;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

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.