Velocity cluster was the first production system of its kind within an academic environment. It was our goal to demonstrate that mid-range to high-end cluster computing systems with industry-standard components can provide the scalable, reliable resources needed for today's technical and business computing environment. We hoped to leverage exceptional economies of scale by integrating software and systems from desktop to HPC system. The CTC Velocity Complex now comprises an aggregate of 250 nodes, including several special purpose clusters.

Benefits of the Move

As a result of the move, CTC has experienced a very large increase in the number of users, especially new users attracted by the Windows HPC environment. We have also seen a dramatic shift in our cost-performance ratio, which is now at least one-fifth that for our prior traditional systems, especially considering the reduced costs of management and maintenance. This reduction in operating cost allows us to stay at the leading edge of the performance curve, which is critically important when the members of your user community need to maintain their competitive edge. At the same time, we have been able to provide extremely reliable resources to our users so that they can get their work done. (All of the nodes in the Velocity Complex are running at between 99.99% and 99.999% uptime.) In fact, our success in attracting new users helped drive the implementation of several special purpose clusters.

Special Purpose Clusters

A number of our users quickly realized that they were competing for a popular resource and decided to seek funding so that they could afford to purchase their own cluster systems. CTC builds and maintains these systems for the research groups in collaboration with their staffs, ensuring that the groups will benefit from our experience and be kept up-to-date on technical advances. At the same time, they have access to a custom environment that is tuned to their specific needs. Examples of special purpose clusters now implemented at CTC include:

Cornell Institute for Social and Economic Research

The Cornell Institute for Social and Economic Research (CISER) purchased a Dell cluster system that is installed and maintained at CTC. Cornell's social scientists interactively access a wide variety of statistical software packages for analyzing their research data or to gain access to CISER's nationally renowned data repository. They transitioned their users from a UNIX system to this Windows cluster in one year and now have more than 300 individuals on the system.

Computational Materials Institute

The Computational Materials Institute (CMI) at Cornell University is involved in a wide variety of research initiatives ranging from earthquake prediction to fracture simulations. CMI users were among the first to move to the initial CTC cluster, Velocity, and quickly found that they needed exclusive access to what is a shared resource. Their first strategy was to join with a few other large groups to drive the acquisition of a second production cluster, Velocity+. However, CMI researchers run 64-processor jobs 24x7; they needed a dedicated system to meet their demand. CMI recently received NSF CISE/RI funding for a dedicated 32-node, 64-processor cluster.

USDA-ARS Center for Agricultural Bioinformatics

The research emphasis of the USDA-ARS Center for Agricultural Bioinformatics at Cornell is on the exploitation of high-performance computing and communication technologies to solve practical problems in agricultural genomics. Initially, they required a cluster system that provided rapid turnaround for large-scale similarity searches on DNA sequences from agronomically important plant species. The output of these searches was used to populate a large data warehouse. This application required short queueing delays; the ability to handle very long jobs; fast access to large, shared file systems; and tight integration with the database server. Since this type of use is not well suited to a large, shared resource, USDA-ARS decided to fund the purchase of a dedicated 48-processor cluster that is managed and maintained at CTC. This cluster is now being used for the analysis of entire genomes of bacterial plant pathogens.

Conclusion

These research groups are finding that they get just what they need out of this working relationship – dedicated, up-to-date custom systems that are well supported and easily accessible. As a result, the CTC cluster complex now includes eight systems, each unique, some general purpose, some specially designed to meet the needs of a special user community. We learn more about clustering design and implementation with each project.

For more information:

CTC: *http://www.tc.cornell.edu*

CTC's Advanced Cluster Computing Consortium: *http://www.tc.cornell.edu/AC3/*

Windows HPC: *http://www.microsoft.com/WINDOWS2000/hpc*

Computational Clustering Technology Preview: *https://microsoft.order-1.com/cctp/*

monitoring tools for larger sites

by Stephen Chan,
Chan PDSF lead. He has spent the last 10 years working either as a system engineer of as a UNIX SA.

Cary Whitney,

Whitney has worked on PDSF at LBNL since 1999, and has played a key role in PDSF's past and ongoing development.

Iwona Sakreja,

Sakrejda started in PDSF User Services in June 2000). Prior to that worked for ten years at LBNL in the Nuclear Science Division.

and Shane Canon

Canon is a system administrator at NERSC where he helps administer a large linux cluster used for computational computing.

sychan@lbl.gov
clwhitney@lbl.gov
isakrejda@lbl.gov
canon@nersc.gov

The Parallel Distributed Systems Facility (PDSF) is a "Cluster of Clusters" managed by the National Energy Research Scientific Computing Center (NERSC) personnel under the auspices of the High Energy and Nuclear Physics Computing Support (HENPC) Group.

Introduction

One of the primary responsibilities of system administrators is to ensure that systems are running and users don't experience any service interruptions. As an environment becomes larger, with more services and more dependencies, it becomes increasingly difficult to track the state of your site. For small sites, or for sites with very specific requirements, administrators often create custom scripts or other monitoring tools to watch their environment and report any problems. However, as sites become larger and more complex, or monitoring policies become more stringent, it makes sense to look for existing tools and build upon them. So long as the tool is stable and well matched for your environment, this is an efficient approach. With the proliferation of the Internet and the burgeoning open source movement, there are more tools than ever before to monitor your site. This article covers some of the most popular, and discusses their design and operation. Tools exhibit the different design decisions of their creators; administrators will have to decide for themselves whether the design suits their requirements.

Our environment is a Linux cluster with roughly 180 servers, used for scientific computing at the Lawrence Berkeley National Laboratory. While it is not as large as some server farms, it is nonetheless large enough so that a monitoring tool not designed for performance and scalability will begin to show its limitations. The server we use for monitoring is a dual Pentium Pro 200Mhz (256K cache) with 64MB of memory. Not an especially powerful machine, but it provides a good platform for testing the efficiency of monitoring packages.

The intent of this article is to present a beginning overview of monitoring and describe several packages that provide a good starting point for further investigation. For readers who have some experience with monitoring, the later section that reviews several packages may be useful, in case you are trying to expand or upgrade your monitoring system. We won't be able to go into much depth in this article, but we hope to give readers a useful comparison of the tools available.

What Is Monitoring?

There are two classes of monitoring tools that are covered in this article: event (fault) monitoring and performance monitoring. Typically both are necessary in a production site, but for this article, we will spend more time on event/fault monitoring. The event monitoring tools we'll discuss are Big Brother, Mon, Big Sister, and NetSaint. For performance monitoring, we will focus on MRTG, one of the most popular packages available for trending network performance. Inevitably, if we talk about MRTG, the topic of SNMP comes up; however, SNMP is a very broad topic, and we cannot hope to do more than provide a high-level overview in this article. It should be clear that we are only covering free, open source tools. There are numerous powerful (and expensive) commercial packages available, but many of the free packages are very useful and more than adequate for many sites.

Event monitoring is essentially watching out for certain interesting changes in the state of your systems. Each such change is an "event." Of course, the term "interesting" is intentionally ambiguous: for most sysadmins, a server experiencing a kernel panic and crashing is "interesting," but something as seemingly benign as the utilization of a finite

For system administrators, often the three most important metrics are availability, utilization, and throughput

resource going above a certain threshold may be interesting as well (even if nothing crashes). This is why we use the term "event monitoring" instead of merely "fault monitoring." Fault monitoring implies that something is broken, but we may be interested in an event, even if nothing is broken (because the event may indicate that something might break soon). Generally, if an interesting event occurs, we want some kind of response to be triggered – it can be as simple as sending a message to a pager or as complex as starting a script that performs diagnosis and possible recovery.

Performance monitoring involves tracking metrics related to how systems are performing. For system administrators, the three most important metrics are often availability, utilization, and throughput. Availability is a measure of the percentage of time that a system is up and available to users, while utilization measures what percentage of the total capacity is in use (for example, what percentage of time a CPU is non-idle). A metric related to utilization that is often used in network monitoring is throughput, or the amount of activity per unit of time (for example, the number of megabits per second flowing through a network switch port).

Performance and event monitoring can overlap because the underlying metrics being gathered are often the same. For performance monitoring, these metrics are processed to produce graphs or some kind of summary statistics. For event monitoring, changes in the metrics, the inability to collect the metrics, or the inability to connect to a service (a "service check") are the events being monitored. As a simple example, if we try to connect to a server and discover that it is not responding, to the event monitor, this event may trigger a page to the person who is on call. To the performance monitor, the fact that the server is down is a data point for calculation of overall availability.

An important protocol for both event and performance monitoring is SNMP, the Simple Network Management Protocol. SNMP is a UDP protocol based on the notion of reading and setting variables that are tied to the state of devices on a network. By reading the value of a variable via SNMP, you can discover information about the device. By setting the value of a variable via SNMP, you can alter the state of the device. Networking hardware typically has SNMP support built in, and SNMP is the main standard used for remotely administering networking hardware. Computers and other devices on the network typically support SNMP as well, but it often requires configuration.

On a computer, SNMP typically requires that an agent be installed and running. This agent handles SNMP requests and can be configured to send SNMP event notifications (SNMP traps) under some circumstances. An SNMP agent can be extremely useful for gathering metrics and remotely administering machines. However, this utility comes with the requirement to administer the agent; an unconfigured or poorly configured SNMP agent is a security nightmare.

Most of the event monitoring tools discussed have their own custom agents. SNMP agents are the de facto standard for performance monitoring, but they can also be used for event monitoring. The choice of whether to use an SNMP agent or the custom agent is up to the administrator. Event monitoring packages usually prefer to use their own agents for gathering metrics, and their default configuration usually doesn't support SNMP agents. However, the security and robustness of these agents can be a big unknown – while SNMP is a security hazard, it is at least an understood hazard. Typically, an SNMP agent is much more general than the custom monitoring agents, and with the appropriate investment of time, it is more powerful and flexible.

Event Monitoring Packages

There are four network monitoring packages that we will discuss: Big Brother, Big Sister, Mon, and NetSaint. Each one is a free, open source package that can be found on the Internet. For the most part, these packages are stable and can be used in a production environment to do monitoring. All of these packages can be extended with plug-ins or, if you are so inclined, by modifying the source code. In addition, they all have Web interfaces – one package, Mon, has a command-line interface as well.

All of these packages will perform basic monitoring, such as pinging hosts or checking if common services (HTTP, Telnet, etc.) are listening. All of the packages support monitoring using *polling* ("pulling" information from services being tested) and some of them support *pushing,* in which clients send information to the central monitoring package ("pushing" information from the clients being monitored).

Polling tends to concentrate all the work on the machine that is doing the polling. Consequently, the polling machine can become bogged down. The benefit is that you have only a single point of administration, simplifying management dramatically. With pushing, where clients send information to the monitoring host, more of the load is borne by clients, but this can require more administration. It also means that an agent has to be installed on each of the machines being monitored, and these machines must be able to initiate connections to the monitoring host. This can be a problem if there are trust issues – for example, if the monitoring host is inside a firewall, but the monitored host is outside the firewall. You typically don't want external hosts initiating connections through the firewall. This is an issue that an administrator needs to consider very carefully in light of the security policies for his or her site.

All of the packages described support external plug-ins. These are custom written programs that monitor services that the basic monitoring package doesn't support. For the most part, the plug-ins are external scripts with well-defined exit values and output that let the system know the state of the tested service. Writing a plug-in in C and then compiling to native code is generally fastest in terms of performance; however, coding something in Perl is usually the most convenient approach. Some of the packages have an embedded Perl interpreter to speed up Perl-based monitors. This is an important consideration if you have a lot to monitor and prefer to use Perl. It is also useful because SNMP offers many monitoring possibilities that may not be supported in one of the canned monitoring tools.

Plug-ins are also valuable for testing more complex applications. For end-to-end testing of an application, it may be necessary to engage in an extended transaction (for example, testing an e-commerce application on a Web server) by writing a custom client. This is an important consideration for monitoring more complex sites.

Big Brother

Big Brother is one of the most popular packages available. It is straightforward to install and configure, and has a large user base that has produced numerous plug-ins for monitoring services. Big Brother is a combination of shell scripts and compiled C programs that will gather information and generate reasonably photogenic Web pages that provide up-to-date status information. If something breaks, Big Brother has a highly configurable policy-based notification system that supports email, pagers, and SMS. Notifications can be acknowledged via the Web page or an email message.

Big Brother also has a script that generates availability statistics, which is nominally a part of performance monitoring. With the base installation, you can monitor the fol-

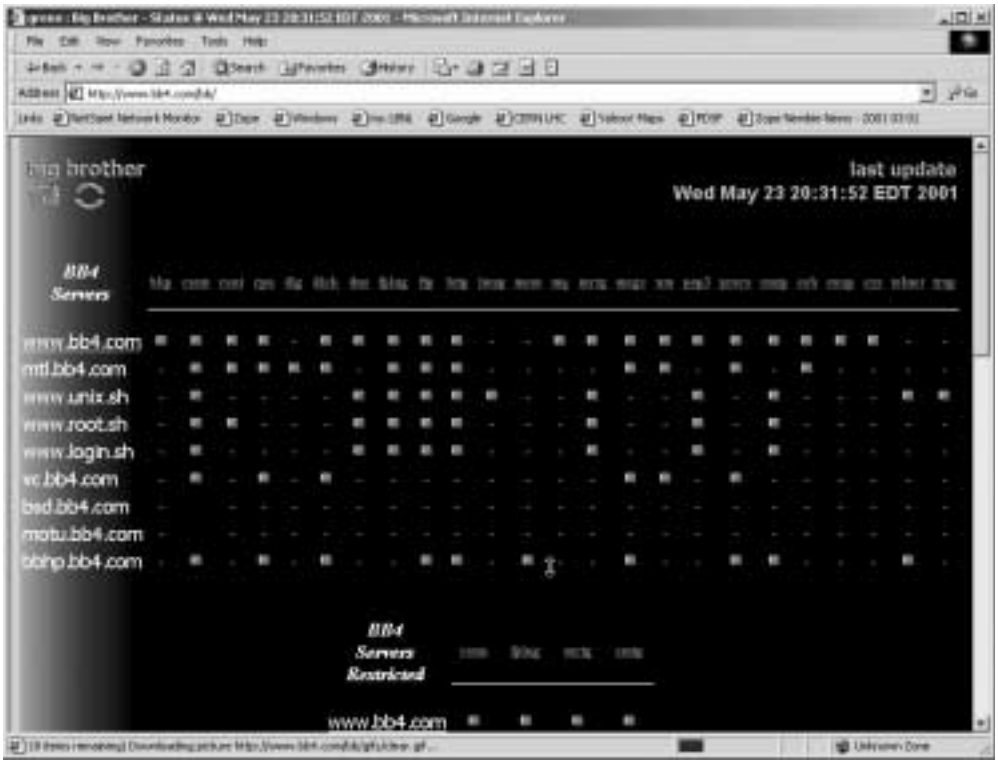


Figure 1: Demo Screen from Big Brother

lowing services: FTP, SMTP, POP3, Telnet, SSH, NNTP, DNS, HTTP, HTTPS. Note that monitoring of HTTP and HTTPS requires the Lynx browser. Plug-ins, in the form of external programs, are supported to monitor services that the base installation does not. Big Brother can monitor most services you can think of, and if a service isn't covered, you can easily write a plug-in using their interface.

If an agent is installed on a machine, it will push information about running processes, disk space, CPU utilization, and similar metrics to the Big Brother server. All this information can be tied to notifications as well, so if an important process dies and doesn't respawn, Big Brother will let you know. Even though Big Brother collects this information, it only stores enough to perform availability reporting and not utilization or performance tracking.

Big Brother does not have SNMP support built in, but it can be extended to support SNMP traps and SNMP polling via plug-ins.

Big Brother keeps its state information in a collection of text files. Since they are text files, they are relatively easy to parse; however, text is not the most efficient format for storing data, or for accessing data. If you keep historical data for a long period of time, the disk space usage really starts to add up.

Big Brother also has support for redundant Big Brother installations and some support for distributed monitoring. This is handy for remote sites and can also be used to scale up the capacity of Big Brother.

Our experience is that Big Brother is excellent for smaller sites, but it is missing some features necessary for monitoring larger sites. For larger sites, you need the ability to control the rate at which tests are being executed; if you have 700 different services being monitored, you don't want to run all 700 at the same time. By the same token, you would not want to run the tests one at a time, because the time to go through all 700 tests may far exceed the interval between tests. It is entirely possible that you will receive error notifications, not because something is down, but because you cannot effectively

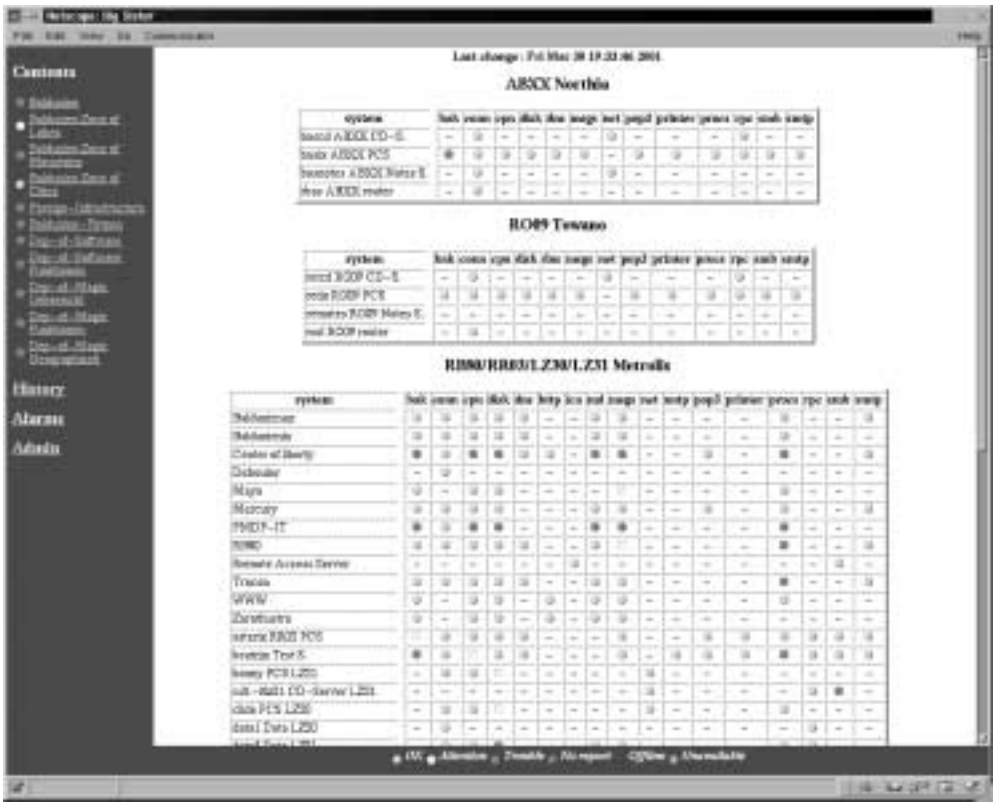


Figure 2: Demo Screen from Big Sister

perform all the tests within the allotted time. The portions of Big Brother that are in C run very quickly; however, the portions in Bourne shell are lacking in performance. Our experience is that Big Brother was not able to run all its tests quickly enough to avoid false errors showing up. It may be possible to avoid this by using a distributed monitoring approach; we chose to examine other tools, however, to see if they would work without requiring more hardware.

Big Brother's reporting tools are reasonable, but it would be better if the data that Big Brother collected were in a database with a standard interface, so that reporting tools could be leveraged to generate custom reports.

Big Sister

Big Sister is a clone of Big Brother. It is compatible with many of Big Brother's plug-ins and clients, and adds many new and useful features. One of the most basic differences is that Big Sister is implemented in Perl and uses the round-robin database tool (rrdtool) to store performance and utilization statistics for trending.

Big Sister's user interface is structurally similar to Big Brother, and it has most of the same features as described for Big Brother. Big Sister goes beyond Big Brother in the following areas:

Performance trending – Big Sister stores performance data in a database and generates graphs to describe performance and utilization. The database back end, rrdtool, is the same tool used for many network performance monitors and has C, Perl, and command-line interfaces for gathering information and generating graphs. If you need this feature, the rrdtool approach has benefits that will be discussed later.

SNMP support – Big Sister can use SNMP agents to gather statistics. However, it appears that SNMP information may not be used for trending. Big Brother supports SNMP as well, via plug-ins, so it is less well integrated.

Syslog parsing – Big Sister will examine the syslog file for errors or other lines matching configurable regular expressions. This is especially handy if you have a centralized loghost.

Tripwire – Tripwire is used to verify file permissions and checksums on important system files for security auditing. This can save you some time if security is a major concern.

Additional monitors – Big Sister has several other monitors built in, which are often covered by Big Brother plug-ins. For reference purposes, the additional monitors include: Oracle, RPC, SAR-based metrics, RADIUS, OpenView trap monitor. In addition, Big Sister comes with support for SNMP traps – event notifications that devices send over SNMP (instead of being polled, a device pushes an event to the server). The benefit of a built-in monitor is that they often have much better integration with the core package and can present information in more detail and many of the built-in Big Sister monitors provide a good level of detailed information.

Big Sister also uses rrdtool for storing and graphing performance metrics. rrdtool will be discussed more in the performance monitoring section; suffice it to say that this adds a lot of flexibility and power with respect to gathering and displaying metrics.

Generally speaking, Big Sister has more monitors in the base installation and better tools for reporting, especially generating graphical reports.

Big Sister does not really add any new functionality when it comes to support for larger sites. The fact that Big Sister is written in Perl is both a boon and a bane. Perl is excellent in terms of modifying and extending the software; however, Big Sister is relatively complex for a Perl script. Perl's garbage collector makes it easy to write code, but the reference-counting implementation tends to be "leaky" on long-running scripts that use lots of objects. In our tests, we found that the resident set size of the Big Sister application could get to 20MB within a few days. This was only aggravated by the fact that we had many machines and many services being monitored. This is not a criticism of Big Sister, but of the limitations of Perl for complex, long-running programs (especially if there are lots of anonymous objects being created, as typically occurs when using the OO extensions). If Big Sister were re-implemented in the latest version of Python, which has an improved garbage collector, or Ruby, with its Mark&Sweep garbage collector, this problem might be avoided. This problem can also be worked around with a nightly restart of the offending Perl scripts.

As memory usage goes up, performance on the monitoring host often starts to degrade. It appeared that Big Sister did not parallelize its tests. Combined with the slowdown from memory usage, it became impossible to complete all the monitors within the scheduled amount of time (15 minutes between tests), resulting in many systems appearing to go down and then coming back up a few minutes later.

Because of the lack of support for larger sites, and the problem we saw with core leaks, Big Sister, despite it's many attractive features, does not seem to be appropriate for a larger site.

Mon

Mon is a set of Perl scripts that, in terms of functionality, is one of the more basic tools covered. However, one of Mon's more interesting facets is that it has multiple interfaces into the system – supporting command-line, Web, and even two-way-pager interfaces. Mon is implemented in Perl, but apparently because the package does not implement all

the monitoring and trending features of Big Sister, it doesn't seem to suffer from the same garbage collection problems.

Mon can be described as a minimalist scheduling and notification framework for monitor programs. It schedules tests to be run and, based on the results, may send out notifications. It has a server that clients can connect to in order to query and update the state of the system, which then implements a Web, command-line, or other interface to the end user.

The following monitors are provided: asyncreboot (monitor host reboots via SNMP), dialin, DNS, fping, FreeSpace, FTP, hpnp, IMAP, LDAP, MySQL, netapp quota/FreeSpace, NNTP, ping, POP3, monitor processes via SNMP, rd (notifies if too many or too few files are in a directory), RPC, SMTP, TCP, Telnet, and network round-trip times. Like Big Sister, Mon supports SNMP traps.

Mon also has event handlers – which are programs or scripts that should be run when certain events are detected. For example, you could configure Mon to run a utility to restart a Web server on a remote machine if it notices that the server has gone down. This is a very useful feature for problems that are well understood and can be automatically resolved.

Mon doesn't have its own agents, preferring to leverage SNMP for many of these functions. In general, Mon is a straightforward monitoring tool. It does not provide a large feature list, but it is dependable, has a low administration overhead, and allows many forms of interaction. Its configuration is very clean and easy to read, more so than the other packages we examined. The Web interface also has useful controls for stopping and restarting the Mon server process. In terms of it's core monitoring and notification functionality, it is on par with most of the tools described, but it does not have a snappy Web interface, trending, or availability reports.

For large sites, Mon's scheduler has a useful feature: it can put a limit on how many tests are being run at any given time. We actually ran Mon for quite some time and it did a

Figure 3: Demo Screen from Mon

good job of watching the site without bogging down. If you need a basic level of monitoring that is dependable and straightforward to configure, but with very little reporting, Mon is a solid performer.

NetSaint http://netsaint.sourceforge.net/

NetSaint is a monitoring package that seems to have been designed for speed and scalability. The package is written in C, and virtually all of the monitors are coded in C as well. For folks who prefer Perl, the latest version includes an embedded Perl interpreter to eliminate the overhead of forking and execing a new Perl interpreter. The package also has one of the more visually appealing Web interfaces of the four we've examined. It should be stated that NetSaint is the package we are most familiar with because it seemed to fit our requirements well, and, as a result, we've invested more time in exploring it.

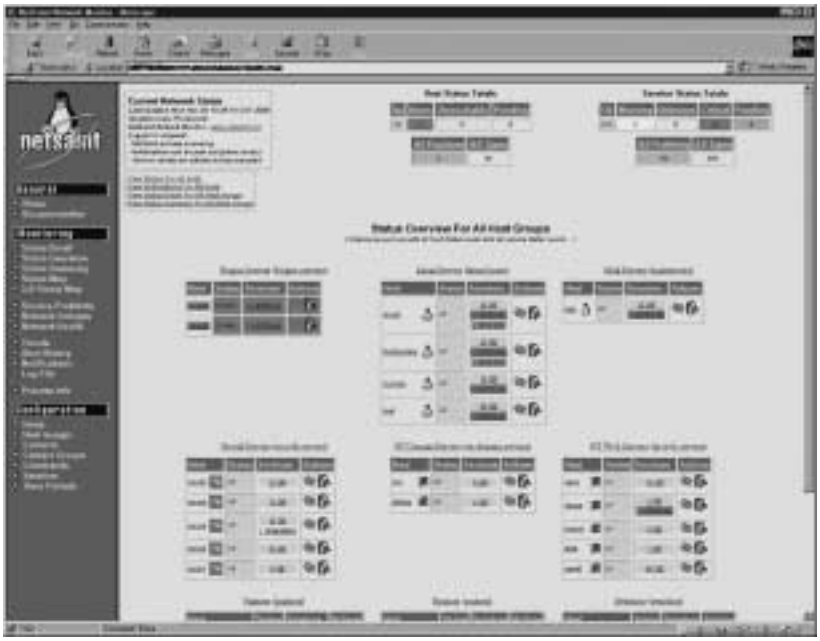


Figure 4: Demo NetSaint Screen

NetSaint has many useful features for supporting larger sites. It parallelizes service checks and provides directives for controlling the maximum number of service checks, as well as a method of interleaving the service checks so that checks are spread out uniformly. With these features proPerly enabled, NetSaint was able to easily complete 700 total service checks against ~180 hosts with extremely low load on the monitoring host (usually 90% idle).

Another feature that NetSaint supports is passive service checks. This allows service checks to be performed elsewhere and then sent to a NetSaint server for tracking and notifications. This distributes the load across multiple machines and can be very useful for large sites or for monitoring remote sites.

NetSaint also has many other useful features including exposing the number of and interval between retries before sending out an error notification. It will often be the case that there is a transient problem that causes a test to fail or time out. Under these circumstances, you would prefer that the monitoring package briefly wait before retrying the test to avoid a false alarm. In many packages you can modify the script or program that does the checking, but in NetSaint, it is parameterized in the configuration file. This was very handy in cutting down on false alarms against our more heavily loaded NFS servers.

NetSaint also generates reasonable availability summaries and trending graphs. However, NetSaint doesn't do any performance or utilization trending. It *does,* however, have an interface that allows performance data from tests to be passed to external trending packages.

NetSaint also has a very handy feature: when using the Web interface, it is possible to put notes on machines or add notes to services that have been marked as "down." This is an excellent feature for communicating between UNIX system administrators, operators, and even end users. It is also a good way to keep a history of problems with machines.

If there is a place where NetSaint is less convenient, it is with the configuration files. The files allow you to have very fine control over the machines being monitored; however, it is not possible to apply a directive to a group of hosts at one time. For example, if you have 180 hosts that you want to monitor for SSHD, you need to enter 180 lines telling NetSaint to do so. The file is also designed for machine parsing, not human parsing, and can be a little hard to read. But you can get around tedious and error-prone configuration file editing with some scripting. An enterprising user came up with a tool that uses NMAP to discover all the services and then output a configuration file of all the hosts and services listed. This is clever and a big time saver, but it isn't a substitute for more expressive configuration file directives. NetSaint would benefit from looking at Mon's approach to configuration.

NetSaint has possibly the best default Web interface, with several very useful views, and a good amount of optional information that can be added. However, it doesn't allow the degree of customization over appearance that many of the other packages provide.

Performance Monitoring Tools

Performance monitoring is an important topic, so we cover it in order to complete the picture of monitoring, and also to contrast it against event monitoring. Due to the short length of this article, we can only touch on the subject briefly, by discussing the motivation for performance monitoring, typical functionality, and one of the most common free packages, MRTG.

Monitoring the utilization and other performance metrics on your systems is important for capacity planning and load balancing. It can help you identify bottlenecks in overall site performance, understand usage patterns, and predict when extra capacity will be required. In contrast to event monitoring, which is typically concerned with a sudden state change, performance monitoring is concerned with describing long-term trends, or providing statistical summaries of system state. The availability statistics that many event-monitoring packages provide is an example of a performance metric and also highlights the relationship between event monitoring and performance monitoring. Many performance-monitoring packages can also be configured to trigger an action if some metric falls below a certain level. For example, a package that is tracking the amount of swap space can generate an alarm if free swap falls below a certain value. This tight relationship between event and performance monitoring is why they are both covered in this article.

MRTG, MultiRouter Traffic Grapher

MRTG is an SNMP-based package, originally designed for monitoring network usage on switches and routers. MRTG is a very popular tool, and many sites may already be using it to watch their network usage. MRTG generates a series of graphs that chart the input and output utilization of ports on switches and routers. This is the most common use of MRTG. However, MRTG has three features that make it especially interesting as a complement to event monitoring: external data sources, rrdtool support, and threshold triggers.

EXTERNAL DATA SOURCES

MRTG can be configured to collect and graph arbitrary pairs of variables served over SNMP. The following URL describes how to monitor server-based metrics over SNMP with MRTG: *http://net-snmp.sourceforge.net/tutorial/mrtg/index.html.*

Figure 5: Demo MRTG Screen

However, sometimes information that you'd like to have is not available fromSNMP. The following URL has links to many such applications: *http://people.ee.ethz.ch/~oetiker/Webtools/mrtg/links.html*.

An example from our site is information from our batch scheduling system about the number of servers currently being used to process batch jobs. Another useful metric is the total number of servers that are available The availability metrics that event-monitoring systems provide usually only measure the availability of individual machines or services, not the availability of the entire cluster. A useful metric would be what percentage of the total site is up at a given time and summary statistics of long-term availability. These can be easily fed into MRTG and graphed using the external data source features.

An issue with external data sources is that they are polled at the same time that MRTG runs its normal SNMP probes, which may or may not be the appropriate polling interval. The next feature we discuss provides a way around this issue.

RRDTOOL SUPPORT

MRTG keeps an internal database of the metrics it has gathered. What rrdtool does is to break out the functionality of this database into a separate tool called the round-robin database tool. This tool keeps a sliding window of the most information gathered – for example, it will only keep the most recent 1000 datapoints, no matter how many you may have actually collected. rrdtool allows you to perform calculations and generate graphs on the data and any calculations you may have made. rrdtool is actually the database back end for the next generation of MRTG; however, recent versions of MRTG can be configured to use rrdtool for its back end.

Because you can insert data into rrdtool independent of MRTG, rrdtool allows much better control over data collection and increases your control over the graphs being generated. MRTG is designed to graph two variables against each other on its graph – this is not always the kind of graph that you are interested in – sometimes you are only interested in a single value being graphed, or you may want multiple values graphed simultaneously. Another issue is that the sampling interval for MRTG may not be the appropriate sampling interval for other services – MRTG performs all the metrics gathering at the same rate. This is especially important if the process of gathering the metrics is expensive or time-consuming: it may be reasonable to gather information about all the ports on a single switch every five minutes, but trying to gather CPU utilization across 180 servers over SNMP every five minutes can be more trouble than it is worth.

If you are willing to invest some time, rrdtool solves both these problems. rrdtool can store arbitrary time series data, and gives many options for creating graphs. As an example, the following image is a graph of processor utilization across our server farm. The source data comes from our batch scheduler (LSF), the data and stored in rrdtool, and a cron job runs rrdtool every 15 minutes to generate an up-to-date graph.

Another advantage of using rrdtool is that it can easily serve as a bridge to the event monitoring package – the event monitoring package can sample the most recently gathered statistics and generate an alarm if something is out of the expected range of values.

THRESHOLD TRIGGERS

MRTG can also be configured to generate alarms if certain metrics go outside an acceptable range. In this capacity, MRTG operates as a fault/event monitor and is a useful adjunct to the main event monitor. Threshold triggers and rrdtool to interface with the event monitor are both effective for tying the event monitoring and performance monitoring systems together.

Figure 6: Sample rrdtool Output

Conclusion

Monitoring is a key aspect of system administration, especially for sites that have 24x7 requirements. Even for sites without such requirements, administrators may be better off having their monitoring package inform them that something is wrong with their systems before a user does. The packages described cover a good range of monitoring requirements, and if used well, should allow an administrator to know what is happening on their site in excruciating detail.

At the minimum, these tools provide an administrator with clear metrics about the performance of their site. Ideally, these tools can clarify broader policies and provide insight into capacity planning and resource utilization.

large clusters for theoretical physics at Fermi Lab

by Don Holmgren,

Don Holmgren is the leader of the Distributed Systems Projects Group (DSP) of the Integrated Systems Development Department of the Computing Division of Fermilab.

djholm@fnal.gov

and Ron Rechenmacher

Rechenmacher is an engineer in the Distributed Systems ProjectsGroup of the Integrated Systems Development Department of the Computing Division at Fermilab in Batavia, IL.

ron@fnal.gov

Introduction

The Fermi National Accelerator Laboratory (*http://www.fnal.gov*) is a US Department of Energy facility located in Batavia, Illinois, about 35 miles west of Chicago. Scientists at Fermilab work on understanding elementary particles, the area of science known as high energy physics. One area of study, performed by theoretical physicists, concerns quarks and gluons. Sets of quarks, bound together by gluons, compose particles such as neutrons and protons, which in turn make up most of the known mass of the universe.

Lattice gauge quantum chromodynamics, or lattice QCD for short, is a numerical technique for modeling the interactions, or QCD, between quarks and gluons. Physicists employ lattice QCD to help interpret the results of experiments, with the ultimate goal of determining the validity of the "Standard Model" of physics. The calculations of lattice QCD are very floating-point intensive, requiring anywhere from a few days on a machine capable of gigaflops for simple calculations, to several years on machines capable of teraflops for the most complicated problems. Once the domain of commercial and purpose-built supercomputers, lattice QCD now often finds a home on clusters of commodity computers.

At Fermilab, we currently operate a lattice QCD cluster consisting of 80 systems, each a dual 700MHz Pentium III computer connected to a high-bandwidth, low-latency Myrinet network. We regard this as a prototype production cluster, and on it we seek to develop the techniques and codes which we will use in the next few years on clusters of 1,000 or more computers.

Lattice QCD codes attempt to solve the Dirac equation using non-perturbative methods. The continuous fields and wave functions of this four-dimensional partial differential equation are discretized onto lattices represented by multidimensional arrays, essentially the same finite-difference technique employed in electromagnetic and structural engineering analyses. As in finite-difference methods, differences in the values of the modeled functions between a given lattice site and its neighbors approximate various derivatives.

Unlike conventional finite-difference techniques, the quantum mechanical properties of the modeled particles require additional corrective terms. Such properties manifest themselves on the distance scales characteristic of the particles, such as neutrons and protons, which are composed of quarks. Lattice spacings on the order of 10^{-16} meters must be used, and lattice sizes of L^4, where L ranges from as low as 6 to as high as 64 or even 128, are employed. Many quantities, in the form of complex 3x3 matrices and complex vectors, must be stored to represent the model at each lattice site.

A large lattice QCD problem may thus require tens or hundreds of gigabytes of memory. Starting from a random configuration, the lattice QCD code will iterate a number of times over the lattice, performing at each site a set of floating-point operations on the values stored at neighboring sites. Such an algorithm may be parallelized and the work spread across a number of processors in order to reduce the time required to obtain a solution. When running on a cluster of computers, the code distributes the lattice over the memory of the machines of the cluster and uses a message-passing API, such as MPI

or PVM, to communicate the values stored at lattice sites at the boundaries between computers. The most common lattice QCD code used on the Fermilab lattice QCD cluster is known as MILC and was developed by a consortium of physicists at a number of universities (see *http://physics.indiana.edu/~sg/milc.html*).

In order to solve the problems of interest over the next decade, teraflops-scale facilities are required. The 80-node production prototype at Fermilab sustains approximately 10Gflops on MILC codes. Therefore, we envision building clusters on the order of 1,000 computers ("kiloclusters") over the next years. To reach the teraflops scale, we will depend both on large numbers of processors and on the heretofore steady increase in processor performance described by Moore's law.

In order to build and operate kiloclusters, we need to develop tools and techniques which scale. In the rest of this article, we discuss a pair of examples. First, we describe the network-based tools we use to install the Linux operating system on our nodes and to upgrade or modify the BIOS and other firmware of our computers. Second, we describe command-invocation tools which we use to perform common administration tasks simultaneously on all nodes of our clusters.

We envision building clusters on the order of 1,000 computers ("kiloclusters") over the next years

Network Installs and Firmware Upgrades

On large clusters which execute programs similar to our lattice QCD codes, it is essential that all of the nodes be as nearly identical as possible. The codes implicitly assume this homogeneity, spreading the lattice evenly across the systems. Each iteration of the lattice requires the same calculations be performed at each site. The slowest machine in the cluster dictates the rate at which iterations occur. Thus, no difference in hardware or software configuration which might affect performance can be tolerated on any of the computers. Further, minimizing the amount of state held by any system in the cluster simplifies the replacement of a failed unit.

We achieve homogeneous software configuration by installing identical system images on the machines in the cluster, modifying only those files related to identity (hostname, network address, etc.). We achieve homogeneous hardware configuration by installing the same versions of the BIOS and other firmware on each of the nodes of the cluster, and by setting identical BIOS configurations on each system. We minimize the man-power required for these operations by employing automated, network-based installation and firmware upgrade tools.

We have grown to fear what we call "the 15-minute catastrophe." Commodity hardware occasionally requires "out of band" maintenance – the sort of operation in which the system administrator hooks up a keyboard and a video monitor, or reboots the system from a special floppy disk. For a cluster of 10 systems, an operation which requires 15 minutes of attention per computer is inconvenient. On a cluster of 1000 systems, this same operation can be a disaster – 1,000 systems times 15 minutes per system equals six person weeks.

For some operations, we cannot avoid directly interacting with a node. For our 80-node cluster, we selected a motherboard, the L440GX+ from Intel, which allows redirection of BIOS input and output to a serial port. We have connected each of the nodes to a serial multiplexer, and from any X Window session we can open a window and interact with the BIOS in a manner identical to using a directly attached keyboard and video monitor. This motherboard also has an "emergency management port" (EMP). Via a second serial line connected to the EMP, we can reset or power cycle the computer.

Operating system installation and BIOS upgrades are two examples of "out of band" maintenance, which usually requires booting from a special floppy disk or CD-ROM. Instead, we rely on the network booting capability available on certain brands of Ethernet interfaces. Installation of Linux on a node in our cluster proceeds as follows. First, on our boot server we modify the /etc/bootptab entry corresponding to the system requiring installation. Our Ethernet interfaces (Intel Pro/100) have PXE (preboot execution environment) support. When reset, the systems execute the PXE code, which checks for a DHCP or BOOTP server willing to provide a system image. Peter Anvin's "pxelinux," a variant of the "syslinux" used on many Linux distributions' installation boot floppies, is loaded via TFTP, as specified in the server's bootptab file. "pxelinux" in turn uses TFTP to load and start a minimal memory-based Linux image. Via the standard init.d mechanisms, this image fetches and starts an installation script. The script partitions the computer's disk drive and makes file systems, fetches master tar files, explodes these tar files, modifies network configuration scripts, and runs LILO to set up future booting from the hard drive. Via rsh the script then unmarks the /etc/bootptab entry on the file server. Finally, the script resets the node. The node then boots from the newly installed image on its system disk.

The installation of Linux over the network using this technique completes in less than five minutes. Other than the initial modification of the server's bootptab file, the operation is unattended. We can reinstall Linux onto all 80 nodes of our cluster in about 40 minutes. The limiting factor in such mass installs is the network bandwidth available from our server. Scaling to a kilocluster will require either multiple servers or multicast techniques. The PXE code from Intel already includes a multicast TFTPD and a multicast TFTP client in the BIOS. However, multicast TFTP clients for Linux are not currently available.

Network-based BIOS installation proceeds along similar lines. Vendor-supplied BIOS installation programs all require booting the system into an MS-DOS or similar environment. Normally this requires running the operating system from either a floppy disk or a system disk. Fortunately, the Netboot and Etherboot projects (see *http://etherboot.sourceforge.net/*) have contributed codes which patch standard DOS to run out of a RAM disk. A BIOS upgrade using these tools proceeds as follows. As before, the server's bootptab file is modified and the system to be upgraded is reset. The PXE code on the system's NIC solicits a DHCP or BOOTP server. "pxegrub," from the GNU Grub project, loads via TFTP and starts. "pxegrub" in turn fetches a DOS image tagged by the Etherboot tools. DOS starts from this image, which includes a large RAM disk containing the various binaries and datafiles required for installation of the BIOS. The standard autoexec.bat mechanism is used to launch the firmware upgrade binaries. Unfortunately, we have not been able to use the same rsh technique to undo the modification to the server's bootptab file. So, prior to rebooting the system, we must manually reconfigure this file, then reboot the node. BIOS installation via this technique completes in less than five minutes.

The installation of the BIOS also results in modification of various BIOS parameters to their default values. In some cases, these values are not appropriate for our cluster. We have had some limited success under Linux with restoring parameters via the special /dev/nvram device. However, we often must manually alter parameters via direct interaction with the various BIOS menus.

Parallel Command Tools

Nearly every system administrator tasked with operating a cluster of UNIX machines will eventually find or write a tool which will execute the same command on all of the nodes. At Fermilab we call this tool rgang. The history of rgang begins in the distant mists of farm computing at Fermilab. It was originally scripted in an afternoon by Marc Mengel to deal with the common administration and installation problems associated with the IBM farms that Fermilab was using. While its code has changed, the basic use of the tool remains the same. rgang relies on files in /usr/local/etc/farmlets/ which define sets of nodes in the cluster. For example, all lists all of the nodes, row1 lists all of the nodes in row 1, and so forth. The administrator issues a command to a group of nodes using this syntax:

```
rgang farmlet_name 'command arg1 arg2 ... argn'
```

On each node in the file farmlet_name, rgang executes the given command via rsh or ssh, displaying the result delimited by a node-specific header. rgang is implemented in Bourne shell.

Because rgang executes the commands on the specified nodes serially, execution time is proportional to the number of nodes. In Python, we have implemented a parallel version of rgang. This version forks separate rsh/SSH children, which execute in parallel. After successfully waiting on returns from each child or after timing out, parallel rgang displays the node responses in identical fashion to rgang. In addition, parallel rgang stores the OR of all of the exit status values in a shell variable. Simple commands execute via parallel rgang on all 80 nodes of our cluster in about three seconds.

To enable scaling to kiloclusters, parallel rgang can utilize a tree, via an nway switch. When so invoked, parallel rgang uses rsh/SSH to spawn copies of itself on multiple nodes. These copies in turn spawn additional copies.

There are currently two major modes for rgang.py: command mode and copy mode.

The command syntax for the modes are 1):

```
rgang.py  [options] <nodes-spec> <command>
```

and 2a):

```
rgang.py -c [options] <nodes-spec> <file> <dir|file>
```

or 2b):

```
rgang.py -c [options] <nodes-spec> <file> <file>... <dir>
```

Of all the various options, the most significant is the –nway=number option. This option determines the "fan-out" behavior of rgang. The default, when the option is not specified, is 0. This makes the fan-out equal to the number of nodes specified, which causes the rgang action (command or copy) to occur in parallel. Other values for nway, cause rgang.py to attempt to build a tree-structure in a "worm-like" fashion to accomplish its task.

For instance, a value of 3 for nway with a list of 10 nodes will attempt to fan out the process in a manner illustrated by diagram 1.

Contrast the above structure with diagram 2, which results when the nway option is not specified (the default nway):

In order for the first tree structure to work, the rgang.py script must be executable from each node that is not at the end of a branch. However, the structure of the copy mode is

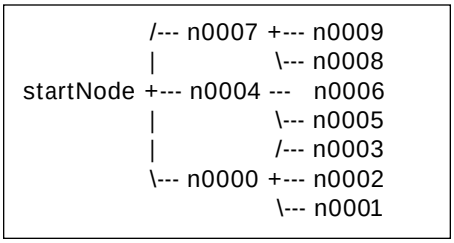


Diagram 1

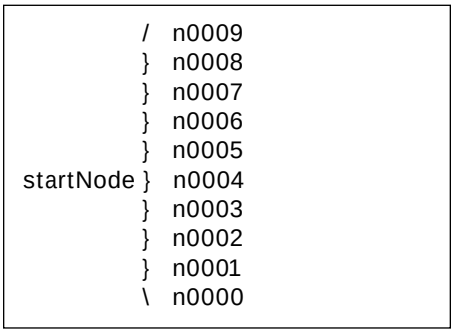


Diagram 2

such that rgang.py can be used to copy itself to the list of nodes. It copies the file to the downstream node first, then, if required, rshells to the node and executes the rgang.py script to continue the propagation.

In other words, rgang.py can be used to install itself.

The system used in the following example is our QCD cluster, which consists of a "home" node and 80 "worker" nodes. The rgang.py script is symlinked to rgang and is installed (executable) from all 81 nodes. The output in the following examples is edited to save space.

Example 1:

```
$ time rgang -nn "{,}qcd0{1-8}{01-10}" echo hi
qcd0101  =  hi
qcd0101  =  hi
qcd0102  =  hi
qcd0103  =  hi
qcd0104  =  hi
qcd0105  =  hi
qcd0106  =  hi
qcd0107  =  hi
qcd0108  =  hi
qcd0109  =  hi
qcd0110  =  hi
qcd0201  =  hi
[edit]
qcd0704  =  hi
qcd0705  =  hi
qcd0706  =  qcd0706: No route to host
qcd0707  =  hi
qcd0708  =  hi
[edit]
qcd0703  =  hi
qcd0704  =  hi
qcd0705  =  hi
qcd0706  =  qcd0706: No route to host
qcd0707  =  hi
qcd0708  =  hi
qcd0709  =  hi
[edit]
Command exited with non-zero status 1
1.45user 2.18system 0:03.84elapsed 94%CPU (0avgtext+0avgdata
0maxresident)k 0inputs+0outputs (20209major+37650minor)pagefaults 0swaps
```

In Example 1, the -nn option specifies a more compact output format. The nodes specification {,}qcd0{1-8}{01-10} results in specifying 160 nodes. (The node names result from the physical configuration of 8 shelves with 10 nodes per shelf.) At the time of the example, node qcd0706, was powered down. The output, exit status, and bulk of the 3.84 seconds reflect this "problem."

Example 2:

```
$ time rgang -nn "{,,}qcd0{1-8}{01-10}" echo hi
Traceback (innermost last):
      File "/usr/local/bin/rgang", line 688, in ?
         if __name__ == "__main__": main()
      File "/usr/local/bin/rgang", line 673, in main
```

```
        try: total_stat,ret_list = rgang( sys.argv[1:] )
      File "/usr/local/bin/rgang", line 531, in rgang
        sp_info = spawn_cmd( node_internal_info[mach_l_idx], opt, opts, args,
branch_nodes )
      File "/usr/local/bin/rgang", line 298, in spawn_cmd
        sp_info = spawn( opt['rsh'], sp_args, 0 )
      File "/usr/local/bin/rgang", line 237, in spawn
        pid = os.fork()
OSError: [Errno 11] Resource temporarily unavailable
Command exited with non-zero status 1
0.30user 0.49system 0:02.73elapsed 28%CPU (0avgtext+0avgdata
0maxresident)k 0inputs+0outputs (367major+24467minor)pagefaults 0swaps
```

In Example 2, an extra comma is added to the node specification, resulting in 240 nodes being specified. The default value for nway is 0, which causes nway to reset to the number of nodes specified, as mentioned above. This means that 240 rsh's are forked (in parallel) and the system cannot handle this! The next example shows one of the possible solutions.

Example 3:

```
$ time rgang -nn —nway 40 —skip qcd0706 "{,,}qcd0{1-8}{01-10}" echo hi
qcd0101  =  hi
qcd0102  =  hi
qcd0103  =  hi
qcd0104  =  hi
qcd0105  =  hi
qcd0106  =  hi
[edit]
qcd0807  =  hi
qcd0808  =  hi
qcd0809  =  hi
qcd0810  =  hi
1.44user 0.63system 0:02.71elapsed 76%CPU (0avgtext+0avgdata
0maxresident)k 0inputs+0outputs (5292major+8393minor)pagefaults 0swaps
```

In Example 3, the UNIX system limitations are avoided by using the nway option, and the skip option is demonstrated. The resulting status is now success.

Example 4:

```
$ time rgang -c -nn —nway=5 —skip qcd0706 "qcd0{1-8}{01-10}" \
     /boot/lsdel.out~ /tmp
qcd0101= qcd0102= qcd0103= qcd0104= [edit] qcd0809= qcd0810=
0.44user 0.31system 0:04.31elapsed 17%CPU (0avgtext+0avgdata
0maxresident)k 0inputs+0outputs (1601major+2320minor)pagefaults 0swaps
```

In Example 4, a file of size 1364734 bytes is copied from the home node, through a tree structure, to 79 nodes (connected to a big switch, all with 100Mb NICs) in 4.31 seconds. 1364734/(1024*1024) * 79 / 4.31 = 23.86 MBps

We have also implemented a tree-based tool, clstcon, which can send individual keystrokes rather than commands to all of the nodes in a cluster. When started, clstcon establishes a tree of shells across the cluster. The stdin of the root node propagates via the tree to all nodes. As an extreme example of the capabilities of the structure, an administrator can invoke a vi session across the cluster, simultaneously editing the files with the same name on all of the machines.

The rgang and clstcon utilities are great time savers.and cost. These programs have been released by Fremilab under the "Fermi Tools" umbrella. The terms and conditions are stated at *http://www.fnal.gov/fermitools/terms/TERMS_AND_CONDITIONS*

Conclusion

Within a few years, we will build and operate kiloclusters at Fermilab for work in theoretical physics. Because of our manpower limitations, we are actively seeking and developing scalable tools for the management of such large systems. For more information on our work or to try some of our tools, please see *http://qcdhome.fnal.gov/*.

eden: a home beowulf

by Robert G. Brown

Robert G. Brown is a theoretical physicist teaching and doing research in condensed matter physics at Duke University. Over the last decade his research has demanded ever more computer power and expertise which he has obtained by using entire networks of workstations as one big "cluster." He is one of the most prolific contributors to the "beowulf" list, where he is known as "rgb."

rgb@phy.duke.edu

Introduction

The Duke University Physics Department is a longtime user of cluster computing, and was an early adopter of Intel/Linux clusters when Intel's P6 family was introduced in the form of the Pentium Pro (which had quite tolerable numerical performance for its time at a far lower cost than competing proprietary UNIX platforms). Over the last eight years I've probably been the largest consumer of compute cycles on campus, at one time running a single computation on more than 100 hosts scattered all over campus and more recently running more or less continuously on the department's Beowulf, Brahma, which I originally designed and built.

Over the years, participation in the Beowulf mailing list has taught me a great deal about Beowulfery, parallel computation, cluster computation, and so forth. Beowulfery has become something more than a hobby and less than my primary profession (which is theoretical physics). I find designing, building, and working with Beowulf-style compute clusters both interesting and just plain fun. I also have an obsession with penguins and open source. Consequently, my clusters tend to run Linux rather than, say, Solaris or Irix.

Naturally, I've had a computer at home capable of connecting to the department network for nearly 20 years now. Over the years, this has transmogrified from an IBM PC with a 1,200-baud modem running Kermit into my current dual Celeron with a firewalled DSL connection; somewhere in there (about the time I could afford a 100BT switch) my home computer(s) became my home network became my home Beowulf: eden. To be really picky, my home cluster, since I have both dedicated and desktop nodes necessarily mixed on my home network.

eden has proven to be an immensely valuable tool for me professionally. It serves as a totally protected sandbox where I can try things out, prototype things, experiment, and develop code. It has been a solid base from which to begin writing a book on the engineering of Beowulf clusters. It has enabled me to develop tools like procstatd (available on the Brahma page) to help monitor and manage Linux LANs and clusters. It has helped me develop benchmark tools and methods like cpu-rate (also available on the Brahma page) to support the engineering and management of clusters.

Sure, a lot of this could be (and was) done at Duke on Brahma's nodes and workstations (also a mixed cluster, with a Beowulf component and a distributed desktop component), but Brahma is generally a production environment and it isn't always easy to find an idle node to benchmark or an idle set of nodes to run a code prototype on.

One final advantage that eden has provided me is that five or six years ago I co-founded a company, Market Driven, by writing and contributing its core operational modeling software (an advanced neural network engine). Market Driven isn't really a "Beowulf company" (it actually does stuff like Web design and product development coupled with predictive modeling and database technology), but of course Beowulf methodology and cluster methodology have formed an important component of tools like the neural network. Having a home cluster to work on has kept my evening and weekend efforts "Duke Resource Free," which is an important thing to be able to demonstrate when worrying about ownership and interest.

One great thing about a home Beowulf is how easy it has become to make, or perhaps a better term might be "grow," one. In the following section I'll discuss the engineering of

Figure 1: eden's core systems "neatly" arranged on over-the-counter shelving. Left to right: 300MHz Celeron rgb.adsl.duke.edu/adam (the gateway, note DSL plug on the wall), 466MHz Celeron abel, and dual 466MHz Celeron lucifer. The switches and so forth are on the middle shelf, printers on the top.

Figure 2: The front-end arrangement of eden. The monitor is connected to lucifer (left bottom), a dual 466MHz Celeron. Note KV switch

eden and show how just about anybody with a similar need or interest can spend at most a few thousand dollars (of their own or, better, somebody else's money) and end up with a perfectly useful and viable home Beowulf for code prototyping, production, or play.

eden's Design

eden is the simplest of all Beowulfish/cluster designs. It is basically an eight-port 100BT switch (a Netgear FS108) with a collection of mid-tower PCs (all running Linux, naturally) connected to each other via the switched network. On the front side, I have (strictly for convenience in a cluster where, like as not, a node will be opened up with some new piece of junk hardware hanging off the bus in the bottom of the case) a very cheap keyboard and video switch.

In the primary core of eden, I have lucifer as a front end and server node with two processors and abel and adam as compute nodes. Most of the components also have to do some sort of double duty: adam is a gateway, abel is a printer/scanner server, lucifer has a printer and is the primary cluster server in addition to being my desktop. The core has four processors with a total of around 1.7GHz of Celeron cycles at its disposal.

However, there are more computers in the house and all of them are or can be in the cluster (up to the capacity of my current switch). There is eve, my wife's computer (an 800MHz Tbird and the latest and fastest addition), which is available for cluster work at all times since it always runs Linux and is always on the network. Then there are my sons' computers (which they only "have" because they are part of my cluster): caine (400MHz Celeron), serpent (400MHz Celeron), and apple (don't ask). These aren't always available, but can be made available if I need their cycles. They are pictured below, in all of their kid-environment glory. The gerbil isn't really running apple (a 133MHz AMD K5),s but it might as well be.

The nodes in the boys' rooms are dual-boot nodes that can boot Windows so that the boys can do their homework on them if absolutely necessary. Note that serpent is running a typical "homework" application (Might, Myth and Magic IV). When all of them are running Linux and on the network, they represent another 1.7GHz of Intel or AMD cycles available to a computation. Even my 200MHz Pentium laptop can participate as a master in a master-slave computation with a wireless interface.

These nodes are all running RedHat Linux, although there is some version drift across them because I've been using one to test 7.1, have 7.0 on most of them, and am running the gateway with a stripped and frequently updated 6.2. They are all running PVM (my personal favorite parallel library) and can be configured into a PVM virtual machine. I also am fond of working on raw socket-based parallel code, being a bit of a masochist. An article on Brahma (see the "Extreme Linux Tutorial" from the 1999 Linux Expo in Raleigh, NC) was developed using eden, and eden was in fact carried to the EL booth and demonstrated at the Expo since my house is only about 30 miles from the Expo site.

The purpose of eden isn't really production, so I won't bore you with benchmark results or claims of fabulous speed. In fact, its 500–600 aggregate Mflops are relatively slow – two 1.33GHz Tbirds would likely replace the whole motley lot and give you back change for your dollar. eden is also highly speed-heterogeneous (as it must be, given that I buy a node or so a year), which gives me an excellent opportunity to work on load balancing parallel code. It is still perfectly adequate to give excellent parallel speedup of some chores that I need to do, and it is ideal for allowing me to do code development and testing without having to work over the DSL link.

The Future of eden

The next-generation makeover of eden (currently underway) will be significantly improved. We are having the attic remodeled into a "proper" home office for my wife (a physician) and me, and this whole space will be networked, wired, and cooled for a much more devoted-function Beowulf. In particular, I'll have some 12 built-in lines in the attic alone and a patch panel that will allow me to add the various rooms and systems downstairs to a larger switch. In addition there is the wireless network in the house that will let me work with my laptop from a hammock in the backyard on a nice day, with a cooling and refreshing beverage close at hand.

I also expect to recycle into the arrangement some old Pentium Pro and PIII-class systems that are no longer useful at Duke to give me more dedicated compute nodes (and maybe upgrade the K5 while I'm at it). Then, of course, it is almost time to buy this year's new addition from my PDA – probably a 1.33GHz Tbird, unless something better comes along in the meantime.

I doubt that I'll ever use this cluster to do a lot of real computing (I have much more powerful resources at Duke to do that with), although I certainly have put it to work in the past. However, it is capable of doing very useful computational work indeed, and as a vocational/avocational tool it is without peer. Then there is a certain cachet associated with having a home supercomputer with an aggregate performance on favorable code to what one might expect from a multimillion-dollar supercomputer only five or six years ago. Home Beowulfery is indeed a source of fun and profit for those who are so inclined. There is no reason that this little home Beowulf cannot do rational drug design or model and predict stock market performance just like its bigger brothers.

Hmmmm, now there's a useful project for the devoted hobbyist . . . .

Figure 3: Four distributed desktop units (eve, caine, serpent, apple), which can participate in a parallel computation. The mess visible in these pictures is a real mess, not a simulated mess, as this cluster is in a real house with real boys and real working parents.

home clusters

by Andreas Boklund

Andreas Boklund is a lecturer at the Department of Informatics andMathematics at the University of Trollhattan-Uddevalla,Sweden. He has also constructed, and remodelled a cluster for thermal spray and welding simulation research for the University's Department of Technology.

andreas@boklund.nu

This article is a description of the cluster that I have assembled in my home and why I did it. One of the most interesting questions is: Why would anyone want to have a cluster of computers in their home?

I have been interested in computers since my parents bought their first IBM PS/2 and I started to learn how to write simple BASIC programs. I have been programming ever since. Nowadays, I do most of my development work in C, although I occasionally write small hacks in other languages.

Back in 1997 I read an article about the Beowulf project, where NASA researchers managed to create a cluster of workstations that was so powerful that it could compete with the "supercomputers" of that time, especially in terms of power to price. The idea of parallel computing and clusters of computers is not new – it had been practiced for decades. The thing that made Beowulf clusters interesting was that they used standard PC parts that could be bought from any computer vendor. Ordinary hardware and the (then) new operating system Linux were used. For me, a computer science student, it was a dream come true. Now I could finally harness the power of a "supercomputer" in my own home. The only question was what would I use such a beast for?

The most tedious tasks that I know are to sit and wait for code to compile or for a movie to be compressed. Compressing a movie is not really a problem; it takes a long time but it can always be run overnight. Compiling is another story; when I compile something, I do not want to contemplate my code for a few minutes or even for a few seconds before being able to see if it works; maybe I am just an impatient person. The basic idea behind my home cluster was to be able to shorten the execution times for compiling and rendering, although I would not mind if the execution times of other programs could be speeded up as well.

Creating a powerful cluster is not an unworkable task if you are skilled in UNIX/Linux administration, understand the basics of parallel computing, and have proper funding. A harder task is to create a cluster with the limited resources of your home. I basically have three limiting factors: my budget, the space that the computers require, and my girlfriend (time). Based on the given factors, I constructed a cluster.

My personal cluster consists of three computers: my workstation, Phoenix, which is equipped with two Celeron 500MHz processors and 128MB of RAM; my girlfriend's workstation, Sunrise, which has a Celeron 466MHz processor and 96MB RAM; and our file server and Internet router, Sabrina, which has a Pentium II 350MHz processor and 128MB RAM. Phoenix and Sunrise have two Fast Ethernet cards each and Sabrina has three.

The network topology used is both simple and cheap, at least as long as it is used in a small network. All computers are connected through a dedicated network interface card to the other (two) computers by a crossover of twisted pair wire. Theoretically this means that all computers can send and receive information to the other two at the same time, at the maximum speed of a Fast Ethernet network card. As an extra bonus, the latencies on the network are lower than in a switched or hubbed network. The reason for this is that the data does not have to be handled by a switch or a hub, it goes straight from network card to network card. The extra, third, network card in Sabrina is connected to the Internet through which Sabrina routes all incoming and outgoing traffic. Sabrina also masquerades the IP addresses of Phoenix and Sunrise, which are from one of the free IP ranges.

The operating systems used on all three computers are various RedHat-based Linux distributions all running a Linux 2.2.x series kernel. The kernels have been patched with

the MOSIX kernel enhancements. The MOSIX project was founded in the early 1980s at a university in Jerusalem. MOSIX extends the functionality to the kernel and allows it to move (migrate) already running processes to other computers that are running a MOSIX-patched Linux kernel. MOSIX is a transparent and universal tool for moving running processes to other computers. The only items the user has to specify are the IP addresses of the computers that the operating system is allowed to move processes to. MOSIX does not know what the process is doing and it does not care; it uses a set of algorithms to decide if a process would benefit from being migrated or not. If a process is doing a lot of I/O it will not be moved since all I/O operations need to be performed on the computer that the processes were initiated on, but if the process is doing a matrix multiplication it might benefit a lot from being moved.

MOSIX is a transparent and universal tool for moving running processes to other computers

I use an extension to MOSIX called MPMake. It is a patch for GNUmake that allows several makes to be spawned over a MOSIX cluster. To make use of MPMake you have to place your source code on a shared volume and mount it on the computers that you want to compile it on. When you compile your code, you use the MOSIX-patched version of GNUmake and specify the number of processes that you want it to use. So how is the performance of MPMake compared to an ordinary make? As always when it comes to computer architectures, it depends on your application and the task that it tries to perform. The performance gain will be different depending on which program you are compiling. I did some measurements a while ago with MPMake on a MOSIX cluster of Intel Pentium III 500MHz computers, compiling a Linux 1.2.14 kernel. When using two processors we got a performance gain of 43% and with eight processors the performance gain was 79%; although this system had been hand-tuned to lower compilation time, that's not too bad! MOSIX and MPMake can be downloaded from their home page, *www.MOSIX.org*. You will also need the corresponding Linux kernel source.

MOSIX is a good tool for many issues, but it has a high demand for bandwidth and does all its I/O through the node that initiated the process. Therefore, I also installed the message-passing libraries MPICH and PVM. They were developed at Argonne and Oak Ridge National Laboratories, respectively. MPICH is an implementation of the MPI standard, which is used on a wide range of different architectures, from two-way SMP machines to large clusters and Cray supercomputers. The message-passing libraries do not work in the same way as MOSIX. When you use MPI or PVM, you have to write the program against one of the libraries. This approach is more time-consuming, and you need the source code; on the other hand, the application will execute faster. Most scientific applications that can run on clusters make use of either MPI or PVM. Message-passing libraries are faster because they start processes on other computer nodes and the processes run there, communicating with the other processes. Nowadays, I don't run many programs at home that use either MPICH or PVM. When I occasionally compress a movie sequence, I use MPICH and the Berkeley MPEG compressor.

So what does this cluster do for me? It lowers the time that is used for compiling by approximately one-third. It also speeds up the execution of all applications that can use more than two CPUs at a time, as long as they do not use a shared memory segment since my setup does not allow shared memory between computers. It also lowers the time that it takes to compress movies.

What did it cost me? I already had all the hardware except for the three extra network cards, and now I do not need a hub or a switch. It did not take me long to set it up correctly, although it took me a while to learn how to do it. But as long as you are learning, the time is well spent.

Affiliated Health Services information systems clusters

by Richard Schilling

Richard is the web integration programmer and webmaster at Affiliated Health Services, a hospital in Mount Vernon, Washington. He has a B.S. in computer-science and is currently an MBA graduate student. Richard's work with-clustering is focused on developing distributed and parallel algorithms for use in the health care industry.

RSchilling@affiliatedhealth.org

I have three clusters, though I would not classify them as true Beowulf-class machines yet. I'll describe them and the work in bridging them, their limitations, the lessons learned, and future plans.

Cluster 1: At Work

I am the Webmaster and Web integration programmer at Affiliated Health Services (*http://www.affiliatedhealth.org*). I have set up a cluster of three Pentium-based workstations that were taken from spares. The head node is a dual-named host which forms a link between the corporate network and cluster. The nodes of the cluster do not see the corporate network, but rather, the head node uses the cluster for extra file storage and processing capacity as needed. The purpose of the cluster is to establish performance minimums and a baseline equipment list for future development. I use the nodes in the cluster for file sharing. As more computers become available, I will make use of them also. In the near future, due to upgrades, I will have quite a few. As time permits, on this cluster, I benchmark various algorithms used in health care, such as decision support routines, medical billing functions, and data warehousing.

The basic approach to this is making the most of available tools while keeping the configuration as simple as possible. I'm not trying to break any speed records with this cluster but rather am focused on getting set up to solve basic business problems in health care (processing patient data) and then scaling up to heartier hardware.

Most of my time spent working on this cluster has been to solve normal business problems. Anything beyond that so far has been dedicated to trying out various tool sets and configurations. Hopefully, I'll soon get more into the serious benchmarking.

The nodes all have the same operating system, FreeBSD 4.1. Here is a list of the software I have on the nodes:

Head Node

I use the head node to start jobs and, as time permits, test out "true" Beowulf-class applications. Day-to-day use has this computer doing more Web serving and data processing related to my work as a Webmaster, such as logfile crunching.

NFS; Apache: modules include mod_ssl, mod_perl, mod_php4; Open SSL; clusterit; CVS; PostgreSQL; MySQL; Kerberos; SSH (PVM uses this for remote logins); Kaffe; OSKit; PVM and jPVM; GNU tool kits (of course); Palm development tools (prc tool set); X Windows; grass (GIS); aero (modeling); Games: freeciv, xpilots, ACM.

NODE 1

This node is used primarily for extra disk space. It hosts the CVS repository and exports the /usr partition to the other nodes on cluster.

Software: Kerberos V; SSH; GNU tool kits.

NODE 2

This node is similar to Node 1 in that it supplies extra disk space and processing power when needed.

Software: Kerberos; SSH; GNU tool kits.

NETWORK
The following is used for networking equipment in the cluster: Intel Etherexpress PRO network cards; Bay Networks Baystack 102 hub.

Cluster 2: At Work

The second cluster we have at work is in the process of being implemented. It is our main information system, and although I don't know enough about it to know if it qualifies as a true Beowulf, it sure seems like one so far. We'll have about 25 nodes, and the software is a canned package created by MediTech (*http://www.meditech.com*). Mostly, this cluster does distributed processing, distributed batch processing, and fail-safe backup work, as opposed to running true parallel algorithms. The manufacturer should be able to verify that. This will be installed this year.

Cluster 3: At Home

As if I didn't have enough to do already . . . .

Here's what the cluster at home looks like:

Sequent 2000 NUMA machine: head node (AT&T System V); two Pentium-class machines (FreeBSD 4.1); one DEC VAX 4000 (being set up; it will eventually run ULTRIX or NetBSD); one Power Macintosh (MkLinux and MacOS).

This cluster is a true experiment in heterogeneous computing environments. I have two late-model Pentium-class PCs, a Sequent 2000 NUMA machine, a Power Macintosh, and a DEC VAX 4000. Like the cluster at work, this one is being used to establish baseline minimums of hardware and software performance for present and future software projects. As opportunity arises and my software base becomes stabilized, I'll get more involved with the serious benchmarking. But this cluster also has another purpose: to reduce the steps of adding new heterogeneous nodes as much as possible. Ideally, once the operating system is installed on a machine, I'll simply plug in a new machine and start up the appropriate software.

Using the Power Macintosh is a curious choice, and I get some flak for it, but the fact is that it has turned out to be a nice little machine to use as a clustering proving ground. It has a PC daughter card in it, too, so I have a little extra PC power when I need it (while running MacOS). I also run MkLinux on this machine, which I use when I'm doing all the clustering work. Since Macintosh printers have almost always been Postscript printers, I have absolutely no problem with network printing from the other nodes on the cluster.

The two PCs mainly work as file servers at this point but will complement the Sequent and VAX boxes when I have those two set up. The Sequent machine is a recent addition and will work as the head node when it is fully configured (sometime this summer).

The focus for this cluster is to identify as many boundaries as possible in a heterogeneous system. The use of older equipment is a non-issue, since I'm sticking to the UNIX (variant) and GNU standards. Overall, this has done well, and I've even gotten the OSKit on one of the PCs to work, so I've made my first custom operating system there.

Network

A common hub bought at the local computer store, any network card that works in the PCs; Mac, VAX, and Sequent machines all have network cards built in, so I don't have to worry about those.

This cluster is a true experiment in heterogeneous computing environments. I have two late-model Pentium-class PCs, a Sequent 2000 NUMA machine, a Power Macintosh, and a DEC VAX 4000

Clustering computing is currently available and affordable for the enterprise and home environments

Bridging the Work and Home Clusters

I use my work laptop to transfer data between the home and work clusters. If the opportunity arises in the future, I'll have the home cluster dial into the work cluster.

Limitations

Ironically, the limitations I am experiencing are not primarily due to software. The OS base for Linux and FreeBSD is very stable for what I'm doing. The primary limitation I have run into so far is finding the time to dedicate to setting up the hardware, adding hard drives, and creating network settings, like IP addresses. Oh, and then there's physical space – my garage is full of computers and spare parts.

The Lesson

Here is what these two clusters have allowed me to demonstrate: clustering computing is currently available and affordable for the enterprise and home environments. I'd be lost had I not received a computer science degree and previous exposure to the UNIX environment, but it's getting easier for non-tech heads all the time.

As a general rule, it's a very bad idea to invest in the latest and greatest hardware unless it actually improves your capabilities from a clustering/functional standpoint. The idea that new (proprietary) software products can stimulate PC sales is preposterous. Hardware should only be invested in when there is too much data to handle with the existing equipment, or the cost of new equipment drops low enough to make it cost effective. Even then, you should not have to change software – recompile, yes, but never change.

Future Plans

Here is a list of future plans I have for my clusters:

- Addition of 64-bit Sun Microsystems nodes
- Work on software, including a distributed version of PostgreSQL, 64-bit
- FreeBSD, MkLinux port for G4, general benchmarking, utility set for aerospace engineering applications, modeling, Java development,
- Windows emulation, and, of course, game development
- Porting of several open source packages for the Windows environment (to try and sway the Windows crowd to the open source arena)
- Robotics
- A variety of artificial intelligence work (my computer science specialization was AI)

Linux clustering for high-performance computing

by Ian Lumb

Ian Lumb is an integration architect at Platform Computing Inc. His interests include computing architectures, parallel computation and high availability.

ilumb@platform.com

Introduction

Traditionally, high-performance computing (HPC) has involved the use of monolithic mainframe, supercomputer, or symmetric multiprocessor (SMP) systems to solve the Grand Challenge problems of the physical sciences and mathematics. Classic HPC, however, has experienced a significant evolution on two fronts. First, it has been infused with interest from extra-classic disciplines ranging from high-tech and industrial engineering, to digital content creation and the life sciences. Second, the availability of commodity processors, in combination with low-latency, high-bandwidth interconnect technologies, has catalyzed the appearance of computing architectures based on clustering paradigms. Linux clustering through distributed operating systems, middleware, and hybrids thereof, represents the current focus.

Distributed Operating Systems

Clustering through distributed operating systems (D-OSes) is not "new." However, it has been Linux-based clustering solutions that have facilitated the commoditization of HPC in nontraditional disciplines. Although MOSIX (*http://www.mosix.org/* and *http://www.mosix.com/*) is notable in its own right as a D-OS, the current illustration of clustering via a D-OS in the Linux context is provided here via Beowulf clustering (*http://www.beowulf.org/*).

In 1994, Thomas Sterling and Donald Becker of CESDIS[1] created the first Beowulf cluster out of 16 Intel 486-generation systems interconnected via channel-bonded Ethernet. An instant success at the time, Beowulf clustering has reached beyond academic circles in its ability to generate attention and has become an acknowledged genre in HPC.

Beowulf clusters consist of:

- Commodity off-the-shelf (COTS) hardware
- LAN interconnect technology
- GNU/Linux operating system
- System software
- Programming environments

Arguably, the enabling aspect of Beowulf clustering derives from the system software that allows interconnected COTS-class hardware, each running its own instance of GNU/Linux, to function as a parallel compute engine.

Although individual GNU/Linux kernel instances abstract process manipulation through the notion of a process identifier (PID), no such abstraction exists for the case in which a parent process has forked child processes that execute on physically distinct systems, each running their own instance of the GNU/Linux kernel. Generally speaking, there is no concept of a "global PID" for clustered Linux systems. To address this shortcoming, the Beowulf system software incorporates a kernel modification to provide a distributed process space (BProc) which allows:

- PIDs to span multiple physical systems – each running its own instance of GNU/Linux

ACKNOWLEDGEMENTS

Through numerous discussions, the author's colleagues Bill McMillan and Chris Smith have indirectly contributed to this article.

Beowulf clustering placed distributed-memory parallel computation in the public domain by making it simultaneously accessible and realizable

- Processes to be launched on multiple physical systems – each running its own instance of GNU/Linux

The development of this distributed process space marks a significant infrastructure advancement in the provisioning of a clustered Linux environment for parallel computation.

Ultimately, the hardware, interconnect technology, operating system, and system software collectively provide an infrastructure for scientists and engineers to develop and execute applications which exploit the distributed-memory parallel computation paradigm. Both the parallel virtual machine (PVM, *http://www.epm.ornl.gov/pvm/*) and the message-passing interface (MPI, *http://www.mpi-forum.org/*) approaches to parallel computation are supported within the Beowulf framework. In addition, various GNU compilers and tools (e.g., editors, debuggers, profilers, etc.) provide a fairly complete development environment.

First-generation Beowulf clusters generated interest for a variety of reasons:

- They demonstrated that a substantial parallel computation infrastructure could be built from readily available, and inexpensive, hardware and software components.
- They leveraged key, existing open source software in the GNU/Linux operating system, plus the programming environments offered by PVM and MPI in tandem with the GNU compilers and tools.
- They demonstrated that a distributed-memory parallel computation approach could rival, and in some cases surpass, the performance characteristics of "more traditional" serial or shared-memory programming paradigms.

In short, Beowulf clustering placed distributed-memory parallel computation in the public domain by making it simultaneously accessible and realizable.

Touted as the next-generation solution, the Scyld Beowulf clustering distribution (*http://www.scyld.com*) offers the following enhancements over its progenitor:

- Installation and administration improvements
- Efficient, single-point distributed process management
- Various 64-bit capabilities
- BProc-aware MPICH
- MPI-enabled linear algebra libraries and Beowulf application examples.

The second-generation Beowulf clustering solution provides a number of significant enhancements beyond what was available in the first-generation Beowulf clustering solution. Save for the BProc system software, a comprehensive framework to effectively manage resources that are distributed over a network remained absent. This significant shortfall is identified by D. F. Savarese and T. Sterling[2] as *The Software Barrier*:

> The earliest Beowulf-class systems were employed as single-user systems dedicated to one application at a time, usually in a scientific/engineering computing environment. But the future of Beowulf will be severely limited if it is constrained to this tiny niche.
>
> The need to enhance Beowulf systems usability while incorporating more complicated node structures will call for a new generation of software technology to manage Beowulf resources and facilitate systems programming.

They articulate the software barrier even more precisely in distributed resource management terms in the following:

Adequate system management may depend on the virtualization of all its resources. This will separate the user application processes from the physical nodes upon which the tasks are executed. The result is a system that dynamically adapts to workload demand, and applications that can perform on a wide range of system configurations trading time for space. Therefore, a new class of workload scheduler will be required, developed, and incorporated in most Beowulf systems. It will support multiple jobs simultaneously, allocating resources on a to-be-defined priority basis. It will also distribute the parallel tasks of a given job across the allocated resources for performance through parallel execution. Such schedulers are not widely available on Beowulfs now and will be essential in the future. They will incorporate advanced checkpoint and restarting capabilities for greater reliability and job swapping in the presence of higher priority workloads. Compilers to use the more complicated structures of the SMP nodes will be required as well to exploit thread level parallelism across the local shared memory processors. The software used on these systems will have to be generally available and achieve the status of de facto standard for portability of codes among Beowulf-class systems.

In addition to resource-management opportunities, there exists a tight dependency between BProc and the Linux kernel; even in the case of routine upgrades and/or patch application, this dependency needs to be considered, and implies system downtime. The architecture of BProc itself has caused scalability concerns[3] and introduced challenges in porting parallel debugging tools due to global-local PID mapping issues. As a Linux kernel modification, BProc is necessarily covered by the GPL. While this provides all the benefits that the open source approach has to offer, it makes it challenging for commercial Independent Software Vendors (ISVs, e.g., Scyld), Linux distribution providers and integrators, and traditional-UNIX system vendors to simultaneously add value and retain a differentiating edge.

Middleware

Middleware can also be used to deliver clustering solutions. In striking contrast to D-OSes, clustering via middleware does not require modifications to the Linux kernel. Instead, such middleware fundamentally provides some form of distributed process abstraction, including primitives for process creation and process control. Broadening the discussion beyond this point, and providing a clearer distinction from distributed-memory programming paradigms (e.g., MPI, PVM), is best presented through the framework of distributed resource management (DRM).

DRM is a class of middleware that facilitates the management of heterogeneous compute resources distributed over a network. DRM directly addresses the tension between supply and demand by matching an application's resource requirements with the compute resources capable of filling the need. By effectively arbitrating the supply-demand budget over an enterprise-scale IT infrastructure, subject to policy-driven objectives, DRM solutions allow organizations to derive maximal utilization from all available compute resources.

In general, DRM solutions make use of dynamic-load-state data to assist in making effective, policy-based scheduling decisions, and in applying utilization rules to hosts, users, jobs, queues, etc., all in real time. This dynamic-load-state capability has significant implications in distributed-memory parallel computing, since task-placement advice (i.e., which hosts are best suited for computational use) can be provided directly to the MPI application (on launching).

In striking contrast to D-OSes, clustering via middleware does not require modifications to the Linux kernel

A remote-execution service is required to allow computational tasks to be communicated over a network

As indicated previously, a remote-execution service is required to allow computational tasks to be communicated over a network. Although the standard remote-shell infrastructure (i.e., rsh-rshd-rexec) offers some possibilities, a more comprehensive service is required to enable:

- Authenticated communications over a network
- A high degree of transparency in maintaining the user's execution environment
- Task control with respect to limits, signal passing, etc.
- An environment for the task to execute in on the remote server

Although DRM solutions do not need to provide a distributed-process space, task-tracking mechanisms are required. Thus application-task identifiers act as a handle to the individual (parent and child) processes that collectively constitute a distributed application. In addition to providing a unique identifier for application control, such cluster-wide identifiers can be used in monitoring, manipulating, reporting, and accounting contexts.

Through the introduction of elements, the task identifier can be further generalized to the level of a one-dimensional array. This abstraction allows the same executable to be run with different inputs, while being referencable as a unit. In some circles, this approach is referred to as parametric processing.

DRM solutions typically employ a policy center to manage all resources, e.g., jobs, hosts, users, queues, external events, etc. Through the use of a scheduler, and subject to predefined policies, demands for resources are mapped against the supply for the same in order to facilitate specific activities.

The extension of DRM solutions to support the programming, testing, and execution of parallel applications in production environments requires:

- Complete control of the distributed processes making up a job in order to ensure that no processes will become un-managed. This effectively reduces the possibility of one parallel job causing severe disruption to an organization's entire compute infrastructure.
- Vendor-neutral and vendor-specific MPI interfaces
- The ability to leverage a policy-driven DRM infrastructure that is cognizant of dynamic load state

Challenges specific to the management of MPI parallel applications include the need to:

- Maintain the communication connection map
- Monitor and forward control signals
- Receive requests to add, delete, start, and connect tasks
- Monitor resource usage while the user application is running
- Enforce task-level resource limits
- Collect resource usage information and exit status upon termination
- Handle standard I/O

To illustrate the value of parallel-application management for developers of MPI applications, consider the following example of fault tolerance. It is beyond the present scope of the MPI, and indeed PVM, to take into account transient situations (e.g., a host that exhibits a kernel panic and crashes) that inevitably occur while an application is in the execution phase.[4] Such situations will affect some of the processes involved in the execution of the MPI-based application. If the application does not include some mechanism to address such situations, it is possible for the remainder of the application to run

to completion and deliver incomplete and (potentially) meaningless results; the effect of this situation is compounded when attempts are made to interpret the results. Although DRM solutions cannot enable fault tolerance in MPI-based applications to the degree that resynchronizations and reconnections are made possible, such middleware can trap and propagate signals, thus affording a significantly improved degree of management during execution – all without the need for additional coding (beyond exception handlers) by the MPI application developer.

As described in this section on clustering via middleware, Platform Computing's Load Sharing Facility (LSF, *http://www.platform.com*) is the only provider of a comprehensive DRM solution. Components of the DRM solution are provided by Sun's GridEngine (*http://www.sun.com/software/gridware/*) and Veridian's Portable Batch System (PBSPro, *http://www.pbspro.com/*); an open source version of the Portable Batch System (PBS) also exists (*http://www.openpbs.org*). TurboLinux's EnFuzion (*http://www.turbolinux.com/products/enf/*) offers a parametric processing solution.

Hybrid Solutions

Clustering via distributed operating systems or middleware shouldn't be taken to imply strict exclusivity. In fact, examples exist in which hybrid approaches have been employed. In one such case, Platform's LSF leverages SGI Array Services (*http://www.sgi.com/software/array/*) and SGI's implementation of the MPI, respectively, for distributed-process-space management and vendor-MPI leverage in the case of parallel computing. With the tremendous interest in Linux clustering solutions from both the open source and commercial ISV, integrator, and system-vendor communities, it is expected that such hybrids will continue to appear.

Summary

In addition to their price-performance appeal, clustering solutions can rival the throughput capacity and capabilities of legacy mainframes, supercomputers, and high-processor-count SMPs. The recent creation of a Top 500 list dedicated to cluster computing (*http://clusters.top500.org/*) can be regarded as one indication of the viability of this approach to high-performance computing. Both distributed operating systems and middleware have proven capable of delivering Linux clustering solutions; hybrid solutions have also been identified as likely to be of increasing value and prevalence in the not-too-distant future. The choice of approach will ultimately need to take into consideration a multiplicity of factors – e.g., total cost of ownership, importance of the service (which distinguishes needs from the interests of the hobbyist to the data center), desired characteristics of the service, etc.

As numerous activities serve to enhance the capabilities of the Linux kernel, and powerful 64-bit processors like API NetWorks Alpha (*http://www.apinetworks.com*) and Intel's Itanium (*http://www.intel.com/itanium/*) are increasingly commodified, the possibilities for clustering solutions increase significantly.

Federating geographically distributed clusters to aggregate resources, or providing access to specialized resources remotely, has been brought into focus in recent times through the notion of the Grid.[5] Much like the ubiquitous, highly available electrical power grid, the global computing grid allows challenging problems in HPC to be addressed. Various academic research (e.g., the Globus Project, *http://www.globus.org*, and the Legion Project, *http://www.cs.virginia.edu/~legion/*) and commercial (e.g., Applied Meta, *http://www.appliedmeta.com/*, and Platform's LSF MultiCluster, *http://www.platform.com*) ventures are already realizing the Grid. Because meta-com-

In addition to their price-performance appeal, clustering solutions can rival the throughput capacity and capabilities of legacy mainframes, supercomputers, and high-processor-count SMPs

NOTES

1. The Center of Excellence in Space Data and Information Sciences (CESDIS) is a division of the University Space Research Association (USRA), located at the Goddard Space Flight Center in Greenbelt, Maryland. CESDIS is a NASA contractor, supported in part by the Earth and Space Sciences (ESS) project. The ESS project is a research project within the High Performance Computing and Communications (HPCC) program.

2. D.F. Savarese, T. Sterling, "Beowulf," in R. Buyya, ed., *High Performance Cluster Computing*, vol. 1, 1999, pp. 625–645.

3. D.H.M. Spector, *Building Linux Clusters* (O'Reilly & Associates, Sebastopol, CA, 2000).

4. K. Dowd, C.R. Severance, *High Performance Computing*, 2nd ed. (O'Reilly & Associates, Sebastopol, CA, 1998).

5. I. Foster, C. Kesselman, eds., *The Grid: Blueprint for a New Computing Infrastructure* (Morgan Kauffman Publishers, 1999).

puting necessitates increased collaboration between all stakeholders, standardization efforts such as the Global Grid Forum (*http://www.gridforum.org*) and the New Productivity Initiative (*http://www.newproductivity.org/*) are expected to be of increasing importance.

Note Added In Proof

Since this article was originally written, the author has been apprised of the following matters noteworthy of communication:

Although Scyld has recently released a significant update of its own Beowulf clustering solution (*http://www.scyld.com/page/products*), and in addition to MOSIX, newcomers SCore (*http://pdsw3.trc.rwcp.or.jp*) and CPLANT (*http://www.cs.sandia.gov/cplant*) also merit serious consideration for clustering via distributed operating systems.

The Two-Kernel Monte (*http://www.scyld.com/products/beowulf/software/monte.html*), or the use of diskless compute nodes, can serve to reduce the strong kernel interdependency noted in the case of clustering via distributed operating systems.

GridEngine is to join OpenPBS as an Open Source middleware contribution from Sun Microsystems, Inc.

The New Productivity Initiative has recently released for public comment a draft of its API for distributed resource management (*http://www.newproductivity.org./pdf/RefModel-V1.pdf*).

Under the auspices of the Department of Defense (DoD, United States), the High Performance Computing Modernization Project (HPCMP) has recently realized a significant production computing grid implementation (*http://www.platform.com/solutions/whitepapers*). Nine of the DoD HPCMP's twenty-one distributed compute facilities are involved in a Phase I implementation that is aimed broadly at improving the organization's overall use of compute capacity, and enhancing their ability for compute capability. This project utilizes a number of DRM technologies from Platform Computing Inc..

The joint agreement between Compaq Computer Corporation and Intel Corporation (*http://www.compaq.com/newsroom/pr/2001/pr2001062501.html*) promises to infuse the Itanium processor family with the proven HPC capabilities of the Alpha processor. This fusion of commodity and technology should eventually invigorate the possibilities for Linux clustering in HPC.

the bookworm

by Peter H. Salus

Peter H. Salus is a member of the ACM, the Early English Text Society, and the Trollope Society, and is a life member of the American Oriental Society. He is Editorial Director at Matrix.net. He owns neither a dog nor a cat.

<peter@matrix.net>

For about 16 months now dot-coms have been dropping all over the place. NASDAQ is down, though higher than it's been, and everyone is looking glum. Well, cheer up.

The market soared. Did we really think it would go straight up forever? Moreover, don't despair: look at those income projections again. By and large, it's the growth rate that has changed. This happens with mature markets. It happened with cars, with TV sets, with washing machines.

E-Commerce

We had a fitful, feverish period where e-commerce was concerned. But the vast majority of automobile manufacturers (Cord, Maxwell, Stanley, LaSalle, DeSoto, Nash, Kaiser, Hudson, Pierce-Arrow, etc., etc.) are no more, yet there are vastly more vehicles. Their failure did not bode the death of the industry. Similarly, I think there is a great future for e-commerce, just not for many of the start-ups.

All of this is a prelude to my comments on a valuable book. Neidorf and Neidorf have turned out a relatively small volume (under 300 pp.) which will reward the reader.

They supply a "tour" of merchandising and management and informed me on a large number of topics I'd not really considered before: inventory management, pricing and promotion, profitability, vendor relations, and even organizational structure. There's also a brief glossary, a bibliography [!], and references to several useful Web sites (including that of the American Bar Association).

A worthwhile and informative read.

Applied Mathematics

Welschenbach's book on "cryptography" isn't really that. In actuality it's a first-rate introduction to the mathematical bases of cryptography.

The meat of the book is divided into two parts: "Arithmetic and Number Theory in C" and "Arithmetic in C++ with the Class LINT." I read the early chapters ("Number Formats," "Fundamental Operations," "Modular Arithmetic," and "Modular Exponentiation") at one go, while waiting for a car dealership to get around to tough tasks like changing oil and rotating tires. It was a great way to spend several hours (though the guy sitting beside me kept asking what the equations and lines beginning "#define" meant).

It's not easy stuff. Applied math isn't really for the faint of heart. But exponentiation and its application to cryptography is important. Welschenbach's class "LINT" (for Large INTegers) contains the data structures and functions that he utilizes in his analysis.

Welschenbach then uses this in talking about RSA (chapter 16) and then moving on to Rijndael, "a successor to the data encryption standard."

David Kramer has done a splendid job in translating Welschenbach's German.

LEGO Mindstorms

LEGO Mindstorms is the toy of choice for those of us who read *Popular Mechanics* and *Popular Electronics*. It is also a useful tool in education. Erwin's volume is, quite honestly, the best I've seen.

Erwin was involved in developing ROBOLAB, so it's not at all surprising that his volume enables the reader to

BOOKS REVIEWED IN THIS COLUMN

E-MERCHANT
JOANNE NEIDORF & ROBIN NEIDORF
Boston: Addison-Wesley, 2001. Pp. 276 + CD-ROM. ISBN 0-201-72169-4.

CRYPTOGRAPHY IN C AND C++
MICHAEL WELSCHENBACH
Berkeley, CA: Apress, 2001. Pp. 432 + CD-ROM. ISBN 1-893115-95-X.

CREATIVE PROJECTS WITH LEGO MINDSTORMS
BENJAMIN ERWIN
Boston: Addison-Wesley, 2001. Pp. 303 + CD-ROM. ISBN 0-201-70895-7.

DNS AND BIND
PAUL ALBITZ & CRICKET LIU
4th ed. Sebastopol, CA: O'Reilly & Associates, 2001. Pp. 624.ISBN 0-596-00158-4.

construct several robots (e.g., Walking Dog) as well as kinetic sculptures and some quite sophisticated projects.

There's a brief foreword by Seymour Papert and a good list of references and further readings.

Revisiting BIND

Albitz and Liu has been the standard handbook on DNS and BIND for nearly a decade. It is not a book for the raw beginner. You need to know something about the Domain Name System prior to cracking it. But it's invaluable. And it keeps getting better. The 4th edition is bigger and yet more useful.

It would have been really useful to Microsoft back in January, when a number of Microsoft-related Web sites just vanished. These included microsoft.com, slate.com, expedia.com, hotmail.com, and msnbc.com.

Microsoft had a DNS problem, which was compounded by a DDoS attack. Apparently, all four of Microsoft's DNS servers share the same routers, which means that all of them are vulnerable to hardware glitches or a technician's error.

There are, of course, reliable DNS services available to nervous customers. Nominium, for example, claims that it has a variety of collections of DNS servers, each with at least two different hardware and OS platforms, and each connected to two different ISPs.

Microsoft's errors and foolishness are obvious. The problem that Microsoft experienced once again illustrated the fact that even if you are a technically competent organization, your business is at significant risk without a highly reliable DNS infrastructure.

Though the apparent lack of diversity in Microsoft's name servers is a major error, there is a more general problem which also affects networks using BIND.

Using FreeBSD and BIND on every name server may be just as bad as employing Windows 2000 and Microsoft DNS on each of them.

If you use identical servers and identical software, no matter how geographically dispersed, a software flaw will affect all your servers at the same time.

And here's the kicker: Men&Mice found that 38% of the dot-com domains surveyed had all their name servers on the same subnet. And 75% had one or more configuration errors (see *http://www.menandmice.com/dnsplace/healthsurvey.html*).

DNS, like most databases, suffers from information entropy. It takes a lot of energy to keep information correctly updated while it is being changed. Anyone who has been hostmaster for even a moderately sized ISP knows there is an amazing number of ways for people to err.

And it's important to realize that one can't assume that IP addresses that are numerically contiguous represent hosts that are topologically close on the network.

The most obvious solution – and one which would take care of many problems – is diversity. The notion behind diversity isn't that diversity is error-free but that the error, whatever it is, is unlikely to strike more than one hardware platform, OS, or application at a time.

So here you are. Try running some Intel boxes and some SPARC or MIPS boxes. Try using UltraDNS on half your DNS servers and BIND on the other half. Try running Solaris on one SPARC and Linux or NetBSD on another; try running Windows 2000 on one Intel box and Linux on another.

And don't be like Microsoft. We've got proverbs that tell us the right thing: don't put all your eggs in one basket; variety is the spice of life.

Finally, understand what you're doing. Albitz and Liu will help with that.

standards reports

Our standards report editor, David Blackwood, welcomes dialogue between this column and you, the readers. Please send your comments to <*dave@usenix.org*>

Austin Group Status Update – June 1, 2001

by Andrew Josey

Andrew Josey is the director, server platforms, for The Open Group in Reading, England, and the chair of the Austin Group. Inc.

a.josey@opengroup.org

The Austin Common Standards Revision Group (CSRG) is a joint technical working group established to consider the matter of a common revision of ISO/IEC 9945-1, ISO/IEC 9945-2, IEEE Std 1003.1, IEEE Std 1003.2, and the appropriate parts of the Single UNIX Specification.

The eighth meeting of the joint technical working group informally known as the Austin Group was held at The Open Group facility in Reading, England, on May 29 – June 1, 2001. The meeting had 12 attendees (three via teleconference).

It was reported that Draft 6 had passed its IEEE ballot, achieving 90% approval. At the ISO level, Draft 6 has been submitted as a second FCD ballot. The Open Group, having achieved consensus, plans to synchronize the final approval with that of the IEEE Standards Board (targeted for September 12, see below).

The objective of this meeting was to review the Draft 6 specifications. A three-and-a-half day plenary session was held. Over 350 change requests were processed.

The most complex issue was resolving the wording to allow profiling of this standard. This was tackled via several teleconferences with interested parties during the meeting. The other main issue that was raised was the omission of the sockatmark() function, which is referenced in section 2.10.12 of XSH (Socket Out of Band data), but not included in the draft. The review group decided to canvass the opinion of the balloters to see if the inclusion of sockatmark() would decrease consensus. If not then it is proposed to include the sockatmark() function in Draft 7, as per the consistency caveat in the project scope. The minutes and documents Austin/SD1, Austin/SD2 should be read for further details.

The final draft (Draft 7) will be made available around June 15th and will include the changes identified by the review of Draft 6. Due to the workings of the IEEE ballot process, this draft may be re-circulated twice more for 10-day review (to allow any new objections to be re-circulated to the IEEE ballot group). The purpose of the Draft 7 review is to ensure that all changes identified have been applied correctly.

The Draft 7 review is scheduled for the second half of June 2001 (exact date to be determined) and will be available for a 10-day review period, with a concurrent 10-day re-circulation IEEE ballot. There will be strict narrowing-down rules enforced for the comments on Draft 7. Discussions on Draft 7 will occur via teleconference. The final document will be submitted to the IEEE on July 29 for consideration at the September 12 IEEE Standards Board meeting, and the September 12 meeting of The Open Group Platform Board.

SAGE news

SAGE Certification Update

June 2001

by Lois Bennett

Chairperson, SAGE Certification Committee

<lois@deas.harvard.edu>

The SAGE Certification project is moving forward thanks to the hard work of many.

The Exam Development committee met in April at the HumRRO offices outside Washington, DC. Members of the Committee reviewed all the questions written up to that point. After an intense discussion of tasks and procedures, the Committee decided to establish regional teams (North East, Central and West Coast) to tackle the balance of the questions to be written. Portions of the material to be covered were assigned to each regional group for completion. The committee will meet again at Harvard University in August to complete the first draft of the item bank.

The Policy Committee has been convening monthly teleconferences. Topics that have been discussed recently include governance and leadership, project management, fundraising, and policies. Lois Bennett has agreed to chair the Policy committee. The Policy Committee will serve as an interim Governing Board until an actual Board can be recruited.

Over the next 6-9 months, the Policy Committee and staff will work with two consultants to accomplish the following activities:

- Development of the Policies and Procedures
- Identification and procurement of testing vendor
- Fund-raising
- Marketing/branding/public relations
- Identification and relationship building with training partners

One consultant, Michael Hamm, will be helping us develop policies and procedures. The other, Stacy Gildenston, will help us with several tasks, including finding a testing vendor, fund-raising, marketing, branding, and public relations.

2000 SAGE System Administrator Salary Survey Now Available

The results of the 2000 SAGE System Administrator Salary profile are now available on the web at *http://www.usenix.org/sage/jobs/salary_survey/*. Thanks to the more than 5,200 system administrators who participated in the survey, making this the most comprehensive survey yet. This year's survey results include more information about consultants as well as system administrators from outside the United States.

USENIX news

Words mean what we choose them to mean, but who is "we"?

by Daniel Geer

President, USENIX Board of Directors

<geer@usenix.org>

It will come as no news to anyone reading this that the Board of Directors chose to make the USENIX Association a plaintiff in Felten, et al., v. Record Industry Association of America, et al. By the time this article is in print, the outcome of that case is likely to be no news either. Newsiness is irrelevant; as has been said, the four verities of governance are that

1. Most important ideas are uninteresting.
2. Most interesting ideas are unimportant.
3. Not every problem has a good solution.
4. Every solution has side effects.

This applies here – joining the Felten case is more important than interesting, it is not a perfect solution to anything, and it is not without side effects. Real life is messy.

There is a difference between knowledge and information, between observable fact and works of creation. Even if you agree prima facie, we are left, as we always are, with what is the definition of each of those terms. Indeed, if experience is any guide, in every issue at law the game is over after the definitions page – the rest is just mechanics. That seems true in Felten v. RIAA as much as anywhere, and while it is up to counsel and the courts to argue the definitions of what is proprietary, what is privileged, what is opinion, what is fact, what is science, what words mean and to whom.

The same will be true in any issue of technology and law that we are likely to see, whether USENIX is plaintiff in another court case, whether as officers we speak on any issue, whether as members you have a chance to put your mark on matters of public or private policy. You see this collision of words and meaning in many fields – technology is not unique in that regard – but I suggest that the rate at which ideas from the technology sphere mutate from interesting to important tends to exaggerate the production of side effects and less than perfect solutions. As the second derivative of technologic innovation remains solidly positive, the pressure technology brings on public process grows more profound.

Most of us in technical fields use the word "politics" as a collective noun for the nonsensical, unavoidable interference of the uninformed and the word "marketing" as a force that can't tell important from interesting. Even if these definitions are true, so what? All of us have probably argued "privacy" with someone in the last year and, if so, have found that the meaning of that word better be well and jointly defined if the argument is to have any lasting value. This is just illustration, and a precursor.

As what technology gives becomes ever more central to modern life, regulation of it becomes ever more the focus of politics and marketing. This drives me nuts, but so what? In some ways, the Felten case is about a notably speedy conversion of a particular technology from merely novel to important enough to fight over in arenas having nothing to do with problem statements, design rigor, imple-

mentation methodologies, or version control. If the technology weren't compelling in some way, it wouldn't be worth fighting over.

What I know as Putt's Law is that "Technology is dominated by two kinds of people, those who understand what they do not manage and those who manage what they do not understand." As long as the rate of change is accelerating, the above can only remain true. So, and this is the point, USENIX as an association will have to wade into fights from time to time absolutely as a side effect of how important technology has become. This is a distraction to what we like to do and to what we do well, but we ignore such moments at our collective peril. You, as individual members, face the same issues every day and all of us here will have to become much more expert in policy and law than we'd ever want to be in proportion to our success in, by our technology, putting a dent in the universe.

A decade of PGP

by Peter H. Salus
USENIX Historian
<peter@matrix.net>

Well, I goofed.

I missed an important anniversary. June 5 was the 10th anniversary of the release of PGP 1.0.

So here I am, a few weeks late. My apologies. I'll let Phil Zimmerman tell the tale himself.

"It was on this day in 1991 that I sent the first release of PGP to a couple of my friends for uploading to the Internet. First, I sent it to Allan Hoeltje, who posted it to Peacenet, an ISP that specialized in grassroots political organizations, mainly in the peace movement. Peacenet was accessible to political activists all over the world. Then, I uploaded it to Kelly Goen, who proceeded to upload it to a Usenet newsgroup that specialized in distributing source code. At my request, he marked the Usenet posting as "US only." Kelly also uploaded it to many BBS systems around the country. I don't recall if the postings to the Internet began on June 5th or 6th.

"It may be surprising to some that back in 1991, I did not yet know enough about Usenet newsgroups to realize that a "US only" tag was merely an advisory tag that had little real effect on how Usenet propagated newsgroup postings. I thought it actually controlled how Usenet routed the posting. But back then, I had no clue how to post anything on a newsgroup, and didn't even have a clear idea what a newsgroup was.

"It was a hard road to get to the release of PGP. I missed five mortgage payments developing the software in the first half of 1991. To add to the stress, a week before PGP's first release, I discovered the existence of another email encryption standard called Privacy Enhanced Mail (PEM), which was backed by several big companies, as well as RSA Data Security. I didn't like PEM's design, for several reasons. PEM used 56-bit DES to encrypt messages, which I did not regard as strong cryptography. Also, PEM absolutely required every message to be signed, and revealed the signature outside the encryption envelope, so that the message did not have to be decrypted to reveal who signed it. Nonetheless, I was distressed to learn of the existence of PEM only one week before PGP's release. How could I be so out of touch to fail to notice something as important as PEM? I guess I just had my head down too long,

writing code. I fully expected PEM to crush PGP, and even briefly considered not releasing PGP, since it might be futile in the face of PEM and its powerful backers. But I decided to press ahead, since I had come this far already, and besides, I knew that my design was better aligned with protecting the privacy of users.

"After releasing PGP, I immediately diverted my attention back to consulting work, to try to get caught up on my mortgage payments. I thought I could just release PGP 1.0 for MSDOS, and leave it alone for awhile, and let people play with it. I thought I could get back to it later, at my leisure. Little did I realize what a feeding frenzy PGP would set off. Apparently, there was a lot of pent-up demand for a tool like this. Volunteers from around the world were clamoring to help me port it to other platforms, add enhancements, and generally promote it. I did have to go back to work on paying gigs, but PGP continued to demand my time, pulled along by public enthusiasm.

"I assembled a team of volunteer engineers from around the world. They ported PGP to almost every platform (except for the Mac, which turned out to be harder). They translated PGP into foreign languages. And I started designing the PGP trust model, which I did not have time to finish in the first release. Fifteen months later, in September 1992, we released PGP 2.0, for MSDOS, several flavors of UNIX, Commodore Amiga, Atari, and maybe a few other platforms, and in about 10 foreign languages. PGP 2.0 had the now-famous PGP trust model, essentially in its present form.

"It was shortly after PGP 2.0's release that US Customs took an interest in the case. Little did they realize that they would help propel PGP's popularity, helping to ignite a controversy that would eventually lead to the demise of the US export restrictions on strong cryptography.

"Today, PGP remains just about the only way anyone encrypts their email. And now there are a dozen companies developing products that use the OpenPGP standard, all members of the OpenPGP Alliance, at *http://www.openpgp.org*.

"What a decade it has been."

Indeed, Phil. We're now at PGP 7.0.3; and we're all indebted to Phil and to the many volunteers.

The year 1991 was important: PGP, the Web, Linux. They're all 10 years old!

(BTW, my VAX 750 version of the 7th edition "User's Manual" is dated June 1981 – 20 years!)

2001 USENIX Awards

Every year USENIX acknowledges the contribution of exemplary members of the computing systems community through its Lifetime Achievement Award and Software Tools User Group Award. The USENIX Lifetime Achievement Award recognizes and celebrates singular contributions to the UNIX community in both intellectual achievement and service that are not recognized in any other forum. The STUG award recognizes significant contributions to the general community, which reflect the spirit and character of those who came together to form the Software Tools User Group (STUG).

This year we are pleased to announce the following recipients of these two awards:

USENIX Lifetime Achievement Award

The 2001 USENIX Lifetime Achievement Award recipient is the GNU Project and all its contributors, for the ubiquity, breadth and quality of its freely available redistributable and modifiable software, which has enabled a generation of research and commercial development.

GNU Project software tools have changed the way the computer world

operates. Today it is difficult to imagine how most of us could do systems work without tools that originated with or derived from GNU code. Much wide-ranging research is based on GNU tools. And in the computing world at large, millions of us have benefited and continue to benefit from the hard work and insight of the GNU Project.

Four aspects of the GNU Project merit special attention:

1. Its scope: GNU took on the whole enchilada, from the kernel to games, from the compiler to the libraries, from windowing systems to security authentication systems.

Some of the people who contributed to the GNU Project

2. Its quality: Bucking the current trend, where it is routine for software to fail, GNU software works.
3. The GNU community: From the start, the GNU Project has been a collaborative effort. Hundreds, and eventually thousands, of contributors, work together in the best hacker tradition.
4. Its provision for access to source: Although somewhat controversial, the GNU Public License and its variants solve a critical problem: how does a programmer ensure that everyone can access and modify his or her code?

The cumulative effect of the GNU Project has been revolutionary, permeating our technical lives at an ever-increasing pace since the project began in 1984. The USENIX Association's Lifetime Achievement Award for 2001 goes to the GNU Project in recognition of its achievements.

For more information about the awards, please visit:
http://www.usenix.org/directory/awards.html

The Software Tools User Group Award

The 2001 Software Tools Award (STUG) recipients are those who contributed to the development of Kerberos, a security system that set the standard for authentication and key management in distributed systems.

Kerberos is based on the revolutionary Needham and Schroeder protocol of 1978. It is a prime example of how to turn a theoretical result into a useful system. The need for authenticating users and services in a distributed environment is critical, and Kerberos provides a solution that is secure, relatively simple to administer, and scalable. Because of this, Kerberos has been implemented as part of the Distributed Computer Environment (DCE), the Andrew File Systems (AFS), and is also part of Windows 2000. No single security system has had as much impact on the way security is managed in distributed networks as Kerberos.

Ted T'so accepting the STUG Award on behalf of the Kerberos developers

For more information about the awards, please visit
http://www.usenix.org/directory/stug.html

US Team Selected For International Computing Olympiad

by Don Piele
USACO Director
piele@cs.uwp.edu

Four of the top young computer programmers in the United States have earned places on the USA Computing Olympiad (USACO) IOI team. The four – Reid Barton of Arlington, Mass.; Tom Widland of Albuquerque, NM; Vladimir Novakovski of Springfield, MA; and Steve Sivek, of Burke, VA – were selected from a field of fifteen candidates during a recently completed training camp at the University of Wisconsin-Parkside.

The team will now represent the US in the International Olympiad in Informatics. The competition will be held in Tampere, Finland, July 14 to 21, 2001.

Barton, a home-schooled high school senior and returning team member, won a gold medal at last year's International Olympiad in Beijing, China. He is joined by Widland, a senior at Albuquerque Academy, and juniors Novakovski and Sivek who attend Thomas Jefferson High School for Science and Technology in Alexandria, VA.

USACO team members were among 15 high school students invited to the training camp. The invitation was made on the strength of their scores on three Internet programming competitions and the US Open. More than 300 students across the country competed for the right to attend the camp.

USENIX sponsors the USACO. Find out more about the IOI team at *http://www.usaco.org*.

USENIX Association Financial Report 2000

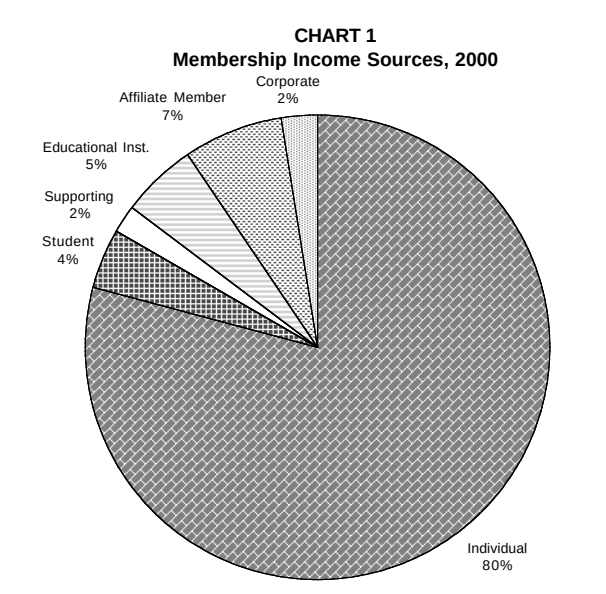
The following information is provided as an annual report of the USENIX Association, and represents the Association's statement of revenue and expenses for the year. Accompanying the statements are several charts that illustrate where your membership dues go. The Association's complete financial statements for the fiscal year ended December 31, 2000 are available on request from the USENIX Association.

The USENIX Association completed fiscal year 2000 with a net operating surplus of $205,031.

Membership for both USENIX and SAGE tops 10,000 members. Sixty-five percent of individual members are also members of SAGE.

Member Dues

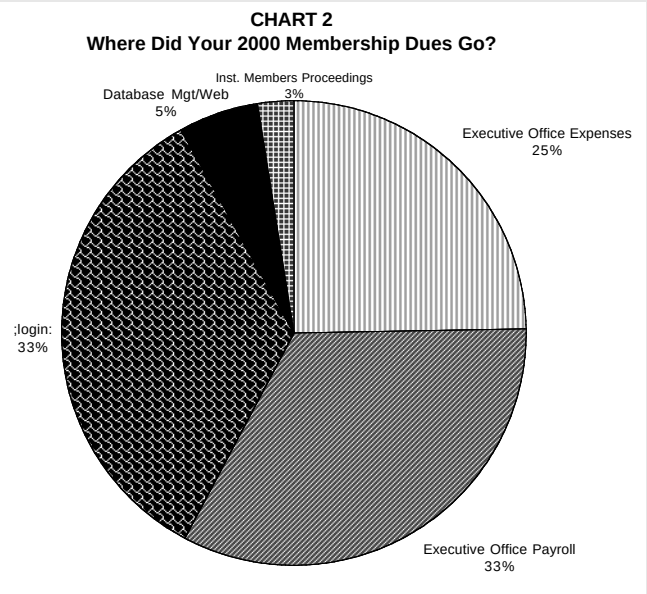
Chart 1 shows the total membership dues income (almost $950,000 in 2000) divided by type of membership. Chart 2 presents how those dues were spent. (Note that income from our conferences cover all costs of the conference department and staff, exhibition, and marketing.) Chart 3 shows how the USENIX administrative expenses were allocated. (The "other" category covers such items as taxes, licenses, bank charges and miscellaneous expenses.) Chart 4 indicates where most of the money allocated to Good Works/special projects and standards activities were spent ($977,000) in 2000. (See the USENIX Web site at *http://www.usenix.org/about/goodworks.html* for a description of our Good Works program. These funds come from the income generated by the USENIX conferences and interest income from the Association's reserve fund.)



Charts 5 and 6 deal with SAGE income (around $306,000 in 2000) and direct expenses (almost $250,000). Allocated expenses (staff and overhead) are not reflected in the direct expenses chart.

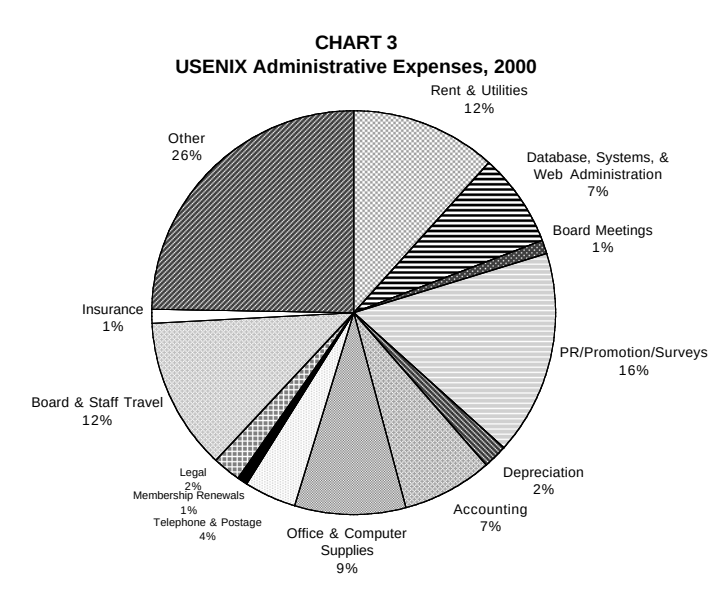
Members Services

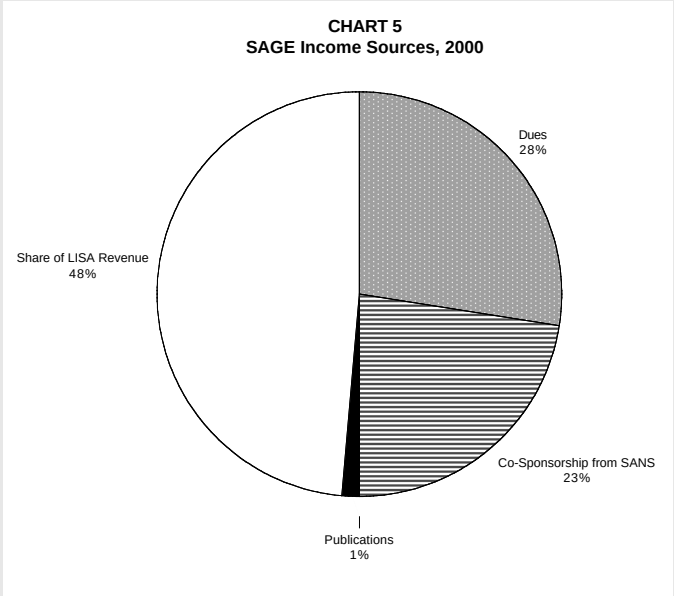
In 2000, member dues increased to $95 for an individual membership. Affiliate membership dues increased to $90. (All other membership dues categories, including SAGE, remained unchanged.) Eight issues of *;login:* were published. For the first time, online issues of *;login:* over a year old were made freely available to everyone. In addition, all standards reports, USENIX and SAGE news, conference reports, book reviews, and the "Using Java" columns were made available to everyone. Members were given access to feature articles less than a year old.

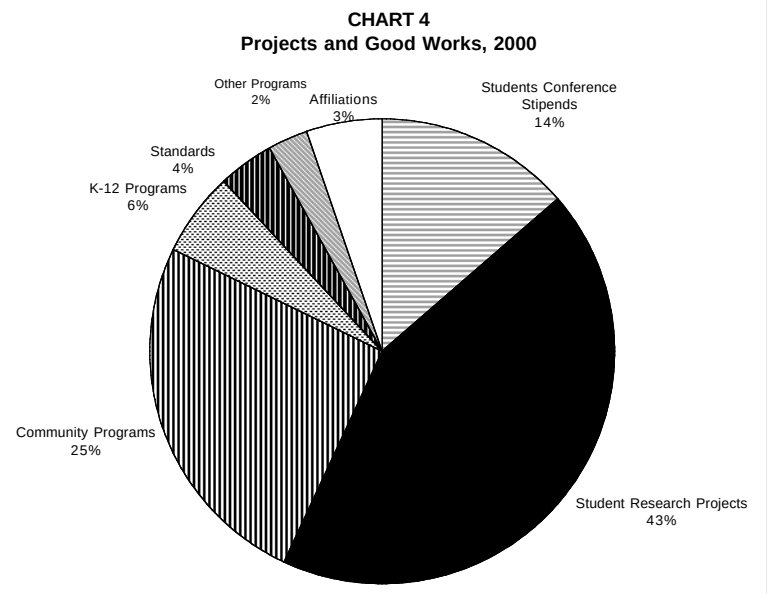


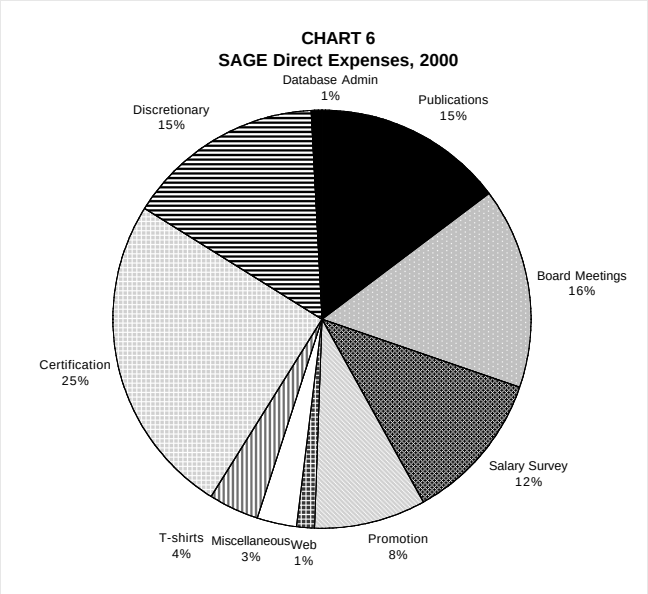
Conferences

In 2000, USENIX hosted four major conferences (USENIX Annual Technical, LISA, Security, and the Annual Linux Showcase) and four smaller symposia (OSDI/WIESS, Windows Systems, LISA-NT, and Tcl/Tk). Over 6500 people attended these events. Conference fees increased in 2000 to $435 for a three-day conference and $410 for a two-day symposium. Tutorial fees increased by $50 per day. Tutorials continued to be popular, and provide a significant proportion of revenue for the Association (around 43% in 2000). Membership as well as new and/or smaller conferences (e.g., ALS, OSDI, Tcl/Tk, NT) operate at a loss . Revenues exceeded expenses for three of the conferences including the Annual Technical, LISA, and Security. Conference proceedings of prior years were made freely available to everyone. Members were also given access to papers from conferences less than a year old.









Projects and Good Works

The Association's healthy year-end budget was supported by strong returns on investments, which netted $295,474 for the Good Works program. USENIX allocated over $1,100,000 for its Good Works program, and spent nearly all of it in 2000. These funds are used to provide stipends to students to attend USENIX conferences, scholarships, support of student research, promote outreach to students on campuses, as well as several innovative, computing-related projects. The student stipend program offers travel grants to enable full-time students to attend USENIX conferences and symposia. Over 360 institutions have been represented in the USENIX Student Stipend Program. To date, 113 schools have designated outreach representatives. The Student Program provides funding for scholarships and student research projects. In 2000, USENIX, in conjunction with Stichting NLnet of The Netherlands, initiated an international research exchange program for computer software-related networking technologies, called ReX. For more information about Student Programs, see: *http://www.usenix.org/students/students.html.*

STATEMENTS OF CASH FLOWS
For the Years Ended December 31, 2000 and 1999

	2000	1999
CASH FLOWS FROM OPERATING ACTIVITIES		
Increase in net assets	$ 88,838	$ 2,257,807
Adjustments to reconcile increase in net assets to net cash provided by/(used for) operating activities:		
Depreciation	67,545	44,498
Realized & unrealized gains on investments	348,608	(2,011,441)
Decr/(Incr) in receivables	(249,427)	(54,841)
Decr/(Incr) in inventory	(1,605)	3,767
Decr/(Incr) in prepaid expense	(36,281)	54,564
Incr/(Decr) in accrued expenses	725,661	(96,864)
Incr/(Decr) in deferred revenue	39,350	-
Total adjustments	893,851	(2,060,317)
Net cash provided by operating activities	982,689	197,490
CASH FLOWS PROVIDED BY/(USED FOR) INVESTING ACTIVITIES:		
Receipt of restricted funds [Note 9]		
Purchase of investments	(6,490,624)	(9,658,833)
Sale of investments	5,812,861	9,563,269
Purchase of property & equipment	(199,911)	(74,236)
Net cash used for investing activities	(877,674)	(169,800)
Net change in cash & equivalents	105,015	27,690
Cash & equivalents, beginning of year	2,107,048	2,079,358
Cash & equivalents, end of year	$ 2,212,063	$ 2,107,048

STATEMENTS OF FINANCIAL POSITION
As of December 31, 2000 and 1999

ASSETS	2000	1999
Current Assets		
Cash & cash equivalents	$ 2,212,063	$ 2,107,048
Receivables	364,982	115,555
Prepaid expenses	94,123	57,841
Inventory	20,149	18,544
Total current assets	2,691,317	2,298,988
Investments at fair market value	8,084,438	7,755,283
Property and Equipment		
Office furniture and equipment	497,378	297,467
Less: accumulated depreciation	(209,984)	(142,439)
Net property and equipment	287,394	155,028
	$ 11,063,149	$ 10,209,299
LIABILITIES AND NET ASSETS		
Liabilities		
Accrued expenses	$ 860,225	$ 134,564
Deferred Revenue	39,350	-
Total liabilities	899,575	134,564
Net Assets		
Temporarily Restricted Assets	51,000	
Unrestricted Net Assets	10,112,574	10,074,735
Net Assets	10,163,574	10,074,735
	$ 11,063,149	$ 10,209,299

STATEMENTS OF ACTIVITIES
For the Years Ended December 31, 2000 and 1999

	2000	1999
REVENUES		
Conference revenue (Exhibits A & C)	$ 5,238,831	$ 4,102,871
Workshop revenue (Exhibits B & D)	520,353	928,746
Conference & workshop sponsorship (Exhibits A & B)	246,325	
Membership dues	947,846	746,402
SAGE membership dues & other income	255,294	270,143
SAGE Certification sponsorship	51,000	
Product sales	30,064	47,712
Total revenues	7,289,713	6,095,874
OPERATING EXPENSES		
Conference expenses-direct (Exhibits A & C)	2,745,040	1,954,395
Workshop expenses-direct (Exhibits B & D)	461,664	728,771
Personnel & related benefits	1,302,055	1,136,834
Other general & administrative	899,228	628,736
Membership; login:	337,923	319,472
SAGE direct expenses	248,576	185,607
Product expenses	45,613	66,360
Projects & Good Works	1,044,583	958,338
Total operating expenses	7,084,682	5,978,513
Net operating surplus/(deficit)	205,031	117,361
NON-OPERATING ACTIVITY		
Donations	50,000	-
Interest & dividend income	298,381	215,943
Gains & losses on marketable securities	(348,608)	2,011,441
Investment fees	(115,966)	(86,938)
Net investment income & non-operating expense	(116,193)	2,140,446
Increase in net assets	88,838	2,257,807
Net assets, beginning of year	10,074,735	7,816,928
Net assets, end of year	$ 10,163,573	$ 10,074,735

Announcement and Call for Papers **USENIX**

2002 USENIX Annual Technical Conference

http://www.usenix.org/events/usenix02

June 9–14, 2002 **Monterey Conference Center, Monterey, California**

Important Dates

FREENIX Refereed Track submission deadline: *November 12, 2001*
General Session Refereed Track submission deadline: *November 19, 2001*
Notification to authors: *January 22, 2002*
FREENIX Track papers due for final shepherding approval: *April 8, 2002*
Camera-ready papers due: *April 16, 2002*

Conference Organizers

Program Committee:
Chair: Carla Ellis, *Duke University*
Darrell Anderson, *Duke University*
Mary Baker, *Stanford University*
Frank Bellosa, *University of Erlangen-Nuernberg*
Greg Ganger, *Carnegie-Mellon University*
Mike Jones, *Microsoft Research*
Patrick McDaniel, *University of Michigan*
Jason Nieh, *Columbia University*
Vern Paxson, *ACIRI*
Christopher Small, *Sun Microsystems*
Mirjana Spasojevic, *HP*
Mike Spreitzer, *IBM*
Amin Vahdat, *Duke University*

Invited Talks Coordinator:
Matt Blaze, *AT&T Labs–Research*

FREENIX Program Committee:
Chair: Chris Demetriou, *Broadcom Corp.*
Chuck Cranor, *AT&T Labs–Research*
Jim McGinness, *Consultant*
Craig Metz, *Extreme Networks*
Toon Moene, *GNU Fortran Team*
Keith Packard, *XFree86 Core Team & SuSE, Inc.*
Niels Provos, *University of Michigan*
Ronald Joe Record, *Caldera Systems*
Robert Watson, *NAI Labs & The FreeBSD Project*
Erez Zadok, *SUNY at Stony Brook*

General Session Refereed Papers

The 2002 USENIX Technical Conference seeks original and innovative papers about the applications, architecture, implementation, and performance of modern computing systems. Some particularly interesting application topics are:

- Cluster computing
- Complexity management
- Distributed caching and replication
- Energy/Power management
- File systems and storage systems
- Interoperability of heterogeneous systems
- Mobile/Wireless computing
- Mobile code
- Networking and network services
- Multimedia
- Reliability and QoS
- Security and privacy
- Ubiquitous computing
- Usage studies
- Web technologies

As at all USENIX conferences, papers that analyze problem areas, draw important conclusions from practical experience, and make freely available the techniques and tools developed in the course of the work are especially welcome.

Cash prizes will be awarded to the best papers at the conference. Please see the Web site for examples of Best Papers from previous years.

FREENIX Refereed Track

FREENIX is a special track within the USENIX Annual Technical Conference. USENIX encourages the exchange of information and technologies between commercial UNIX products and the free software world as well as among the various free operating-system alternatives.

FREENIX is the showcase for the latest developments and interesting applications of freely-redistributable software. The FREENIX forum includes Apache, Darwin, FreeBSD, GNOME, GNU, KDE, Linux, NetBSD, OpenBSD, Perl, PHP, Python, Samba, Tcl/Tk and more. The FREENIX track attempts to cover the full range of software which is freely redistributable in source-code form and provides pointers to where the code can be found on the Internet.

Submissions to the FREENIX track should describe freely-redistributable software. FREENIX encourages submissions which describe mature work, and for which the authors are ready to fully describe the background, new ideas, experiments, and results of their work. The FREENIX track also seeks to gather reports on projects that are current and solidly under way, but may not yet be 100% finished. This differs from a Works-In-Progress session, which is really a poster session for ideas.

FREENIX is looking for papers about projects with a solid emphasis on nurturing the open source and freely-available software communities. The purpose for the FREENIX track is not as an archival reference for all available projects with freely-redistributable source code, but rather a place to let others know about the project on which you are working and to provide a forum from which to expand your user and developer base. Papers should advance the state of the art of freely-redistributable software or otherwise provide useful information to those faced with deploying, selling, or using free software in the field.

Areas of interest include, but are not limited to:

- Cross-platform source portability and binary compatibility
- Desktop metaphors
- Distributed and parallel systems
- Documentation
- File system design
- Graphical user interface tools
- Highly-available systems
- Highly-scalable and clustered systems
- How free software is being developed and managed today
- Interesting deployments of free software
- Large scale system management
- Network design and implementation
- Operating system design
- Print systems
- Quality Assurance
- Security
- Software development tools
- Storage systems
- System and user management tools
- Technical aspects of commercial use of free software

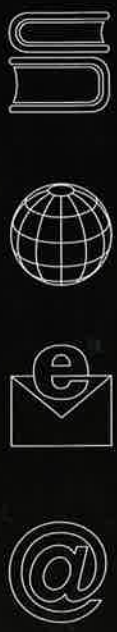
Interesting applications of freely-redistributable software include: wearable computers, video or audio processing, ubiquitous computing, studio graphics, robotics and automation, large-scale environments, high-speed networking, high-performance simulations, embedded systems, clustering, automation, and more.

Cash prizes will be awarded for the best paper and the best paper by a student.

How to Submit to the General Session and to the FREENIX Refereed Track

Please refer to our Web site at *http://www.usenix.org/events/usenix02/* for detailed information.

CONNECT WITH USENIX & SAGE



MEMBERSHIP, PUBLICATIONS, AND CONFERENCES

USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710
Phone: +1 510 528 8649
FAX: +1 510 548 5738
Email: <office@usenix.org>
<login@usenix.org>
<conference@usenix.org>

WEB SITES

<http://www.usenix.org>

<http://www.sage.org>

EMAIL

<login@usenix.org>

COMMENTS? SUGGESTIONS?

Send email to <ah@usenix.org>

CONTRIBUTIONS SOLICITED

You are encouraged to contribute articles, book reviews, photographs, cartoons, and announcements to *;login:*. Send them via email to *<login@usenix.org>* or through the postal system to the Association office.

The Association reserves the right to edit submitted material. Any reproduction of this magazine in part or in its entirety requires the permission of the Association and the author(s).

USENIX & SAGE

The Advanced Computing Systems Association &
The System Administrators Guild

;login:

USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710

POSTMASTER
Send address changes to *;login:*
2560 Ninth Street, Suite 215
Berkeley, CA 94710

PERIODICALS POSTAGE
PAID
AT BERKELEY, CALIFORNIA
AND ADDITIONAL OFFICES