;login:

THE MAGAZINE OF USENIX & SAGE
August 2003 • volume 28 • number 4

inside:

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

USENIX Upcoming Events

BSDCon 2003

SEPTEMBER 8–12, 2003
San Mateo, California, USA
http://www.usenix.org/events/bsdcon03/

5th IEEE Workshop on Mobile Computing Systems & Applications (WMCSA '03)

Sponsored by the IEEE Computer Society, in cooperation with ACM SIGMOBILE and USENIX

OCTOBER 9–10, 2003, Santa Cruz, CA, USA
http://wmcsa2003.stanford.edu

17th Systems Administration Conference (LISA '03)

Sponsored by USENIX and SAGE

OCTOBER 26–31, 2003 SAN DIEGO, CA, USA
http://www.usenix.org/events/lisa03
Final Papers due: August 4, 2003

Internet Measurement Conference 2003 (IMC '03)

Sponsored by ACM SIGCOMM and co-sponsored by USENIX

OCTOBER 27–29, 2003, MIAMI, FL, USA
http://www.icir.org/vern/imc-2003/

The 6th NordU/USENIX Conference (NordU2004)

Co-sponsored by EurOpen.SE and USENIX

JANUARY 28–FEBRUARY 1, 2004
COPENHAGEN, DENMARK
http://www.nordu.org/NordU2004/

2004 USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI '04)

Co-sponsored by USENIX, ACM SIGCOMM, and ACM SIGOPS

MARCH 29–31, 2004, SAN FRANCISCO, CA, USA
http://www.usenix.org/events/nsdi04
Paper submissions due: September 15, 2003

Third USENIX Conference on File and Storage Technologies (FAST '04)

MARCH 31–APRIL 2, 2004, SAN FRANCISCO, CA, USA
http://www.usenix.org/events/fast04
Paper submissions due: October 7, 2003

Third Virtual Machine Research and Technology Symposium (VM '04)

MAY 6–7, 2004, San Jose, CA, USA
http://www.usenix.org/events/vm04
Paper submissions due: October 13, 2003

USENIX Annual Technical Conference (USENIX '04)

JUNE 27–JULY 2, 2004 BOSTON, MA, USA

13th USENIX Security Symposium

AUGUST 9–13, 2004 SAN DIEGO, CA, USA

18th Systems Administration Conference (LISA '04)

NOVEMBER 14–19, 2004 ATLANTA, GA, USA

Sixth Symposium on Operating Systems Design and Implementation (OSDI '04)

DECEMBER 6-8, 2004, San Francisco, CA, USA
Web site: http://www.usenix.org/events/osdi04
Paper submissions due: May 14, 2004

For a complete list of all USENIX & USENIX co-sponsored events, see
http://www.usenix.org/events

contents

;login: Vol. 28, #4, August 2003

;login: is the official magazine of the USENIX Association and SAGE.

;login: (ISSN 1044-6397) is published bi-monthly by the USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, CA 94710.

$50 of each member's annual dues is for an annual subscription to *;login:*. Subscriptions for nonmembers are $110 per year.

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

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

Cover: MobiSys 2003: Mary Baker, Robert T. Morris, Program Chairs; Dan Siewiorek, General Chair.

motd

by Rob Kolstad

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

<kolstad@usenix.org>

Mental Nutrition

When we talk about nourishment, we often mention concepts like vitamins, carbohydrates, calory counting, food groups, and so on. But the physical spirit isn't the only part of the body that requires careful feeding.

What you feed your brain is also important. I awoke early on that September 11th when the World Trade Center went down. Just before 7:00 am, my living room laptop beeped (CNN alert, ya know) that a plane had run into one of the towers. Recalling news reports of the plane that ran into the Empire State Building in the 1930's, I switched on the TV. I continued to watch in fascination and horror for eight hours.

That was a mistake.

I fed my brain a full day of fear, uncertainty, and incitement. I spent background intellectual time for several days coming to grips with the experience and trying to get back to a regular pace.

Normally, I try to nourish my brain a bit more carefully. I've spent a little time figuring out what to feed and how it all works.

The input circuits appear to be connected to the standard five senses: sight, sound, taste, smell, and touch (we'll ignore the ESP senses and so on for this discussion). For information bandwidth, sight and sound are the big leaders. Some people think that sight is the big winner, since it obviously takes more bits to represent in data (just look at the number of bits on DVDs vs. CDs). However, if you've ever tried watching TV without the sound or listening to TV without looking at the picture, it often appears that sound has the more important data.

Years ago The Diagram Group published a book about the human body (*The Male Body: An Owner's Manual*) that showed two caricatures of the human body with senses exaggerated by data rate and also by impact on the psyche. Intriguingly, the low bit-rate "touch" sense was the big winner in the second diagram.

I feed facts and logical information to my brain from many sources: the Web, newspapers, trade magazines, hobby magazines, television, electronic mail, books, and the occasional movie.

While on my sabbatical a couple years ago, I found I was spending two hours every morning just to keep up with technical and trade magazines. (This number is borne out by results from the SAGE salary survey that show that a sizable fraction of administrators spend this much time, continually, just keeping up).

Recently, I've observed some other people feeding their brains. I hadn't realized that one can feed "junk information" to your brain just like one can feed "junk food" to your body. Empty information, useless databits, wasted time (time, of course, is the ultimate non-renewable resource) are consumed just as handily as potentially more stimulating works.

One of my longest-term friends has only recently begun to read fiction. He used to eschew it completely, saying ". . . but there's so much non-fiction to read!" He finally discovered that exploring things that might be or could be can lead to insights that simple deductive thinking might not reveal nearly so quickly.

These days I'm a real fan of those personal video recorders (like Tivo and Dishnet). They enable 30 minute television shows to be viewed in as little 18 minutes, with the added bonus of eliminating commercial announcements that often border on the inane. (Don't construe this comment as saying the shows are always that much better; no guarantees there, either.) When combined with the living room laptop multiplexed for web-surfing, light programming, or light e-mail reading, evenings can push lots of data (not necessarily information, unfortunately) into the brain. I am continually amazed that it's hard to keep up even when spending a few hours per night on such activities. The internet truly is a firehose of data.

I am also a big fan of "resting the brain" (i.e., vacations, "easy" activities, sleeping, etc.) but some things do seem like they don't nourish as well as others. Some people hang out in chatrooms whose conversation is little more than "hello" and "goodbye" to a seemingly endless procession of visitors. Maybe these are more like popcorn than they are like protein-rich alternatives.

I often try to find data that stimulates my thinking. This includes identification of issues important to me (technologies, a few world and political issues, a couple local issues) and sources for reliable information about them. The google news robot (*http://news.google.com*) is interesting because it presents news stories from dozens (more usually hundreds) of news sources. Reading the Bahrain newspaper's viewpoint of the Iraq war is sometimes more illuminating than that of our local newspaper's.

Every so often, I try to read opposing viewpoints to cherished beliefs. Occasionally, new data emerges that requires me to reassess various thoughts that I hold dear. Of course, this is somewhat frustrating, but it does seem like a "good thing to do".

The popular site Slashdot (*http://www.slashdot.com*) is interesting. It has a huge readership and a few hundred reader-commentators who write snippets about the various articles. Among other things, I have learned that I am apparently now officially an "old fart" (simply too old to participate in the "Linux movement" based on my chronological age). Apparently, I read the articles and commentary on slashdot dramatically differently than the majority of the contributors. Is it my years of experience and wisdom? Is it simply being completely out of touch with "the new generation of programmers?" Am I really just an old fart? I don't know. It's fascinating to watch in small doses, though.

Have you ever thought about what you feed your brain? Such contemplation makes for a very interesting week or two as you watch what it eats. It's then equally interesting to see how your brain processes the data. Good brain nutrition is as important as good body nutrition. I'm just sure that proper brain-food will lead to the same sort of happier, more productive lives that good body-food does.

Corrigendum

In the April issue (Vol. 28, No. 2) the reference to the CAIDA Sapphire worm should have included all the authors who contributed: David Moore, Vern Paxson, Stefan Savage, Colleen Shannon, Stuart Staniford, and Nicholas Weaver. The fascinating article in its entirety is available at *http://www.caida.org/outreach/papers/2003/sapphire/sapphire.html* .

;login:

EDITORIAL STAFF

EDITOR:
Rob Kolstad *kolstad@usenix.org*

CONTRIBUTING EDITOR:
Tina Darmohray *tmd@usenix.org*

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

COPY EDITOR:
Steve Gilmartin

TYPESETTER:
Festina Lente

MEMBERSHIP, PUBLICATIONS, AND CONFERENCES

USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710
Phone: 510 528 8649
FAX: 510 548 5738
Email: *office@usenix.org*
login@usenix.org
conference@usenix.org
WWW: *http://www.usenix.org*
http://www.sage.org

letters to the editor

To the Editor:

I enjoyed reading Timo Sivonen's excellent and highly informative article on IPv6 configuration in the April edition of *;login:*.

I have a few comments though, regarding the section on IPv6 and DNS.

After reading the article, one gets the impression that DNS resource records of type AAAA are deprecated in favor of A6, DNAME, and binary labels (bit-strings).

However, more recent RFCs indicate almost the opposite. Specifically, RFC 3363, written by Randy Bush, indicates a status change of RFCs 2874 and 2673 from "proposed standard" to "experimental."

Another change, proposed in RFC 3152, deprecates the use of the ip6.int domain, in favor of the newer ip6.arpa domain, but without use of binary labels.

If I read things right, there is IETF consensus on these matters. Before we all rush off to implement IPv6, we should probably make sure that we run in the same direction when it comes to the DNS part of the job. At least for now, it seems that RFC 3363 indicates the way to go.

An Internet-draft at *http://www.ietf.org/internet-drafts/draft-ietf-dnsop-ipv6-dns-issues-02.txt* may also be of interest. It discusses several IPv6 DNS transition issues. For example, should a name server listen on IPv4, IPv6, or both?

Finally, a note on IPv6 and managing reverse DNS data. For some time, we've been using tinydns as a content DNS server for some of our zones. The tinydns program is a part of Dan Bernstein's djbdns package. One nice feature is that tinydns automatically generates PTR records, eliminating the need to register your hosts twice in DNS namespace. A typical data line looks like this:

```
=host.example.com:10.0.0.5
```

The equals sign indicates a two-way mapping, so both an A and a PTR resource record are generated. (Of course, proper delegations must be defined.)

The current version of tinydns lacks IPv6 support, but patches do exist (*http://www.fefe.de/dns/*). This shows that name server software actually can relieve us from much of the messy work of maintaining reverse zone data.

Regards,

Mads E. Eilertsen
MadsE.Eilertsen@hist.no

Timo Sivonen Responds:

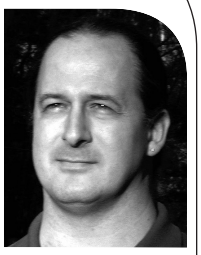
Mads E. Eilertsen had valid points on IPv6 and DNS in his letter. Since much of IPv6, and especially in relation to the DNS, is still under development, the current consensus resembles a moving target. I decided to cover both the RFC-1886 and RFC-2673/RFC-2874 approaches for IPv6 DNS, since both have their merits. The RFC-1886 approach is easier to implement, but the newer proposal would have had the flexibility to survive in the corporate world of mergers and acquisitions and constant network renumbering exercises. Yet RFC-3363 makes perfect sense, as it favours the conservative approach – the AAAA Resource Records and nibble format reverse records – in order not to break anything.

Finally, the most reliable way to find out where to go with IPv6 DNS probably is to start with the AAAA RRs and decide in the next 6–12 months whether the A6, DNAME, and RFC-2673/RFC-2874 are worth the effort.

opinion

On Choosing Usernames

by Adam Moskowitz



Adam Moskowitz is a system administrator, programmer, manager of system administrators, certified barbecue judge, and author of the recently published SAGE Short Topics booklet *Budgeting for SysAdmins*.

adamm@menlo.com

Every site faces the problem of how usernames (or "login names") are created or selected, by whom, and how conflicts are resolved. Every solution has its proponents, all with seemingly sound arguments – arguments that I think miss the real point. Based on over a dozen years as a user and another dozen as a system administrator (or manager of system administrators), I believe I finally understand the real problem and the best solution.

First, though, let me address two simple technical issues: conflicts and length. Schemes based solely on the user's name(s) will almost certainly result in conflicts. There are any number of ways to resolve such conflicts, but the point is that for almost every username scheme, there will be conflicts. With regard to length, there are still plenty of operating systems that limit usernames to eight characters or fewer. For a reasonably large segment of the population, this results in awkward truncations such as "amoskowi" or "cdagdigi" (for my co-worker with the last name "Dagdigian").

So back to the questions at hand: How should usernames be chosen, and by whom?

I think the best way is to let the users choose their own usernames; in the case of a conflict, let the users resolve it however they wish. Limitations, if any, should be based on lengths and character sets acceptable to the systems on which the username will be used. Businesses may choose to prohibit obscene usernames, but there's no technical reason why this must be done.

Why?

First and foremost, usernames are perceived by users as names. One can argue that they're not actually names, but if the user perceives them as names, then they are. In general, if a person says, "Hi, my name is Joe," it's considered rude to say, "Well, I'm going to insist on calling you Joseph." Imagine, then, when a user says "I prefer 'joef' as my username" and hears, "We're going to give you 'jsfritze' instead."

Second, it's the users who have to type their usernames every day, maybe several times a day. Not the system administrator, not the IT director – the users. An awkward username (like "amoskowi") can only detract from the user's overall experience. Maybe not by much, but if the user mistypes the name a few times, they're going to get frustrated.

Over the years, I've heard plenty of arguments against letting users chose their own usernames; most of the arguments boil down to one of these two: "It's not convenient for me [the system administrator]" or "That will cause problems with email."

On the first point, system administrators need to remember that their job is to help other people (users) get their jobs done (with due consideration to overall company benefit). If we can do that in a way that also makes our jobs easier, great — but user convenience trumps sysadmin convenience. On the technical side, nothing should be "encoded" in the usernames; that is, any additional information about the user should be stored somewhere else, in some other form. This could be as simple as how the UID is assigned or as complex as Jon Finke's use of Oracle databases.

The second argument I hear, that of email problems, is specious: Usernames are not email addresses! In most companies there may be an email address that happens to match a username, but there doesn't have to be. As long as the system has some way to

Many of us are set in our ways and can't see that what makes our job easier is often not what's best for our customers.

map email addresses to usernames (and this can happen fairly late in the delivery process), usernames never have to be known by anyone except the receiving user. In fact, there's probably even some (small?) security benefit to not propagating usernames beyond the institution's security perimeter.

Inside the company there should be some sort of "shared address book," whether an LDAP system or something like Microsoft Exchange, that users can consult for email addresses. Even this system doesn't need to return a username, just a valid email address. Outside the company there might be several email addresses for each person: f_lastname, firstname_lastname, f_oldlastname and f_newlastname, etc. Conflicts (Joe or John Smith) can be resolved in any number of ways, a common one being that messages to j_smith get a reply to the effect of "ambiguous email address; please use joe_smith or john_smith."

I will be surprised if what I'm proposing doesn't leave many of you spitting and sputtering and saying things like, "Well, *this* argument will change his mind." Before you send me email, please take a moment to consider whether your "killer reason" isn't really just a variation of "sysadmin convenience." Many of us are set in our ways and can't see that what makes our job easier is often not what's best for our customers. I certainly was, and it took working in a very special research environment with some very bright, articulate people to see that not only could I make my users happy, I could still do my job without any real loss of convenience. In fact, I also got a pat on the back from my boss after she got a note from a user saying, "Thanks for hiring Adam; he fixed my problem (changing my username) for me with no hassle at all, and I've been trying to get that done for over a year now." Not bad for having been on the job barely a month.

As a friend of mine says, "Change is required; growth is optional."

one up on LRU

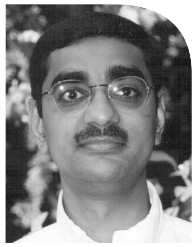
by Nimrod Megiddo

Nimrod Megiddo has published in the areas of optimization, algorithms and complexity, game theory, and learning, and has taught and lectured in several universities.

megiddo@almaden.ibm.com

and Dharmendra S. Modha

D. S. Modha has published on caching algorithms, information and coding theory,data mining, learning theory, signal processing, and data visualization. He holds 6 patents.



dmodha@almaden.ibm.com

Introduction

The concept of caching dates back (at least) to von Neumann's classic 1946 paper that laid the foundation for modern practical computing. Today, caching is used widely in storage systems, databases, Web servers, middleware, processors, file systems, disk drives, RAID controllers, operating systems, and in varied and numerous other applications.

Generically, a cache is a fast, usually small, memory in front of a presumably slower but larger auxiliary memory. For our purposes, both memories handle uniformly sized items called *pages*. We also assume demand paging. Host requests for pages are first directed to the cache for quick retrieval and, if the page is not in the cache, then to the auxiliary memory. If an uncached page is requested, one of the pages currently in the cache must be replaced (often requiring that the page be flushed back to the auxiliary memory, if it was written to by the host). A *replacement policy* determines which page is evicted. LRU is the most widely used replacement policy.

Until recently, attempts to outperform LRU in practice have not fared well because of overhead issues and the need to pre-tune various parameters. Adaptive Replacement Cache (ARC) is a new adaptive, self-tuning replacement policy with a high hit ratio and low overhead. It responds in real time to changing access patterns, continually balancing between the recency and frequency features of the workload, and demonstrates that adaptation eliminates the need for workload-specific pre-tuning. Like LRU, ARC can be easily implemented. Even better, its per-request running time is essentially independent of the cache size. Unlike LRU, ARC is "scan-tolerant" in that it allows one-time sequential requests to pass through without polluting the cache. ARC leads to substantial performance gains over LRU for a wide range of workloads and cache sizes.

ARC's Paradigm

Suppose that a cache can hold c pages. The ARC scheduler maintains a cache directory that contains 2c pages, c pages in the cache and c history pages. ARC's cache directory, referred to as DBL, maintains two lists: L1 and L2. The first list contains pages that have been seen only once recently, while L2 contains pages that have been seen at least twice recently. The replacement policy for managing DBL is: Replace the LRU page in L1 if it contains exactly c pages; otherwise, replace the LRU page in L2.

The ARC policy builds on DBL by carefully selecting c pages from the 2c pages in DBL. The basic idea is to divide L1 into a top T1 and bottom B1 and to divide L2 into top T2 and bottom B2. The pages in T1 are more recent than those in B1; likewise for T2 and B2. The algorithm includes a target size target_T1 for the T1 list. The replacement policy is simple: Replace the LRU page in T1, if T1 contains at least target_T1 pages; otherwise, replace the LRU page in T2.

The adaptation comes from the fact that the target size target_T1 is continuously varied in response to an observed workload. The adaptation rule is also simple: Increase target_T1, if a hit in the history B1 is observed; similarly, decrease target_T1, if a hit in the history B2 is observed.

LRU

Consider a very simple implementation of an LRU cache to motivate ARC. A typical implementation maintains a cache directory comprising cache directory blocks (CDB), often with a structure like this:

```
struct CDB {
    long                page_number;   /* page's ID number */
    struct cache_page   *pointer;      /* page's location in cache */
    int                 ARC_where;     /* not used for LRU */
    int                 dirty;         /* if 'dirty', write before replacing */
    struct CDB          *lrunext;      /* for doubly linked list */
    struct CDB          *lruprev;      /* for doubly linked list */
   };

struct  CDB *L;                        /* the LRU list */

long    LLength;                       /* length of list L */
```

LRU caches with c pages require c CDBs. The following code manages the LRU list and is invoked for each page request:

```
/* keep the cache descriptor list, L, in LRU order */

struct CDB *
LRU (long page_number, int dirty) {
    struct CDB  *temp;
    temp = locate(page_number);              /* search L for page #page_number */
    if (temp != NULL)                        /* page in cache? */
       remove_from_list(temp);               /* page in cache: remove now, reinsert later */
    else {                                   /* page is not in cache ... */
       if (LLength == c) {                   /* cache full? */
          temp = lru_remove(L);              /* cache full: remove the LRU page at end of list */
          if (temp->dirty)                   /* dirty -> page out changed pages */
             destage(temp);
       } else {                              /* cache not yet full */
          temp = get_new_CDB();              /* populate & bookkeep */
          temp->pointer = get_new_page();
          LLength++;
       }
       temp->page_number = page_number;      /* bookkeep */
       temp->dirty = dirty;                  /* bookkeep */
       fetch(page_number, temp->pointer, dirty);/* put new page in place */
    }
    mru_insert(temp, L);                     /* this page now goes to head of LRU queue */
    return temp;
}
```

We leave the simple routines locate, remove_from_list, mru_insert, lru_remove, destage, get_new_CDB, get_new_page, and fetch as an exercise. If the page is dirty, that is, a write request, then the fetch routine simply uses the changed page supplied by the host if the page is a read request, then the fetch routine reads the page from the auxiliary memory. Any existing LRU implementation already has these routines.

ARC

ARC requires 2*c CDBs. The extra directory entries maintain a history of certain recently evicted pages. The key new idea is the use of this history to guide an adaptation process. The cache directory consists of four disjoint doubly linked LRU lists along with their lengths:

```
struct CDB *T1, *B1, *T2, *B2;
long T1Length, T2Length, B1Length, B2Length;
```

Any given CDB will occupy a spot on one of the four lists. The field ARC_where will be set to 0, 1, 2, or 3, depending on the list in which it appears (T1, B1, T2, or B2, respectively).

The T1 and T2 lists describe c pages currently resident in the cache. The B1 and B2 lists contain c, a "history" of pages that were very recently evicted from the cache.

Furthermore, the T1 and B1 lists contain those pages that have been seen only once recently, while the T2 and B2 lists contain those pages that have been seen at least twice recently. The B1 list contains those pages evicted from T1, while B2 contains those pages that are evicted from T2.

The T1 and B1 lists capture "recency" information, while the T2 and B2 lists capture "frequency" information.

The algorithm adaptively – in a workload-specific fashion – balances between the recency and frequency lists to achieve a high hit ratio. It tries to maintain the number of pages in the T1 list to contain target_T1 pages. This parameter is adapted on virtually every request.

When the cache is full, the page to be evicted will be either the LRU page in T1 or the LRU page in T2.

This code demonstrates the page replacement procedure:

```
#define _T1_   0
#define _T2_   2
#define _B1_   1
#define _B2_   3

struct cache_page *
replace() {
     struct CDB* temp;
     if (T1Length >= max(1,target_T1)) {     /* T1's size exceeds target? */
                                             /* yes: T1 is too big */
        temp = lru_remove(T1);               /* grab LRU from T1 */
        mru_insert(temp, B1);                /* put it on B1 */
        temp->ARC_where = _B1_;              /* note that fact */
        T1Length—; B1Length++;               /* bookkeep */
     } else {
                                             /* no: T1 is not too big */
        temp = lru_remove(T2);               /* grab LRU page of T2 */
        mru_insert(temp, B2);                /* put it on B2 */
        temp->ARC_where = _B2_;               /* note that fact */
        T2Length—; B2Length++;               /* bookkeep */
     }
     if (temp->dirty) destage(temp);         /* if dirty, evict before overwrite */
     return temp->pointer;
}
```

The main algorithm comprises five cases which correspond to whether a page request is found in one of the four lists or in none of them. Only hits in T1 and T2 are actual cache hits. Hits in B1 and B2 are "phantom" hits that affect adaptation.

In particular, the cache parameter target_T1 is incremented for a hit in B1 and decremented for a hit in B2. This means that B1 hits favor recency while B2 hits favor frequency. The cumulative effect of the continual adaptation leads to an algorithm that adapts quickly to evolving workloads.

If a page is not in any of the four lists, then it is put at the MRU position in T1. From there it ultimately makes its way to the LRU position in T1 and eventually B1, unless requested once again prior to being evicted from B1, so it never enters T2 or B2. Hence, a long sequence of read-once requests passes through T1 and B1 without flushing out possibly important pages in T2. In this sense, ARC is "scan-tolerant." Arguably, when a scan begins, fewer hits occur in B1 compared to B2. Hence, by the effect of the adaptation of target_T1, the list T2 will grow at the expense of the list T1. This further accentuates the tolerance of ARC to scans.

If the list B1 produces a lot of hits, then ARC grows T1 to make room for what appears to be localized requests, and, hence, favors recency. If the list B2 produces a lot of hits, then ARC grows T2 to favor frequency. ARC continually balances between recency and frequency in a dynamic, real-time, and self-tuning fashion, making it very suitable for workloads with a priori unknown characteristics or workloads that fluctuate from recency to frequency. ARC requires no magic parameters that need to be manually tuned or reset.

Here's the straightforward code that implements this procedure:

```
ARC(long page_number, int dirty) {
    struct CDB *temp, *temp2;
    temp = locate(page_number);                                  /* find the requested page */
    if (temp != NULL) {                                          /* found in cache directory? */
        switch (temp->ARC_where) {                               /* yes, which list? */
            case _T1_:
                T1Length—; T2Length++;
                /* fall through */
            case _T2_:
                remove_from_list(temp);                          /* take off whichever list */
                mru_insert(temp, T2);                            /* seen twice recently, put on T2 */
                temp->ARC_where = _T2_;                          /* note that fact */
                if (dirty) temp->dirty = dirty;                  /* bookkeep dirty */
                break;

            case _B1_:
            case _B2_:
                if (temp->ARC_where == _B1_) {                   /* B1 hit: favor recency */
                    target_T1 = min(target_T1 + max(B2Length/B1Length, 1), c);
                                                                 /* adapt the target size */
                    B1Length—;                                   /* bookkeep */
                } else {                                         /* B2 hit: favor frequency */
                    target_T1 = max(target_T1 - max(B1Length/B2Length, 1), 0);
                                                                 /* adapt the target size */
                    B2Length—;                                   /* bookkeep*/
                }
                remove_from_list(temp);                          /* take off whichever list */
                temp->pointer = replace();                       /* find a place to put new page */
                temp->page_number = page_number;                 /* bookkeep */
                temp->dirty = dirty;                             /* bookkeep */
                mru_insert(temp, T2);                            /* seen twice recently, put on T2 */
                temp->ARC_where = _T2_;                          /* note that fact */
                fetch(page_number, temp->pointer, dirty);        /* load page into cache */
                break;
        }
    } else {                                                     /* page is not in cache directory */
        if (T1Length + B1Length == c) {                          /* B1 + T1 full? */
            if (T1Length < c) {                                  /* Still room in T1? */
                temp = lru_remove(B1);                           /* yes: take page off B1 */
                B1Length—;                                       /* bookkeep that */
                temp->pointer = replace();                       /* find new place to put page */
            } else {                                             /* no: B1 must be  empty */
                temp = lru_remove(T1);                           /* take page off T1 */
                if (temp->dirty) destage(temp);                  /* if dirty, evict before overwrite */
                T1Length—;                                       /* bookkeep that */
            }
        } else {                                                 /* B1 + T1 have less than c pages */
            if (T1Length + T2Length + B1Length + B2Length >= c) {/* cache full? */
                /* Yes, cache full: */
                if (T1Length + T2Length + B1Length + B2Length == 2*c) {
                    /* directory is full: */
                    B2Length—;                                   /* find and reuse B2's LRU */
                    temp = lru_remove(B2);
                } else                                           /* cache directory not full, easy case */
                    temp = get_new_CDB();
                temp->pointer = replace();                       /* new place for page */
            } else {                                             /* cache not full, easy case */
                temp = get_new_CDB();
                temp->pointer = get_new_page();
            }
        }
        mru_insert(temp, T1);                                    /* seen once recently, put on T1 */
        T1Length++;                                              /* bookkeep: */
        temp->ARC_where = _T1_;
        temp->page_number = page_number;
        temp->dirty = dirty;
        fetch(page_number, temp->pointer, dirty);                /* load page into cache */
    }
}
```

The Proof Is in the Pudding

Although ARC uses four lists, the total amount of movement between lists is comparable to LRU. The space overhead of ARC due to extra cache directory entries is only marginally higher – typically less than 1%. Hence, we say that ARC is low-overhead.

To assess ARC's performance, we conducted trace-driven simulations, results of which populate Table 1. ARC outperforms LRU for a wide range of real-life workloads – sometimes quite dramatically. For brevity, we have shown only one typical cache size for each workload. In fact, ARC outperforms LRU across the entire range of cache sizes for every workload in our test!

Traces P1–P14 were collected by using VTrace over several months from Windows NT workstations running real-life applications. ConCat was obtained by concatenating the traces P1–P14, while Merge(P) was obtained by merging them. DS1 is a seven-day trace taken from a database server at a major insurance company. The page size for all these (slightly older) traces was 512 bytes. We captured a trace of the SPC1 (Storage Performance Council) synthetic benchmark, which is designed to contain long sequential scans in addition to random accesses. The page size for this trace was 4KB. Finally, we considered three traces – S1, S2, and S3 – that were disk-read accesses initiated by a large commercial search engine in response to various Web search requests over several hours. The page size for these traces was also 4KB. The trace Merge(S) was obtained by merging the traces S1–S3 using timestamps on each of the requests.

Conclusion

ARC is an easily implemented, new, self-tuning, low-overhead, scan-tolerant cache replacement policy that seems to outperform LRU on a wide range of real-life workloads. We have outlined a simple implementation that may be adapted to a variety of applications. The reader interested in a formal presentation of ARC, a detailed literature review, and extensive simulation results can consult the full paper, "ARC: A Self-Tuning, Low Overhead Replacement Cache," in USENIX Conference on File and Storage Technologies (FAST '03), March 31–April 2, 2003, San Francisco, CA (*http://www.usenix.org/events/fast03/*).

Workload	Size (MB)	LRU (% hits)	ARC (% hits)
P1	16	16.55	28.26
P2	16	18.47	27.38
P3	16	3.57	17.12
P4	16	5.24	11.24
P5	16	6.73	14.27
P6	16	4.24	23.84
P7	16	3.45	13.77
P8	16	17.18	27.51
P9	16	8.28	19.73
P10	16	2.48	9.46
P11	16	20.92	26.48
P12	16	8.93	15.94
P13	16	7.83	16.60
P14	16	15.73	20.52
ConCat	16	14.38	21.67
Merge(P)	128	38.05	39.91
DS1	1024	11.65	22.52
SPC1	4096	9.19	20.00
S1	2048	23.71	33.43
S2	2048	25.91	40.68
S3	2048	25.26	40.44
Merge(S)	4096	27.62	40.44

Table 1. At-a-glance comparison of LRU and ARC for various workloads. It can be seen that ARC outperforms LRU, sometimes quite dramatically.

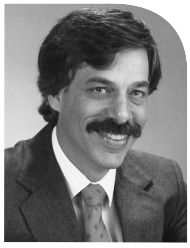
what's in a name?

by Tina Darmohray

Tina Darmohray, contributing editor of *;login:*, is a computer security and networking consultant. She was a founding member of SAGE. She is currently a Director of USENIX.

tmd@usenix.org

and Dan Appelman



Dan Appelman is a partner in the international law firm of Heller Ehrman, White & McAuliffe, LLP. He practices intellectual property and commercial law, primarily with technology clients, and is president-elect of the California Bar Association's Standing Committee on Cyberspace Law.

dan@hewm.com

[Editor's note: *In this article, Tina Darmohray relates an experience that many other domain name owners have been facing. As desirable domain names have been registered and locked up by their owners, the demand for good ones has exceeded the available supply. The result is twofold: a thriving market for the sale of domain name registrations and an upswing in threats of legal action to dispossess owners of their domain name rights. In this article, Tina describes the encounters she has had with others over her registered domain name, and Dan Appelman suggests some strategies to deal with prospective buyers and litigants.*]

Tina: My email address rolls off my tongue these days as I've had it longer than some of my kids have been alive. I registered my domain name many years ago when I decided to go into consulting full-time. Initially, I only used it for email, but as the Worldwide Web took hold, I put up a makeshift home page. I've never been much for Web design, so my page isn't very fancy; sometimes I don't host one at all. Still, people know to find me via my domain name, and I value the contacts that I receive through it.

About five years ago I was responding to a request for my email address when the recipient jotted it down, looked up and commented, "Three-letter domain name; that's got to be worth something."

"Beg your pardon?" I said.

"Your three-letter domain name. They aren't available any more. Yours has got to be worth something," he repeated.

Actually, I hadn't tried to register a new domain name since I registered mine originally, and it had never occurred to me that the shorter domain names were mostly taken and might have some inherent value. Maybe it was obvious to everyone else. I suppose it didn't dawn on me because I hadn't considered selling it. I used it for my business, and that was that.

Not long after the guy made the comment to me, I began receiving inquiries about selling my domain. I get three or four inquiries a year these days. One gentleman has been asking for years. I've learned that responding to such inquiries with "I'm not interested in selling it" doesn't always satisfy folks, so lately I've added a number that usually conveys my message, something like, "I'm not opposed to selling it, but it's worth six figures to me, so it's usually out of the price range folks are considering."

To my surprise, I actually got a response to my "six figures" email recently. She wrote back, "Give a figure and I'll decide." I was taken aback. Someone had called me on it. It forced me to give some serious thought to what my domain name was actually worth to me. To that end, I sent out mail to several of my consulting friends outlining my dilemma, asking them to help me fairly and accurately determine what my domain name was worth. But before I could receive a single response from my friends, I was being pummeled by the requestor with somewhat pointed emails asking me if I had a trademark on my name, telling me my phone number doesn't work, and informing me of ICANN's Uniform Domain Name Dispute Resolution Policy (UDRP) (see *http://www.icann.org/udrp*). Something just didn't seem right, so I turned to Google. Turns out, she was a "domain name attorney" for a large corporation.

With the help of my friends, I set a price and emailed it to her. Her response was aggressive and incessant. She implied that she'd only approached me to be nice, and that she intended to use other methods to take the domain away from me. What was of

particular concern about this was that several references on Google indicated that's the kind of thing she does. I decided I needed the help of an attorney.

I contacted Dan Appelman, USENIX's attorney and a personal friend. After describing my situation, Dan crafted a piece of email to her:

> "I'm well aware of the Uniform Domain Name Dispute Resolution Policy; and I know I have legitimate rights to the domain name and also to the service mark associated with my consulting business. I wasn't looking for an opportunity to sell my rights in the domain name to anyone; but since you asked, I would need to know who you are representing and what plans they would have for using it. Transferring the domain name registration would mean significant disruption to my consulting business, so it would have to be worth my while financially. I've already given you some idea what that would be. If your client is interested in discussing a purchase price at that level, I'd be happy to continue this correspondence. Otherwise, I'm sure your client can find another equally satisfactory domain name."

I haven't heard from her since.

Dan: Tina's experience is not unusual. I'm getting calls more frequently now from people who have either received offers to buy their domain names or been threatened with litigation if they don't agree to transfer their domain name registrations to someone else. Often, the callers are confused about what they should do; and this confusion is compounded by their uncertainty over what rights they have and how much their domain names are worth.

Domain names are freely transferable, like most other personal property. The market value of any given domain name is highly variable. One would have thought that the addition of a number of top-level domains, such as .biz and .info, would have made the value of second-level domain names such as Tina's go down significantly, but that doesn't seem to be the case.

The value of some famous brand names used as domain names is probably easily determined, but those cases are relatively few. On the other hand, the value of most domain names seems to have a large subjective component, and that makes valuation for purposes of sale somewhat difficult. When Tina asked me how much I thought her domain name was worth, all I could tell her was to try to test it in the marketplace. There are Web-based businesses that buy and sell domain names. You can even buy or sell domain names on eBay. But most individual owners set the price for their domain names by talking with their friends, reading about recent sales, and establishing a minimum value based on subjective factors.

You can even buy or sell domain names on eBay.

The other aspect of Tina's account is the confusion most people have about their rights in their domain names. Although domain names are personal property, the registration of domain names does not give rise to any protectable intellectual property rights. In fact, use of a domain name constitutes infringement if it causes, or could cause, confusion with a valid trademark. This is true even if the domain name has been accepted for registration by one of the domain name registries.

The solicitor who threatened Tina could have made her stop using her domain name if she could prove that Tina's use of that domain name was confusing the marketplace for some good or service that was branded with her or her client's trademark. Where domain names and trademarks collide, trademark rights will win because trademark rights are protected by law and domain name rights are not.

As an aside, in the United States trademark rights will trump domain name rights whether the trademark is registered or unregistered. All that is required is a convincing showing that the trademark has become associated in the minds of a relevant portion

Most domain name owners are relatively secure against challenges from third parties.

of the public with a particular commercial good or service. In most other countries, the law only protects the owners of registered trademarks.

In Tina's case, I did a search for uses of her domain name or its derivatives that included the records of the US Patent and Trademark Office (USPTO) and also a common-law search of telephone directories, trade catalogs, and the Web. On the basis of that search, I concluded that it was unlikely that anyone else was using Tina's domain name in the United States in conjunction with some product or service, and certainly not the person who was threatening Tina. Thus, I advised Tina that she was probably in no danger of being sued for trademark infringement or of having the rights to her domain name trumped by the other's trademark rights.

Domain name registrars have had to pay attention to the conflicts between domain names and trademarks in order to avoid getting drawn into lawsuits. Most registrars have adopted the UDRP, to which Tina's contact referred. At its heart, the UDRP recognizes that the proper place for resolving such disputes is in court and that the registrars should not be making legal determinations. But the UDRP does provide rules for dealing with complaints by trademark owners in the interim (i.e., prior to a judicial resolution of the dispute) and upon receipt of a court order. The UDRP tells domain name registrants how they can challenge requests by trademark owners to suspend, cancel, or transfer their domain name registrations.

The person threatening Tina was bluffing in order to get her to sell her domain name at a low price. She couldn't show us that Tina's domain name conflicted with any trademark she or her client was using in commerce. Consequently, she had no basis for suing Tina for trademark infringement. And because she couldn't demonstrate superior trademark rights, she also had no basis for asking Tina's domain name registrar to suspend, cancel, or transfer Tina's registration.

As a generalization, most domain name owners are relatively secure against challenges from third parties who seek to threaten them into selling their domain names inexpensively. Under the UDRP, absent a court order, domain name registrars will not suspend, cancel, or transfer domain names unless the challenger can show (1) that the domain name at issue is identical or confusingly similar to the challenger's trademark, (2) that the domain name registrant has no legitimate interests in the domain name at issue, and (3) that the domain name is being used in bad faith. That's a very high threshold, and most challengers cannot meet it. And trademark infringement lawsuits are expensive, so challengers are not likely to go that route unless they are convinced that they have a likelihood of success.

The USPTO site, *http://www.uspto.gov*, supplies free trademark searches and is the best site for searching for registered US marks. But since trademark owners have trademark rights in the United States even with respect to unregistered marks, the USPTO database does not tell the whole story. One would be wise to also do several Web searches and to commission a professional search from a company such as Thomson & Thomson (*http://www.thomson-thomson.com/*), which will also include state trademark registrations, searches of trademarks that appear in industry directories and other industry-specific publications, and domain name registrations. Professional searches range from $350 to $800, depending on how comprehensive and how quickly one needs the turnaround.

Tina was able to call this person's bluff because the legal research we did gave us confidence that her domain name was safe. However, many domain name registrants are not aware of their rights nor the steps they need to take to properly assess those rights in the face of a challenge. Undertaking a comprehensive trademark search prior to registering any domain name, and becoming familiar with the UDRP, will help.

production HPC reinvented

by Ian Lumb

Ian Lumb is a product solutions architect at Platform Computing, Inc. His interests include computing architectures, parallel computing, parametric processing, and grid computing. In a former life, Ian was a physical scientist working on problems in global geophysics at York University (Toronto, Canada).

ilumb@platform.com

Overview

Over the years, production High Performance Computing (HPC) was synonymous with scientific computing on "Big Iron" supercomputers. No longer dominated by just physical scientists and their Grand Challenge Equations, production HPC now embraces a variety of compute architectures. Though framed in the broader context of non-traditional HPC, attention here focuses on parallel computing via the Message Passing Interface (MPI). Problems cast as MPI applications are seen to have a parallel-computing bias that reaches back into the numerical methods that have been used and even to the originating science. Whereas MPI itself shows significant promise in addressing current computing challenges, in practice some serious shortcomings must be addressed in order for production HPC to be realized.

Workload-management system software closes the gap between MPI applications and their compute architectures, resulting in a solution for production HPC. A specific example of production HPC for the Linux operating environment shows that such solutions exist today. Moreover, the workload-management methodologies that apply at the cluster level have a natural affinity for extension to the Grid. Overall, organizations are able to better empower the pursuit of science and engineering during MPI application development, deployment, and use.

Five Steps to Scientific Insight

To motivate the applications and architectures discussion, consider a scientific-inquiry example from the physical sciences.[1] Once the problem under investigation has been determined, the first task is to determine the relevant physics, chemistry, etc. (Figure 1, Step 1). This consideration results in a mathematical description of the problem that needs to be solved (Figure 1, Step 2). On the positive side, the mathematical description typically exists, i.e., there is rarely a need to invent the mathematical description. In many cases, the required mathematical description can be formulated by combining existing descriptions. Although mathematics is the oldest and most deeply explored discipline, mathematical methods are often insufficient, except in idealized situations subject to simplifying assumptions, to solve the resulting equations. In mathematical terms, it is often difficult to near impossible to derive *analytic* solutions to many scientific equations. To make matters worse, in some cases it is difficult to prove that such solutions even exist. Such existence theorems serve as a cornerstone for the practice of mathematics. Thus science exposes serious mathematical challenges – in stark contrast to our childhood experiences with mathematics.

Given this challenging mathematical context, numerical methods are used to permit progress on otherwise unsolvable scientific problems (Figure 1, Step 3). Typically, this involves a discrete representation of the equation(s) in space and/or time, and performing calculations that trace out an evolution in space and/or time.[2] It's important to note that the underlying structure of the resulting set of equations influences the types of numerical methods that can be applied.[3]

Thus, numerical experiments are acts of modeling or simulation subject to a set of pre-specified constraints (Figure 1, Step 4). Problems in which time variations are key need to be seeded with *initial conditions*, whereas those with variations in space are

1. The analogous steps for informatics-heavy sciences such as the life sciences will be left for future consideration.

2. Symbolic algebraic manipulation (SAM) provides a notable analytic exception to this discrete approach. Maple ([1]) serves as a representative example of SAM technology. It is not uncommon to use SAM in conjunction with the discrete approach described here.

3. There is an extensive literature base on this topic. Implementations of equations involving a collection of methods are often organized into libraries.

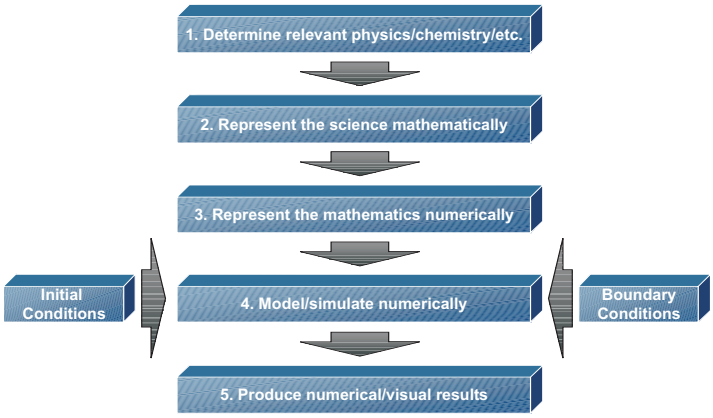


Figure 1. Five steps to scientific insight

subject to *boundary conditions*; it is not uncommon for problems to specify both kinds of constraints. The numerical model or simulation subject to various constraints can be regarded as a scientific application. Thus the solution of a scientific problem results in numerical output that may or may not be represented graphically (Figure 1, Step 5). One of four primary types (see "Applications and Architectures", below), this application is further regarded as scientific workload that needs to be managed as the calculations are carried out. Because the practices of science and engineering are actually undertaken as a process of discovery, Figure 1 should be regarded as a simplifying overview that does not depict the recursive nature of investigation.

It may appear that numerical methods are the cure-all for any scientific problem that cannot be solved by mathematical methods alone. Unfortunately, that is not the case. On their own, many of the equations of classical physics and chemistry push even the most powerful compute architectures to the limits of their capability. Irrespective of numerical methods and/or compute capability, these Grand Challenge Equations afford solutions based on simplifying assumptions, plus restrictions in space and/or time. Because these particularly thorny equations are critical in science and engineering, there is an ongoing demand to strive for progress. Examples of Grand Challenge problems are provided elsewhere ([2]).

Applications and Architectures

Science dictates mathematics and mathematics dictates numerics[4] (Figure 1). Thus a numerics bias exists in all applications of scientific origin. This predisposition motivates four types of applications (Figure 2) revealed by exploring process granularity. Granularity refers to the size of a computation that can be performed between communication or synchronization points ([3]). Thus, any point on the vertical axis of Figure 2 identifies a specific ratio of computation (increasing from bottom to top) to communication (increasing from top to bottom).[5] Task parallelism, increasing from left-to-right on the horizontal axis, refers to the degree of parallelism present in the application; "fine" through "coarse" are used as qualitative metrics, as shown.

Most scientific problems are implemented initially as *serial* applications (Figure 2, Quadrant II).[6] These problems require that each step of the scientific calculation be performed in sequence. Serial applications can be executed on compute architectures ranging from isolated desktops, servers, or supercomputers to compute farms. Compute farms are loosely coupled compute architectures in which system software is used to virtualize compute servers[7] into a single system environment (SSE).

Various factors – time-to-results, overall efficiency, etc. – combine to demand performance improvements beyond what can be achieved by "legacy" serial applications alone. For those applications whose focus is on data processing, it is natural to seek and exploit any parallelism in the data itself. Such *data parallel* applications (Figure 2, Quadrant I) are termed *embarrassingly parallel* since the resulting applications

4. "Numerics" is used here as a shorthand for numerical methods.

5. If the interest is computing, then communication is viewed as the 'overhead' required to achieve the computation. Similarly, computation might be regarded as the overhead required to facilitate certain communications.

6. The mathematical convention of numbering quadrants counter-clockwise from the upper-right-hand corner is used here.

7. Although high-density, rack-mounted single/dual processor servers have been used in compute farms, there is an intensifying trend toward the use of higher density blade servers in these configurations.

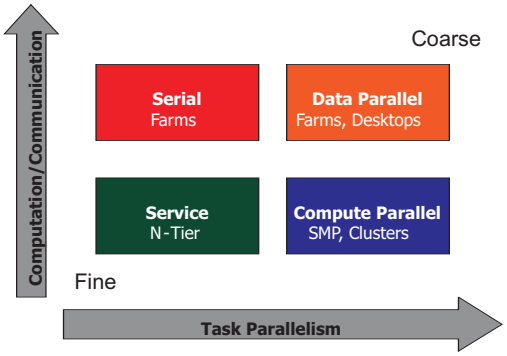


Figure 2. Applications and architectures

are able to exploit the inherent coarse granularity. The parallelism in data is leveraged by (Figure 3):

- Subdividing input data into multiple segments;
- Processing each data segment independently via the same executable; and
- Reassembling the individual results to produce the output data.

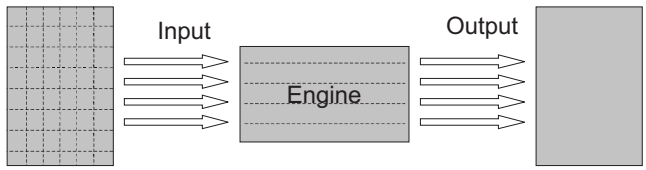


Figure 3. Data-driven parametric processing

This data-driven approach accounts for one of four classes of parametric processing (Figure 4). Although data-parallel applications are a solid match for isolated systems and compute farms, there is an increasing trend to harvest compute cycles from otherwise idle desktops. Regarded as opportunistic compute resources, desktops "pull" processing tasks from coordinating servers that they will execute as a background process – especially in place of a traditional screensaver. Desktops as compute elements gained initial popularity through the peer-to-peer (P2P) movement and more recently in the context of grid computing. Ray-tracing applications provide a classic demonstration of this processing architecture. Despite the added challenge of managing data, successful implementations of embarrassingly parallel applications exist in many industries – genome sequencing in the life sciences, Monte Carlo simulations in high energy physics, reservoir modeling in petroleum exploration, risk analysis in financial services, and so on. This approach is so appealing and powerful that it's often perceived to be of general utility. Unfortunately, this simply isn't the case – and this is especially true for the Grand Challenge Equations identified previously.

If it exists at all, parallelism in the Grand Challenge Equations can be exploited at the source-code level – e.g., by taking advantage of loop constructs in which each calculation is independent of others in the same loop. Parallelism in data is absent or of minor consequence. This code-level parallelism lends itself to *compute parallel* (Figure 2, Quadrant IV) applications. Compute parallel applications are further segmented on the basis of memory access – i.e., shared versus distributed memory. With minimal language extensions, and explicit code-level directives, OpenMP ([5]) and, more recently, Unified Parallel C (UPC, [6]) offer up parallel computing with shared-memory programming semantics. Symmetric multiprocessing (SMP) systems allow for shared-memory programming semantics via threads[8] through uniform (UMA) and nonuniform (NUMA) memory access architectures.

Parallel Virtual Machine (PVM, [7]) has given way to the Message Passing Interface (MPI, [8]) as the standard for parallel computing with distributed-memory programming semantics. Likened to the "assembly language" for parallel programming ([9]), MPI requires a significant investment at the source-code level.[9] In contrast to the use of threads in the shared-memory context, distributed processes are employed in the MPI case to achieve parallelism. MPI applications are typically implemented for tightly coupled compute clusters (see HPC Application Development Environment, below, for additional details). Although other factors (e.g., architecture access) need to be considered, numerics do influence the choice of parallel computing via shared versus distributed memory.[10] Both shared and distributed-memory parallel computing methodologies have been applied to scientific and engineering problems in a variety of industries – e.g., computational and combinatorial chemistry in the life sciences, computational fluid dynamics, crash simulations and structural analyses in industrial manufacturing, Grand Challenge problems in government organizations and educa-

8. In addition to the fork-and-exec creation of child processes, a parent process may also involve threads. Distinguishable by the operating system, threads can share or have their own memory allocations with respect to their parent process.

9. Recent advances allow serial applications to be automatically enabled for MPI ([10]). By identifying code regions suitable for parallelization (e.g., repetitive calculations) via a templating mechanism, code-level modifications are applied. This approach is being applied extensively in financial services, where numerical models change frequently.

10. Hybrid OpenMP-MPI applications allow scientists and engineers to simultaneously use threads and distributed processes when using a combination of SMP and clustered architectures.

tional institutions, and scenario modeling in financial services. MPI compute parallel applications, traditional HPC, will be the focus of our attention here.

Because MPI provides such a rich framework for computing in general, there are examples of MPI applications that communicate extensively while carrying out minimal processing (Figure 2, Quadrant III) – e.g., remote-to-direct-memory applications (RDMA), or certain classes of search algorithms. In addition, *service* applications whose focus is networking itself or Web services ([11]) themselves would also fall into this area. As before, MPI applications would require tightly coupled architectures; whereas networking applications can be applied in a variety of contexts, loosely coupled architectures can be used in the instantiation of Web services.

HPC Application Development Environment

Together with the "commoditization" of low-processor-count, high-density servers and the emergence of low-latency, high-bandwidth interconnect technologies, MPI has played a key role in the widespread adoption of tightly coupled compute clusters for distributed memory-parallel computing ([12]):

> MPI is available everywhere and widely used in environments ranging from small workstation networks to the very largest computers in the world, with thousands of processors. Every parallel computer vendor offers an MPI implementation, and multiple implementations are freely available as well, running on a wide variety of architectures. Applications large and small have been ported to MPI or written as MPI programs from the beginning, and MPI is taught in parallel programming courses worldwide.
> Applied through source-code-level modifications, MPI-specific directives are referenced against an MPI library at application link time. MPI libraries are architecture specific and may come from a variety of sources – e.g., a system vendor, an interconnect vendor, or via an open source contribution. In each case, the relevant library implements the MPI specification11 to some degree of compliance. This MPI library, in combination with the tools and utilities that support developers, collectively forms the application development environment for a particular platform (Figure 4).

11. Most MPI libraries fully implement version 1.x of the MPI specification, while many libraries are today supporting some subset of the version 2.x specification.

The situation described above might lead one to conclude that all of the requisites are present to smoothly enable MPI adoption. In practice, however, MPI has the following challenges:

- Resynchronization and reconnection were not even factored in at the specification level ([9]). There are no MPI implementations that account for this shortcoming. This is in striking contrast to PVM, whose implementation allows for this.
- Fault tolerance was not factored in, even at the specification level ([9]); again, there are no MPI implementations that account for this shortcoming. This can mean, for example, that an application can lose some of its processes, run to completion, and yield results of dubious validity.
- Hosts and numbers of processors need to be specified as static quantities, irrespective of actual usage conditions.
- Single point of control is absent. Although some recent MPI implementations offer a process daemon to launch MPI applications, there is little in the way of real application control.
- Multiple versions of MPI may exist on the same architecture. Applications need to carefully identify the relevant MPI libraries. The situation is more complex for MPI applications that span more than one execution architecture.

The upshot is clear: MPI places the responsibility for these shortcomings on the application developer and user. Because MPI applications are a challenge to control and audit, a better production HPC solution is required.

Production HPC Reinvented

There is a gap between the potential for distributed-memory parallel computing via MPI and what is actually achievable in practice. The use of system software allows this gap to be closed and the promise of MPI to be fully realized. In the process, the notion of production HPC is redefined. To start, consider a modified version of Figure 4 in which the newly added workload-management system software is shown in black on white (Figure 5).

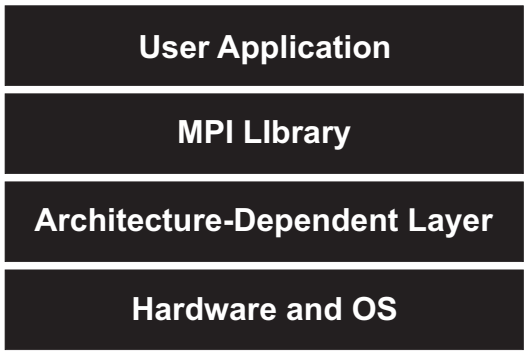


Figure 4. MPI application development environment

Figure 5 introduces the following three components to the MPI application development environment:

- **Core workload management services.** This system software component allows a heterogeneous collection of compute servers, each running its own instance of an operating system, to be virtualized into a compute cluster.[12] Sensory agents are used to maintain static and dynamic information in real time across the entire compute cluster. Primitives for process creation and process control across a network are also provided.
- **Parallel application management.** Challenges specific to the management of MPI parallel applications include the need to:
  - Maintain the communication connection map;
  - Monitor and forward control signals;
  - Receive requests to add, delete, start, and connect tasks;
  - Monitor resource usage while the user application is running;
  - Enforce task-level resource limits;
  - Collect resource usage information and exit status upon termination; and
  - Handle standard I/O.
- **Parallel scheduling services.** Workload management solutions typically employ a policy center to manage all resources – e.g., jobs, hosts, interconnects, users, and queues. Through the use of a scheduler, and subject to predefined policies, resource demands are mapped against the supply of resources in order to facilitate specific activities. Scheduling policies of particular relevance in parallel computing include advance reservation, backfill, preemption, and processor and/or memory reservation.

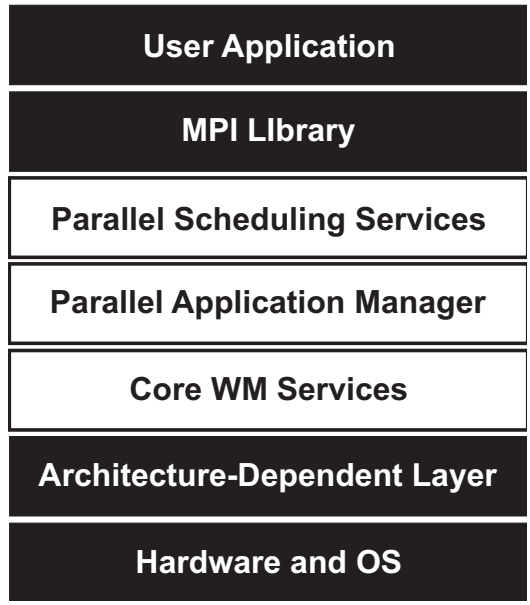


Figure 5. Enhanced application development environment via workload-management system software

The combined effects of these three layers of a workload-management infrastructure allow the shortcomings of MPI to be directly addressed:

- Absence of resynchronization and reconnection: Although the workload-management infrastructure (Figure 5) cannot enable resynchronization or reconnection, by introducing control across all of the processes involved in an MPI application, there is greatly improved visibility into synchronization and/or connection issues.
- Absence of fault tolerance: At the very least, the introduction of a workload-management infrastructure provides visibility into exceptional situations by trapping and propagating signals that may be issued while workload is executing. These signals can be acted upon to automatically re-queue workload that has thrown an undesirable exception. Even better, when integrated with the checkpoint/restart infrastructure of a workload manager, interrupted workload can continue to execute from the last successful checkpoint, often without user intervention.

12. This layered-services approach has been contrasted with Beowulf clustering (via a distributed process space) elsewhere ([13]).

- Absence of load balancing: The need to explicitly identify hosts and numbers of processors can be regarded as an absence of load balancing – i.e., the specification of resources that ignores current resource-usage conditions. Because workload-management infrastructures maintain dynamic load state across the entire compute infrastructure in real time, this shortcoming is completely eliminated.
- Absence of single point of control: A parallel application manager provides a single point of control. This ensures that all the distributed processes that collectively make up the MPI application are managed and accounted for. Again a key shortcoming becomes a core competence via workload-management system software.
- Multiple versions of MPI: Through configuration information, MPI applications are able to specify the relevant MPI library. A very flexible architecture also allows the workload-management system software to work in concert with the existing parallel application development and runtime environment.

Together with the existing parallel application development and runtime environment, workload-management system software allows the inherent shortcomings of MPI to be effectively eliminated. The cumulative effect is to practically reinvent HPC via MPI. This threefold reinvention is captured in Figure 6.

The process starts with MPI applications that are developed in-house or acquired from commercial providers.

On job submission, these applications are accompanied by a description of their runtime resource requirements – e.g., a processor-count range,[13] data-management directives (e.g., a file transfer), plus other environmental requirements.[14] The scheduling agent takes into account the pre-specified resource requirements, the unique characteristics of the compute architectures that it has available, and the policies that reflect organizational objectives. Because the scheduler creates a runtime environment for the workload, its task is one of dynamic provisioning. On dispatch, the scheduler has optimally matched the workload to an appropriate runtime architecture subject to established policies. The application executes in a fully managed environment. A comprehensive audit trail ensures that all activities are accounted for. In this way, there is a closed loop for production HPC, one that enhances the developer and user experience rather than encumbering it. A specific solution example is provided in the following section.

Production HPC for Linux

It has been suggested that production HPC can be reinvented through the use of an integrated development and runtime environment in which workload-management system software plays a key role. A complete example is considered here to further illustrate this reinvention. Consider the integrated production HPC solution stack shown in Figure 7. Although this example is based on the Linux operating environment, similar stacks for other operating environments can be crafted.

At the base of the production HPC solution stack for Linux are low-processor-count servers,[15] each running its own instance of the GNU/Linux operating system. Ethernet-based network interface cards (NICs) allow for standard TCP/IP-based services between these systems. Because TCP/IP over Ethernet necessitates assumptions regarding shared access and the occurrence of collisions, the result is a communications protocol that is latency heavy and therefore inefficient for message-passing in support of parallel computing. Hence, each system implements Myricon's GM message-passing

13. Experience dictates that every parallel application show acceptable performance characteristics over a range of processors. A typical criterion is that the performance remain close to linear as the number of processors increases. This is referred to as "linear speedup." Effective workload-management systems allow this processor count to be specified as a range at workload submission time. This serves to automate the load balancing situation and to enhance overall effectiveness of the scheduling services.

14. The need to bind processes to processors serves as one example of an execution environment requirement. In such cases, the workload-management infrastructure works in tandem with the operating system, interconnect manager, etc., to address the requirement.

15. Historically, Alpha-based processors were used because of their excellent floating-point-performance characteristics. With the advent of first-generation Itanium processor family CPUs, it is expected that 64-bit Intel Architecture (IA-64) will eventually dominate in this space. In the interim, fourth-generation IA-32 Pentium processor family CPUs hold the price/performance niche.

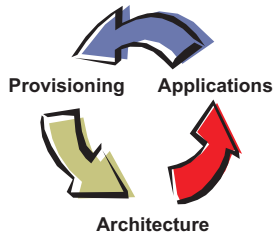


Figure 6. Production HPC reinvented

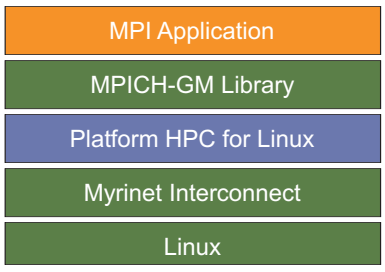


Figure 7. Production HPC for Linux

protocol across low-latency, high-bandwidth, multi-port Myricom Myrinet switches ([14]) used solely to support parallel computing via MPI and the GM driver. By identifying each available Myrinet port as a resource to the core workload-management services provided by Platform HPC for Linux ([15]), the provisioning component of this infrastructure is aware of the static and dynamic attributes of the architecture it can apply parallel scheduling policies against. Platform HPC for Linux also provides the control and audit primitives that allow parallel applications to be completely managed. Users' MPI applications need to be compiled and linked against the application development environment provided by Myricom. This ensures that the appropriate GM-protocol modifications to the widely adopted open source MPICH implementation of MPI are used. Through configuration information, the workload-management system software based on Platform HPC for Linux is made aware of the enhanced MPI runtime environment.

Portability was identified as a design goal for MPI. This objective has been carried through in MPI implementations such as MPICH. Despite this fact, heterogeneous parallel applications based on MPI must not only use the same implementation of MPI (e.g., MPICH) but also the same protocol implementation (e.g., GM). In other words, MPI is a multi-protocol API (Figure 8) in which each protocol implements its own message formats, exchange sequences, and so on.

Because they can be regarded as a cluster of clusters, it follows that computational grids might provide a suitable runtime environment for MPI applications. The fact that grids are concerned with resource aggregation across geographic (and other) domains further enhances the appeal. Fortunately, Platform HPC for Linux is consistent with Platform MultiCluster – system software that allows independent clusters each based on Platform LSF to be virtualized into an enterprise grid. This combination introduces the possibility for exciting new modes of infrastructural provisioning through various grid-centric scheduling policies – e.g., Grid Advance Reservation, Grid Fairshare, and Grid Resource Leasing.

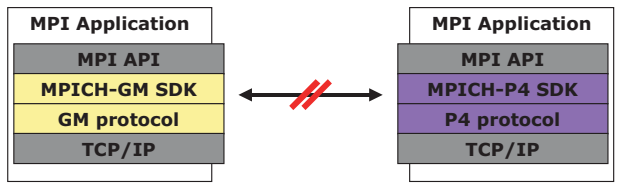


Figure 8. The multi-protocol nature of MPI (after [16])

Of these grid-centric policies, Grid Resource Leasing (GRL) is particularly novel and powerful. GRL allows sites to make available fractions of their resources for use by other sites participating in their enterprise grid. These collective resources can be occupied by MPI applications through requirement specifications. In this fashion, users can *co-schedule* MPI applications to span more than one geographic location. Even more impressive is the fact that this co-scheduling is automatically enabled at runtime, that is, existing MPI applications do not need to be re-linked with special libraries. By jointly leveraging the parallel application management capability already present in Platform HPC for Linux, in concert with this grid-level scheduling policy of Platform MultiCluster, MPI application users take advantage of their entire enterprise grid in a transparent and effective fashion. Platform MultiCluster and its grid-level scheduling policies are considered in detail elsewhere ([17]).

Summary

Production High Performance Computing (HPC) incorporates a variety of applications and compute architectures. Widespread use of the Message Passing Interface (MPI) is better enabled through the use of workload-management system software. This software allows MPI applications and the compute architectures on which they execute to be provisioned on demand. This critical link significantly reduces complex-

ity for the scientist or engineer, thus reducing time to results and ensuring overall organizational efficiency. A tightly coupled cluster based on the Linux operating environment was shown to be a particularly attractive and viable compute architecture. The incorporation of this environment into a compute grid was also shown to be a natural progression. Overall, organizations are able to better empower the pursuit of science and engineering during application development, deployment, and use.

Acknowledgments

Chris Smith, Brian MacDonald, and Bill McMillan are all acknowledged for their contributions to this article.

References

[1] Maple, *http://www.maplesoft.com*.

[2] The Grand Challenge Equations, *http://www.sdsc.edu/Publications/GCequations.*

[3] B. Wilkinson & M. Allen, *Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel Computers* (Upper Saddle River, NJ: Prentice-Hall, 1999).

[4] I. Lumb, B. Bryce, B. McMillan, and K. Priddy, "From the Desktop to the Grid: Parametric Processing in High-Performance Computing," Sun User Performance Group, Amsterdam, Netherlands, October 2001.

[5] OpenMP, *http://www.openmp.org*.

[6] Unified Parallel C, *http://upc.gwu.edu*.

[7] Parallel Virtual Machine, *http://www.csm.ornl.gov/pvm/pvm_home.html*.

[8] Message Passing Interface, *http://www.mpi-forum.org.*

[9] K. Dowd and C.R. Severance, *High Performance Computing*, 2d ed. (Sebastopol, CA: O'Reilly & Associates, 1998).

[10] Powerllel, *http://www.powerllel.com/powerllel/content/index.shtml*.

[11] Web services, *http://www.w3.org/2002/ws/*.

[12] W. Gropp, E. Lusk, and A. Skjellum, *Using MPI: Portable Parallel Programming with the Message-Passing Interface*, 2d ed. (Cambridge, MA: MIT Press, 1999).

[13] I. Lumb, "Linux Clustering for High-Performance Computing," *;login:*, vol. 26, no. 5, August 2001, pp. 79–84.

[14] Myricom Myrinet, *http://www.myricom.com/myrinet*.

[15] Platform Computing Corporation, "Using LSF with MPICH-GM," technical documentation, April 2002.

[16] MPICH-G2, *http://www.hpclab.niu.edu/mpi.*

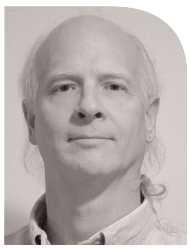
[17] Platform Computing Corporation, *Platform MultiCluster Guide*, technical documentation, June 2002.

musings

by Rik Farrow

Rik Farrow provides UNIX and Internet security consulting and training. He is the author of *UNIX System Security* and *System Administrator's Guide to System V*.

rik@spirit.com



A funny thing happened to me last May. I was asked a question about Microsoft Windows desktop security that actually caught my attention.

In a single week, two women contacted me wanting to know how to stop their ex-boyfriend from reading their email. I immediately thought of a trojan running a keystroke logger and suggested that they get, or update, their antiviral software. But then I decided to look further.

I didn't have very far to look. Just enter "keystroke loggers" into a Google search, and you will turn up thousands of entries, as well as sponsored ads for spyware, the more common name for keystroke loggers. Based on the advice of compadres in the world of security more familiar with Windows, I pointed both women to SpyCop, a spyware detector. But I didn't stop there.

As I continued to search, I found a site (*http://spywareinfo.com*) that listed 223 spyware products (this is actually an anti-spyware site). Another vendor stated that there were over 300 spyware programs available. I found a hardware dongle (Keyghost) as well as a number of spyware detectors.

Spyware, in the generic sense, is nothing new. Microsoft made headlines when it included a mechanism to scan hard drives for software and upload those listings with pre-release versions of Windows 95. Microsoft quickly removed that feature. But current versions of Windows Media Player contact Microsoft every time you play a CD or DVD, to collect a track listing for you. Unscrupulous programs could, in fact, log that data for later reference or analysis (a unique ID is registered for each copy of the WMP).

And spyware for X Window has been around even longer than Windows. You can search for and find versions of the xkey program, an X client that collects all keypress events in any X Window server it can connect to. Of course, you are protecting your X Window server (it appears that any reasonable UNIX distribution uses xauth, but X Window for Windows products often fail to do this). One weakness of xkey is that it reports all keypresses, without identifying the application reading those keys. X events do include a window ID, and it should be possible to actually match keys with applications with a bit of programming.

Note that someone who adds a line to a UNIX user's startup files can run xkey as that user and will have complete access. Protect your dot (startup) files!

But Microsoft makes things much easier for Windows programmers, one of the reasons for Microsoft's phenomenal success. Just as personal firewall products for Windows can identify which application has requested a network connection, keystroke loggers for Windows can identify the application that will receive any keystrokes. That made it possible for the Badtrans.B trojan to focus on the characters presented to particular applications, such as Telnet and connections to RAS (Remote Access Servers). A similar keystroke logger helped someone invade Microsoft's networks in the summer of 2000 by capturing a username, password, and RAS server address.

Anti-spyware products often work like antiviral products, looking for signatures. Another sign of a spyware installation is changes to certain registry keys and startup files. I have been told that there are at least 56 different methods that a trojan writer can use so that the trojan gets restarted with every reboot or login. The most common ways include modifications to the various Run keys in the registry, to startup files like WIN.INI, and to users' startup folders. You can actually download a free program

If only George Orwell (*1984*) had known about PCs, it wouldn't have been the TV watching its viewers.

from Sysinternals (*http://www.sysinternals.com/ntw2k/source/misc.shtml#autoruns*) that will report on the list of all software started with each reboot, including services. Installing a "special" service, such as slanret, has become a popular way of rootkitting Windows 2000 systems, so you do want to pay attention to the .sys files included when your system boots. Microsoft wants you to stick to signed drivers, which should help you avoid being rootkitted.

And if you are wondering about how someone installs a keystroke logger, a common way to do it is through physical access. But Windows viruses and trojans will also do this. Note that the virus Bugbear.B, which was making the rounds in early June, uses as one of its vectors the MIME-auto-exec bug that Microsoft announced a patch for in March 2001 (*http://www.microsoft.com/technet/security/bulletin/MS01-020.asp*). Bugbear.B also uses a more recent, just patched, vulnerability in Internet Explorer to execute its code. If you use IE, check for patches often.

As I learned just how common spyware has become, I wondered about the legality of it. I heard from a police dispatcher, for example, about a father who had installed spyware to monitor his daughter's Internet usage. So I contacted an acquaintance within the US Department of Justice who deals in computer crime and asked him if he could help me. He agreed, but only as a background source.

Using spyware without authorization is a federal crime, a felony in the US. But that word, "authorization," is the slippery one. Your employer is authorized to sniff your keystrokes or network connections if you have agreed to a policy that includes monitoring or the systems you use have logon banners announcing that use amounts to an agreement to be monitored.

Parents have been given a free ride by the courts when they are acting as responsible custodians for their children. But an ex-boyfriend, girlfriend, or husband who installs software that collects electronic communications would be considered to be committing the same offense as tapping a telephone (the same laws apply). Same thing when someone, without permission, installs spyware in your computer to collect keystrokes. My Justice Department contact told me that if that information were used to break into a computer, he would be more likely to use the break-in during a prosecution, because it would be easier for a jury to comprehend than keystroke logging.

But keystroke loggers are not the only applications spying from within Windows desktops these days. Many "free" services include clauses within the EULAs that permit them to install software that monitors Web usage, music tracks downloaded, and so on, all the better to target the user for advertising and spam. And this is an authorized use, because the end user agreed to the EULA. Kazaa users beware!

In these scary times, it turns out that the government is much too busy watching for terrorists – that is, anyone from certain countries who is visiting on a visa or praying in a certain church – to spy on most citizens. It is actually much more difficult for a government agency to go through the red tape necessary to wiretap you (even after PATRIOT Act v1) than it is for private industry to include a request to bug your desktop in their EULA. If only George Orwell (*1984*) had known about PCs, it wouldn't have been the TV watching its viewers.

ISPadmin

Service Provider Book Reviews II

In this edition of ISPadmin, I review two new books that are of interest to most in the service provider business and many in the field of information technology. (No, I am not going to turn this column into a book review column; I promise these are my last book reviews for a while!)

In the interests of full disclosure, I must state that I was paid to evaluate the manuscript of *LDAP System Administration*. However, I purchased both of the books reviewed (even with my complimentary copy of the LDAP book!).

by Robert Haskins

Robert D. Haskins is an independent consultant specializing in the Internet Service Provider (ISP) industry.

rhaskins@usenix.org

LDAP System Administration

LDAP SYSTEM ADMINISTRATION

BY GERALD CARTER

Sebastopol, CA: O'Reilly and Associates. Pp. 294. ISBN 1-565-92491-6.

You might already be asking yourself, "Why would anyone write a book on LDAP when a major author of the protocol (Tim Howes) has already written a book on it?" While there is nothing wrong with the Howes et al. book, it is not "hands-on" enough to be useful for everyone. It covers the theory very well, leaving most of the implementation as an "exercise for the reader." That's where the O'Reilly book is different.

One of the issues with many open source projects is the lack of high-quality documentation to go along with the project. This book is very close to what I would have wanted to produce if I were to write a book on LDAP. Gerald Carter has created an outstanding, well-rounded reference for the OpenLDAP 2.x server, covering enough LDAP basics to enable most system administrators to complete their (LDAP-related) tasks with minimal overhead. If what you are looking for is a protocol reference, then the Howes book will do the trick. However, if you are looking for an LDAP nuts-and-bolts "how to" manual, then the O'Reilly book is what you need.

The book's first section ("LDAP Basics") lays down the necessary theory a typical system administrator needs to know to be able to implement an LDAP-based project. It does an acceptable job of building a foundation from which to base the second section of the book ("Application Integration"). It is by no means an in-depth reference, nor is it intended to be.

The example chapter in this section (Chapter 4: "Building a Company White Pages") does an adequate job of tying the pieces together, though it also exemplifies the "out of order–ness" of some aspects of the book. It would have made more sense to have chapter 5, "Replication, Referrals, Searching, and SASL Explained," come before the example chapter, which, it seems to me, should come last and tie everything in the section together. The book contains other organizational missteps, but these qualify as annoyances more than substantive flaws.

The second section is where this book really shines. The integration of LDAP into applications is much needed coverage for those administrators who like the "cookbook" format. Chapter 6, "Replacing NIS," is an excellent tutorial on how one could use LDAP to handle authentication for a large group of users, either from an existing NIS infrastructure or from scratch. Chapter 7, "Email and LDAP," does a fine job covering integrating mail clients and servers with LDAP. Chapter 8, "Standard UNIX Services and LDAP" excellently explains how to integrate LDAP with Apache, ProFTPD, Samba, FreeRADIUS, BIND 9, and printers.

Chapter 9, "LDAP Interoperability," addresses the issues surrounding LDAP's integration with other types of directory servers such as Microsoft Active Directory and Kerberos. Finally, no book with "System Administration" in the title would be complete without some sort of Perl coverage, and Chapter 10 nicely documents the "Net::LDAP" Perl module. The book contains five appendixes, the most useful being Appendix B,

which contains the OpenLDAP command options. Anyone who has searched through the sources to find the debug levels will appreciate this appendix.

So what's not to like about this book? Not much. Some might object to the "light" theory coverage, but that is a plus in my estimation. The only minor thing missing would be in-depth coverage of the more useful Web-based LDAP browsers/editors available; the single-page coverage here is insufficient in, my opinion.

RADIUS
BY JONATHON HASSELL
Sebastopol, CA: O'Reilly and Associates.
Pp. 190. ISBN 0-596-00322-6.

RADIUS

I was very pleased to see a book on RADIUS finally; it's about time something was published on this topic, though I can understand publishers' reluctance to bring out books in such a limited subject area.

After presenting a good overview of the theory behind RADIUS, the book describes such RFC material (though more accessibly) as packet types, TCP vs. UDP, attribute/value pairs, authentication methods, and realms. Chapter 3 describes authentication and authorization attribute properties and seems to focus on the "Livingston" variant of the standard dictionary, with no coverage of the slight differences in the "Merit" type of dictionaries. This seems like a significant omission, since these slight differences have caused me great headache in the past. Also, no coverage of vendor-specific attribute properties is available, nor of the de facto entries such as those on Ascend.

Chapter 5 is where the FreeRADIUS server coverage begins. The documentation that now ships with FreeRADIUS tells you to use replacement configuration files, which renders obsolete those described in this book! In addition, the entries describing the new format files are incomplete. Chapters 6 and 7 function as catch-all chapters; topics include PAM, proxying, working with particular RAS gear, using MySQL to authenticate (but not for RADIUS accounting), Web authentication, LDAP (this should have had its own chapter), and processing RADIUS accounting records with RadiusReport.

Chapter 8 provides a nice treatment of RADIUS's security issues, and what (little) can be done about them. Chapter 9 describes some of the more widely used RADIUS-related draft standards, including VPN tunnels, the Extensible Authentication Protocol, and interim accounting. Again, there isn't anything here that one cannot get out of the draft RFCs, it's just a little easier to read and understand. Chapter 10, "Deployment Techniques," covers deployment from a broad high-level view but doesn't get into enough detail. Such topics as scaling and switching were omitted, and I found the "case studies" a little too contrived for my liking.

A number of relevant topics are missing from this book. How about some concrete example configurations? (I am a big fan of the cookbook approach.) For example, I wanted to set up a RADIUS proxy environment, but this book was not helpful in my research. There is also no discussion of such basic RADIUS-related Perl modules as perl-RADIUS, and treatment of RADIUS's application to authentication problems is limited. Why not detail how to set up a news server that authenticates via RADIUS, or how to perform wireless authentication via RADIUS?

The book's primary weaknesses – not enough detail and spotty coverage of important topics – are magnified by the fact that FreeRADIUS has evolved significantly since this book was written. Despite its numerous shortcomings, this is currently the only book on RADIUS that I am aware of, so if you have RADIUS and want a reference, this will have to do. Perhaps O'Reilly will publish a second edition after the FreeRADIUS has moved past its current active-development phase, so that some of the book's more glaring deficiencies can be corrected.

Next time, I will take a look at the ISPman LDAP-based software for providing and provisioning ISP-type services. In the meantime, please send me your questions and comments!

patch work

by Dan Geer

Dan Geer is a USENIX Past President and is Chief Technology Officer at @Stake, Inc.

geer@atstake.com

Absolutely no one in this audience needs an introduction to patching. It's a pervasive requirement as well as a pervasive nuisance, and that's putting it mildly. If it weren't such a nightmare it wouldn't be as interesting, and vice versa.

This is not a security article. For a mature environment, security is a subset of reliability; in other words, this is a reliability article. The logic is thus:

If a system is insecure, then
It is unreliable, therefore
Security is necessary for reliability, yet
Security is insufficient for reliability, therefore
Security is a subset of reliability.

The axis of evaluation of a single patch or a system for patching or a regime for the management of patches is one that begs for the metrics of reliability. Reliability metrics are well defined in other fields, so just steal them fair and square. While we're at it, let's not ignore business reality – just as public lotteries are a tax on those who cannot do math, a patch is a tax on those, upstream or down, who make software reliability something that can only be bought on the installment plan.

What, then, is the "physics" of patching? First: The more serious the hole that the patch is to fill, the less time you have to fill it. This is especially true for security patches, because security patches come with the shortest fuses, already lit. For those who actually know physics, this might be our equivalent of "inflation."

Second: Patch density is itself a function of risk over space and time. It is a function over space since no automatic system ever works completely, hence shoe leather is still required, which breaks the first rule of scalability for distributed systems: No shoe leather. It is a function over time as exploitability increases monotonically once a flaw has been discovered and one must always assume that a flaw that is officially known was discovered unofficially much earlier, i.e., the countdown clock is already running by the time you realize that there is something to patch.

Third: Patches are the high-energy radiation around the black hole of a desktop monoculture. The more massive the black hole, the higher the energy of that radiation. The gravity of the desktop monoculture in particular absorbs such a huge mass of the slings and arrows of outrageous fortune seekers (the exploit writers) that it radiates particularly high energy photons (patches whose terminal velocity is proportional to the curve of susceptibility, which in true platform monocultures is a singularity).

Fourth: Critical mass – it's not just for weapons anymore. All the failures that actually matter are cascade failures. Cascade failure, like an epidemic or a chain reaction, is a big-number interaction of virulence (radioactivity) and immunity (isotope stability). What may matter most to you is not whether you patch but whether your communicating counterparties patch. If they are all sending you a tsunami of Slammer, it might not actually matter if you are patched or not. Your susceptibility to the clumsiness of others is the other half of Metcalfe's Law, the half with the sign bit reversed. In the sense of critical mass, "we" are all in this together even if it approximates roping together amateur mountain climbers.

If you're losing a game, first try to change the rules. What might that mean here? For starters, we might consider source reduction, delivery automation, mitigation by interdiction, and risk transfer.

Source reduction means what it sounds like, and it is easier to say than to do. Let's be abundantly clear: This is a reliability matter related to trying to cram 10 pounds of

To err is human; to really foul things up you need computers.

functionality into a five-pound bag, and the only way we'll get more reliable software is to raise prices for software. The richest software vendor in the world mostly doesn't care how much it costs to cure quality flaws so long as it doesn't cost time to market, but source reduction can come as a side effect of liability judgments against suppliers, directly gated by quality metrics imposed by energized buyers, or the market pricing impacts of differential insurance premiums driven by payouts. Somebody will pay; if not, you call it risk transfer.

Delivery automation, also known as automatic update, takes a tough problem and makes it brittle. So long as the automatic update works it is a solid win. When it fails, whether by incompetence or treachery, it demonstrates that while to err is human, to really foul things up you need computers. Nevertheless, it is in our future by contract even if not by technical evaluation. Absent real emergencies, pushed patch notifications with lazy evaluation by the notified client recommends itself, especially if coupled with . . .

Mitigation by interdiction has scaling factors that strongly recommend it if, and this is a big if, there is a perimeter at which to apply that interdiction. Removing toxic attachments from email is a kind of patching – not that the Devil didn't have a backslappingly good day when some clown decided that shoe-horning executables into text messages sounded cool – and "we" think nothing of automated patching of email these days just as I'm sure we'll soon think nothing of patching instant messages, SMS opacities, and even the stuff that comes from P2P "networks."

But let's get back to the net-net of patching with respect to reliability, which is what this is about. As Mike O'Dell used to say, left to themselves creative engineers will deliver the most complex system they think they can debug. He was talking about the counterweight of having enough old hands around whose "sadder but wiser" experience balanced out "what could go wrong?"; but I'm thinking about complexity and the debuggability margin as where complexity and reliability meet. There's no doubt that reliability and complexity have a difficult marriage where each has caused the other pain. Reliability can enable complexity and complexity can enable reliability, but creativity-fueled high rates of change generate the kind of complexity that is decoupled from reliability. That can be good in a greenfield startup; it can be bad in the long distance call switching system.

The needed maturity of the area of application determines the rate of change that can be tolerated, which brings me to a central point: If you think of a patch as a software release, then you divide the world of software releases into the voluntary and the involuntary from your, the end-using customer's, point of view. That, dear friends, allows you to use the ratio of involuntary to voluntary software releases to just flatly rank software vendors. You know what I am thinking, but this is a family magazine.

There are not enough of us (USENIX) to go around, so there has to be substantial automation of patch delivery or a substantial reduction in patch necessity. If the number of people with Internet access is more-or-less doubling every six months, then only one in a million was here when Mosaic first appeared. That's boring, but curves like that absolutely tell you that automation is going to have to substitute for "been there, done that" levels of experience among the great mass of the patch-needy. If what obtains is that there's neither automation nor enough of "us" to go around, then the inevitable injuries to "the innocent" guarantee that the liability law system will take over and genuine innovation will get harder to fund and deploy.

This is bad news, overly simplistic, unignorable. Those of you who can measure things – please do some measurement and publish your numbers. Numbers get believed. Bellyaching gets a cool compress. Clock's ticking.

Oracle hot backup

by Matthew E. Hoskins

Matthew Hoskins is a UNIX system administrator for the New Jersey Institute of Technology, where he maintains many of the corporate administrative systems. He enjoys trying to get wildly different systems and software working together, usually with a thin layer of Perl (aka "MattGlue").

matt@njit.edu

Introduction

I am not really a DBA. However, like many UNIX sysadmins, I am charged with the care and feeding of a number of Oracle database servers. In this short article I will explain the concepts behind various methods of backing up Oracle database servers, a critically important subject for anyone in my position. This is not an introductory article to Oracle, and I do assume at least a basic understanding of SQL and Oracle command line utilities.

Enterprise database servers can seem complex and daunting, but Oracle has a long history and is based on proven technology and concepts that most UNIX system administrators are comfortable with, like textual configuration files, scripting capability, and command line interfaces. Even if your DBAs or programmers manage their own Oracle backups, it can be of value to understand the methods and possibilities.

Oracle Concepts

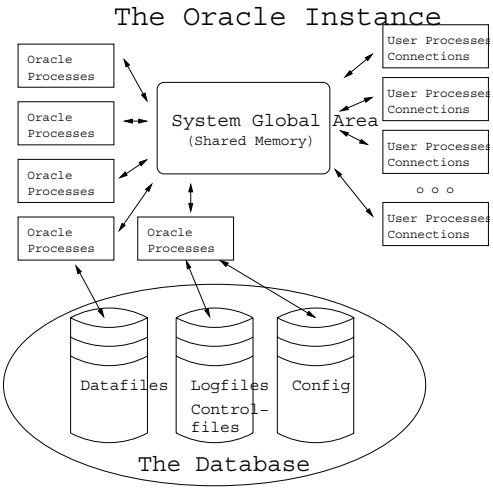
Oracle database servers are composed of "instances" which are identified by an SID (system identifier). An instance contains memory structures, processes, tablespaces, and various data and configuration files. In most circles, the word "database" is synonymous with "instance," at least in Oracle, but in actuality the database is a component of the instance.

Tablespaces comprise a group of datafiles which can be dynamically allocated. During normal operation, Oracle is constantly updating various files inside the instance; these files can be (and usually should be) spread between physical partitions/devices on the system. If Oracle files are spread across physical partitions in a logical manner, the loss of any one partition should not result in any data loss. Oracle calls this layout "OFA" (Oracle Flexible Architecture). Using the concepts of OFA is key in creating a recoverable and manageable Oracle installation. After all, the first step in implementing a successful backup system is having a properly configured database server.

In most cases, Oracle gives you the option to maintain multiple copies of critical files (such as control files(s) and online redo logs) automatically; Oracle calls this "multiplexing." This can give you an extra layer of protection against physical media failure. Every update applied to a datafile is handled as a transaction and can be "rolled back." This is handled internally by maintaining sets of online redo logs. For additional data recoverability, the online redo logs will be automatically archived if an instance is placed in ARCHIVELOG mode. Archive logs enable one to play back every update applied to datafiles/tables literally. This is very useful for recovery from disaster or human error. Using archived redo logs, a DBA can "time travel" and restore any table to any point for which uninterrupted archive logs are available, a truly powerful feature.

Types of Oracle Backup

Three major methods exist for backing up Oracle databases: offline, RMAN, and "online hot." Offline backup is virtually unacceptable in 24/7 production environments, but it is the safest and also easiest from which to restore. During offline backup, the Oracle instance is completely shut down and all files can be safely backed up using normal backup tools (since none of them is changing). Offline backups are still very useful; it is quite common to perform an offline backup both before and after applying Oracle software patches/upgrades.



RMAN (Recovery Manager) is a relatively new method of online backup and is mainly designed to provide an API for integrating automated enterprise backup software with Oracle. One major feature of RMAN is its ability to make incremental backups, which is sometimes necessary for huge tablespaces. However, RMAN has a large overhead of complexity that can be avoided unless you need its advanced features.

Finally, "hot backups" put individual tablespaces in a special mode that enables datafiles to be copied or archived using normal backup tools. The instance is required to be in ARCHIVELOG mode for online hot backups to work. Hot backups generally make the most sense to the average sysadmin, because they may be scripted using tools with which we have great familiarity.

A fourth method of backup, mentioned only for completeness, is the "logical backup" in which one does a partial or full export of the schema and data in an instance. An export creates a single file (which might be broken up) that contains the data and commands to recreate whatever was exported. While this method might seem elegant for backups, in order to do a restore one has to have an already working Oracle installation, which would require running Oracle Universal Installer (OUI) and applying all patches to the same point as the export was created from. This can be rather time-consuming in Oracle and is not recommended as a backup method. Logical imports/exports are better used for point-in-time archives and moving data from one server to another. Import/export can also be useful for copying data from a production instance to a test or development instance. Since import/export recreates database components using data manipulation operations, it can be quite time-consuming on large databases, because the data must be re-indexed as it is loaded.

Hot Backup Concepts

The remainder of this article focuses on hot backups. In my opinion, they are the most flexible method from the standpoint of a sysadmin for small to medium-sized tablespaces. When datafiles are put into hot backup mode, Oracle is still writing to them; this makes your average system administrator raise an eyebrow. Oracle slightly changes the way it performs updates to datafiles by copying whole datafile blocks, rather than just what has changed, into the online redo logs. Hot backups, by their very nature, take a corrupt copy of the datafiles. But relax, it's OK. Since the instance is in ARCHIVELOG mode, all changes to the datafiles are available upon restore to correct any discrepancies created as the archiving process passes over any given datafile. This is why Oracle records redo information in whole blocks during hot backup: to help recover any inconsistent blocks in the datafiles upon restore. The basic steps in online hot backups are as follows (SQL statements are in parenthesis):

1. Get listing of tablespaces and datafiles (SELECT tablespace_name, file_name FROM sys.dba_data_files).
2. For each tablespace
    a. Put tablespace in hot backup mode (ALTER TABLESPACE $TABLESPACENAME BEGIN BACKUP).
    b. Copy, archive, or snapshot each datafile in this tablespace.
    c. End hot backup mode for this tablespace (ALTER TABLESPACE $TABLESPACENAME END BACKUP).
3. Back up the Oracle "controlfile" (ALTER DATABASE BACKUP CONTROLFILE TO TRACE and/or ALTER DATABASE BACKUP CONTROLFILE TO $FILE).

4. Confirm all tablespaces returned to normal mode (SELECT FILE#,STATUS,CHANGE#,TIME FROM v$backup WHERE STATUS != 'NOT ACTIVE').
5. Perform an archive log switch (ALTER SYSTEM ARCHIVE LOG CURRENT).
6. Backup archived redo logs.
7. Backup everything else (binaries, config files, etc . . .).

Most of the steps are performed by executing SQL statements in SQL*Plus or some other method; the archiving of datafiles and software can be done with any normal methods (e.g., tar, cp, cpio). As you can see, this process spans two distinct operating environments. Steps must be performed inside Oracle, then further operations must be executed at the OS and file-system level. This sounds like a job for a script! More on that later.

Proper Treatment of Archive Logs

Archive logs are rather important and should always be hosted on a separate physical disk or array from the datafiles. If you lose the partition that houses datafiles, the archive logs can be used to *completely* recover the datafiles up to the moment they were lost. Depending on the application and its importance, it might be beneficial to copy archive logs off-site a number of times per day.

The "rsync" (*http://rsync.samba.org/*) utility works well for this task, because it can keep a remote copy of a file system up to date by only transferring incremental changes. Since every restore done from hot backups requires datafile media recovery using archived redo logs, these logs should always be archived after all tablespaces are taken out of hot backup mode and kept with the archived datafiles. Archive logs must be treated with respect; a single missing or corrupt archive log file will cause all the logs created after it to be useless! Generally archive logs compress well – many DBAs use gzip or bzip2 to compress the archive logs nightly with a cron job – but make sure you don't try to compress the one that Oracle is writing to.

The Restore

Restoring from hot backups is a multi-step process that is mostly handled by Oracle itself, provided you supply all the files necessary. When one or more datafiles have been lost, the following high-level steps are performed:

1. Take the affected tablespace offline (if Oracle is still running, ALTER TABLESPACE $TABLESPACENAME OFFLINE TEMPORARY, where $TABLESPACE is the tablespace that lost the datafile).
2. Restore lost datafile(s) from last available hot backup.
3. Confirm that archived redo logs are available in the location Oracle expects them to be, and that they are uncompressed. (Oracle will prompt interactively if the ones it's looking for are not found.)
4. Tell Oracle to recover the datafile(s) automatically (ALTER DATABASE RECOVER AUTOMATIC TABLESPACE $TABLESPACE). You can monitor the progress by tailing the alert.log files.
5. Bring tablespace online (ALTER TABLESPACE $TABLESPACE ONLINE).

And you're open for business again.

For a complete listing of scenarios and proper restore procedures, see the Oracle Backup and Restore manual referenced at the end of this document. It is important to make a regular habit of testing your backups. In fact, I would go so far as to say it

would be insane to move forward with a backup solution without testing various restore scenarios. I would suggest testing at least loss of a single datafile and loss of the entire ORACLE_HOME and all datafiles. For testing, it is quite easy to set up a secondary Oracle instance which can be used to test backups. If this system is in a different geographic location, it can be used for disaster recovery also.

In any event, it is a good idea to completely script the process of restoring all datafiles and recovering the instance. This can make disaster recovery much less painful. If you are the person people come looking for when things are broken, this is definitely a good thing. It is, of course, important to keep this script up-to-date as circumstances change around it. Also, while this standby system is not being used for disaster recovery, it can be used as a perfect playground for development.

Script Design

There is no shortage of good Oracle hot backup scripts freely available; simply do a Google search for "Oracle hot backup script". However, none of the ones I found exactly scratched my itch. The solution I designed uses disk-to-disk backups for our various databases (MySQL, LDAP, Oracle, etc.). Disks are quite cheap these days; large IDE RAID systems and either direct attached or NAS (Network Attached Storage) devices are surprisingly affordable. These high-capacity/low-cost disk arrays can be used to widen your backup window by first staging data to disk before tape. Also note, bandwidth permitting, these disks might be off-site, thus automating off-site backups and eliminating the expense of an off-site backup service.

For certain high importance databases, we may keep several backup images on disk to save time if we need a restore. Using this methodology, our database dump process does not have to be synchronized with our enterprise network backup system which finally puts the data on tape. Another method we use (snapshots) exploits features available in some modern file systems that enable you to create a temporary read-only view of a file system from a particular point-in-time. Since these snapshots are normally instantaneous, it is common to place *all* tablespaces in hot backup mode during snapshot, then immediately disable hot backup after snapshot is completed. Since I use both of these methods of backing up databases, the script I constructed is able to handle both procedures. Some considerations when selecting any hot backup solution are:

1. Graceful error handling. If an error occurs during a hot backup, it is vitally important that the tablespace be taken out of hot backup mode before the script bombs out. During hot backup mode, Oracle takes a performance hit because it is recording redo data in full blocks, and it is important that the database not be allowed to remain in this mode for long periods.
2. Sanity checking. It is very important that the script check that the proper environment variables are set and are sane values. Also, some checking should be done to confirm the database is in a good state to be backed up. For instance, no datafiles should be listed as "Needs recovery." The solution should also have sufficient locking that overlapping instances of the backup process will not clobber each other.
3. Event reporting. Errors (and success) should be reported. Syslog works nicely for this; in my organization we use a home-grown solution for real-time monitoring logs from all our systems for various events and use this as a basis for sending alerts.

4. Ease of restoring. Backups created should be in a universal and open format that preserves owner, mode, and original location of the datafiles (like tar with full path).
5. Should be able to clean up its own messes. If a backup is interrupted in the middle of its execution, it can leave tablespaces in hot backup mode. This condition should be detected and there should be a method for correcting the situation without needing to enter SQL*Plus manually.

The script I cooked up is written in Perl, has been tested on Oracle 8i and 9i on Linux and Solaris, and is released under GPL, so it is free for use in your organization. Get it at *http://www.njit.edu/~matt/oracle-hot-backup/*.

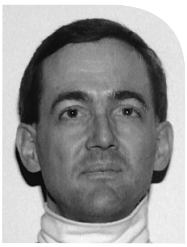
References

Oracle, *Oracle9i User-Managed Backup and Recovery Guide*, June 2001, Part No. A90134-01.

Michael Wessler, *Oracle DBA on Unix and Linux* (Sams, 2002).

working with C# classes

by Glen McCluskey



Glen McCluskey is a consultant with 20 years of experience and has focused on programming languages since 1988. He specializes in Java and C++ performance, testing, and technical documentation areas.

glenm@glenmccl.com

In previous columns, we've looked at some of the basics of the C# language. It's now time to start examining in more detail how C# classes work. A class is a user-defined type, and serves as the fundamental unit of design and composition for C# programs.

A Class to Represent X,Y Points

Let's start our discussion by looking at a class whose instances represent X,Y points. Previously we defined a class as being a combination of some data (think of a C struct) plus operations that manipulate instances or objects containing that data. For a Point class, the data would likely be a couple of integers representing the point (like 25,100), along with operations to initialize a point object, access the X,Y values in an object, compare an object to other point objects, convert an object to a string for printing and formatting, and so on.

Here's some C# code that illustrates these ideas:

```
using System;

public class Point {

    private int x, y; // X,Y data fields

    // constructor
    public Point(int x, int y) {
        this.x = x;
        this.y = y;
    }

    // copy constructor
    public Point(Point p) {
        this.x = p.x;
        this.y = p.y;
    }

    // accessor methods
    public int getX() { return x; }
    public int getY() { return y; }

    // conversion to string

    public override string ToString() {
        return String.Format("({0},{1})", x, y);
    }

    // equality check against another Point object
    public override bool Equals(object obj) {
        if (!(obj is Point))
            return false;
        Point p = (Point)obj;
        return x == p.x && y == p.y;
    }

    // hash code for the object
    public override int GetHashCode() { return (x << 16) | y; }
}

public class Test {
    public static void Main() {

        // create some Point objects on the heap:

        Point p1 = new Point(50, 75);
        Point p2 = new Point(50, 75);
        Point p3 = new Point(100, 150);
        Point p4 = new Point(p3);// copy constructor for Point
        // access the X,Y values of an object

        Console.WriteLine("p1 X,Y = {0},{1}", p1.getX(), p1.getY());

        // convert object to string and print it

        Console.WriteLine("p4 = {0}", p4);

        // exercise the overridden Equals() method

        if (p1.Equals(p2))
            Console.WriteLine("p1/p2 equal");
        if (p2.Equals(p3))
            Console.WriteLine("p2/p3 equal");
        // get the hash code for an object

        Console.WriteLine("p1 hash code = {0}", p1.GetHashCode());
    }
}
```

The Point class defines two data fields to hold the X,Y values. These are private fields, which means that only methods of the Point class can access the fields. In particular, it's not legal to say:

```
Point p = new Point(10, 20);
p.x = -125;
```

If this kind of operation is allowed, then the internal representation details of a Point object are exposed to the user, generally an undesirable thing (especially if the representation changes at a later time). Also, allowing the field to be set directly may violate the integrity and domain of the object. For example, if a

check is made that X,Y are positive integers when the object is created, then setting the X value to -125 might cause havoc.

The first two methods of the Point class have the same name as the class itself (Point), and thus are constructors, special methods used to initialize Point objects. The memory for objects is allocated from the heap, and the constructor is called to initialize the raw memory.

The second constructor is a copy constructor, used to make a copy of an already existing Point object. In lieu of using such a constructor, we could instead say:

```
Point p1 = new Point(10, 20);
Point p2 = new Point(p1.getX(), p1.getY());
```

but a copy constructor is a more general mechanism. Note that a byte-by-byte copy of an object is rarely the right choice, given that an object may contain internal references to other objects.

getX and getY are accessor methods, used to access the private X,Y fields in Point objects.

ToString, Equals, and GetHashCode are methods found in the root class (System.Object), and overridden in Point to provide custom behavior.

For example, the method System.Object.ToString has certain generic behavior, illustrated in this example that defines a dummy Point class:

```
using System;

public class Point {}

public class Test {
      public static void Main() {
          Point p1 = new Point();
          Console.WriteLine(p1);
          Point p2 = new Point();
          Console.WriteLine(p2);
      }
}
```

When Console.WriteLine is called, its argument (p1 or p2) must be converted to a string for printing; the method System.Object.ToString is used for this because we didn't override ToString in the dummy class. The default ToString behavior uses the name of the class itself as the value returned by ToString, so the result of running this program is:

```
Point
Point
```

For real classes, ToString should have per-object customized behavior, by including, for example, the distinct X,Y values found in a Point object.

The same consideration applies to the Equals method, used to compare two Point objects for equality. Such equality checking needs to be more sophisticated than merely doing a binary comparison of the corresponding bytes in the two objects. For example, objects may contain references to sub-objects, and comparing the memory pointers of the sub-objects will not work.

GetHashCode is used to compute a hash code for an object. The hash code is used by collection classes that represent groups of objects, such as a hashtable.

When the Point class demo is executed, the output is:

```
p1 X,Y = 50,75
p4 = (100,150)
p1/p2 equal
p1 hash code = 3276875
```

Other Ways to Implement the Point Class

There are other ways we could define a Point class. For example, we could use a struct instead of a class. A struct is a simpler form of a class, with some restrictions and differences. Struct objects are allocated on the stack instead of the heap, and, as discussed in our previous column, have value rather than reference semantics.

Another possibility is to use a class in a very simple and low-level way:

```
public class Point {
      public int x, y;
      public Point(int x, int y) {
          this.x = x;
          this.y = y;
      }
}
```

This approach is not much different from using a C struct. Doing things this way violates the whole object-oriented paradigm but, at the same time, is a simple approach sometimes useful in casual programming, e.g., when you're building a prototype.

A third alternative is to use C# properties. A property is kind of a cross between a data field and a method. Properties are referenced like data fields, but have get/set methods to control the access.

Creating and Reclaiming Objects

In the discussion above, we talked about how memory is allocated for class objects, with the class's constructor called to initialize the memory. What happens when an object is no longer in use? How do you get rid of it and reclaim the space?

C# uses automatic garbage collection to reclaim objects, so most of the time you don't need to worry about the details of memory management. What does this mean in practice? Let's look at an example:

```
using System;
using System.Threading;

public class CtorDtor {

    // constructor
    public CtorDtor() {
        Console.WriteLine("constructor called");
    }

    // destructor (actually the finalize method
    // commented below)
    ~CtorDtor() {
        Console.WriteLine("destructor called");
    }

    //protected override void Finalize() {}
}

public class Test {

    // create a CtorDtor object, which
    // becomes garbage when f() exits

    static void f() {
        CtorDtor cd = new CtorDtor();
    }

    public static void Main() {
        f();

        //GC.Collect();
        //Thread.Sleep(500);

        Console.Write("Press Enter to quit program: ");
        Console.ReadLine();
    }
}
```

In this code, the Main method calls f, and f calls new to create an object of the CtorDtor class on the heap. Then f returns, and at this point, there is no way to reference the CtorDtor object created in f, and thus this object has become garbage.

Such garbage is subject to reclamation at any time, but garbage collectors typically run in a separate program thread and try to minimize performance overhead. For example, a garbage collector may run only when free memory is getting low. It's unwise to assume particular garbage collection behavior.

When the garbage collector runs, one of the things it does is call finalize methods on objects. A finalize method is used to perform any cleanup that is required (other than reclaiming space) – for example, freeing up system resources not under the control of the C# runtime system.

C# supports C++ destructor syntax, like this:

```
~CtorDtor() {}
```

but this syntax is actually an alias for:

```
protected override void Finalize() {}
```

Finalize methods are not really the same as destructors. In the C++ world, a destructor is called when an object goes out of scope (such as a stack-based object at function exit) or when the delete operator is called for a heap-allocated object.

By contrast, a finalize method is called by the garbage collector, and it's risky to rely on the garbage collector behaving in a specific way (or running at all).

When the CtorDtor demo program runs, the output indicates that the destructor (finalize method) is not called until the program exits. One way of forcing the finalizer to run sooner is to explicitly call garbage collection, using the commented lines of code. This approach works, but is not recommended. Before you start relying on explicit garbage collection calls, it pays to sit down and really study how garbage collection works – it's a tricky area to make assumptions in.

Explicitly Disposing of Objects

Garbage collection isn't always effective in calling finalize methods in a timely way. What do you do if you have a class whose objects represent critical system resources, resources that are in short supply? One solution is to implement the IDisposable interface in a class:

```
using System;

public class CtorDispose : IDisposable {

    // a resource, with a value of -1 indicating
    // that the resource is currently unallocated

    private int resource = -1;

    public CtorDispose() {
        // allocate resource
        resource = 100;
    }

    ~CtorDispose() { DoDispose(); }
    void DoDispose() {          // free up resource
        resource = -1;
    }

    public void Dispose() {
        DoDispose();
        GC.SuppressFinalize(this);
    }
}
```

```
public class Test {
    public static void Main() {
        CtorDispose cd = new CtorDispose();
        cd.Dispose();
    }
}
```


An interface like IDisposable declares certain methods, in this case Dispose. A class that implements the interface must define the methods. Implementing IDisposable and defining Dispose means that a class offers a way to directly force object cleanup, without waiting for the finalize method to be called by the garbage collector (the finalize method cannot be called directly, and garbage collection may not occur in a timely way).

In the code above, Dispose is called directly, and the method frees the system resource. The garbage collector is then informed that the object (this) should not be finalized at garbage collection time.

In future columns, we'll look at some further details of how classes work, and how classes and interfaces can be combined to build applications.

the tclsh spot

by Clif Flynt

Clif Flynt is president of Noumena Corp., which offers training and consulting services for Tcl/Tk and Internet applications. He is the author of *Tcl/Tk for Real Programmers* and the *TclTutor* instruction package. He has been programming computers since 1970 and a Tcl advocate since 1994.

clif@cflynt.com

Monitor your system with expect

The previous few *Tclsh Spot* articles described generating IP packets to attack a firewall system being tested. Once the packets can be generated, the next step is to confirm that they are captured and logged correctly.

For someone familiar with the log files, this is a simple exercise. You can tail -f the appropriate file and watch for the log messages, or grep for the expected pattern.

For someone familiar with the log files, this is also a tedious exercise. Not too bad when you are looking for something tricky, but too much monkey-work if you are generating hundreds of attacks and need to confirm that all are caught correctly.

Automating tedious tasks that would otherwise be done by a skilled (or at least easily bored) human is what the expect extension was written for.

The expect extension to Tcl was developed by Don Libes of the NIST in 1990, very shortly after John Ousterhout's first Tcl paper was presented at the USENIX Annual Technical meeting.

Expect will interact with a text-oriented task, looking for patterns, and sending responses as appropriate.

For those with fond memories of UUCP chat scripts, you can think of expect as a chat script with much more power.

The expect extension can be dynamically loaded into a running Tcl shell with the package require command like this:

```
% package require expect
5.38.0
```

It's more common to use the standalone expect shell – invoked as expect. When invoked as expect a special prompt is provided showing the depth of the evaluation stack, and the number of commands that have been processed.

Defining a simple procedure within the expect shell looks like this:

```
$> expect
expect1.1> proc a {a b} {
+> puts "$a"
+> }
expect1.2>
```

The expect package is extensive with many commands and options to support the various interactions one might need to perform. There are only three essential commands, however.

spawn	Starts a new task for the expect script to interact with.
expect	Defines a set of patterns and actions to perform when a pattern is matched.
exp_send	Sends a stream of data to the child task.

The spawn command starts a child process to be controlled by the expect script. It starts a new process and connects the process's stdin and stdout to the expect interpreter.

Syntax: spawn *options commandName commandArgs*

options
: The spawn command supports several options, including
-noecho
: The spawn will echo the command line and arguments unless this option is set.
-open fileID
: The -open option lets you process the input from a file handle (returned by open) instead of executing a new program. This allows you to use an expect script to evaluate program output from a file as well as directly controlling a program.

commandName
: The name of the executable program to start.

commandArgs
: The command line arguments for the executable program being spawned.

Opening an ssh session would look like this:

```
expect1.1> spawn ssh -l clif 127.0.0.1
spawn ssh -l clif 127.0.0.1
1105
expect1.2>
```

Once a new child task has been spawned, your script needs to interact with it. The expect command listens to the child task, while the exp_send command talks to it.

The expect command will scan the input for one of several patterns. When a pattern is recognized the associated action will be evaluated.

Syntax: expect *?-option? pattern1 action1 ?-option? pattern2 action2 ...*

-option Options that will control the matching are:
- *-exact* Match the pattern exactly.
- *-re* Use regular expression rules to match this pattern.
- *-glob* Use glob rules to match this pattern.

pattern A pattern to match in the output from the spawned program.

action A script to be evaluated when a pattern is matched.

This code snippet will look for the password prompt and report that the prompt was seen:

```
expect1.3> expect {
+> word: {puts "The password: prompt was seen"}
+> }
```

The pattern/action format of the expect command resembles the switch command. The switch command uses the meta-pattern default to process unexpected conditions. The expect command also supports meta-patterns to process unexpected input.

The eof meta-pattern matches when the expect interpreter loses contact with a child task. Usually this happens when a child task dies.

The timeout pattern will match if no other pattern has been matched within a given period of time. This script checks for the password prompt, and complains if it doesn't appear:

```
expect {
    word: {puts "The password: prompt was seen"}
    timeout {puts "There was no password prompt!"}
}
```

The default timeout period is 10 seconds. This can be changed by setting a variable named timeout to the number of seconds to wait before matching the timeout pattern.

Note that the timeout value is a maximum value, not a minimum value. The timeout is implemented by watching the system's one-second timer. Each time this fires, the timeout value is decremented by one. When the value hits zero, the timeout pattern is matched, and the associated script is evaluated.

Thus, setting the timeout value to 1 will never wait longer than a single second, but could timeout in .0001 seconds.

This code snippet will watch for the password prompt and complain if either the child task has lost contact with the expect script or there is a timeout.

```
expect {
     word: {puts "The password: prompt was seen"}
     eof    {puts "The child task has died!"}
     timeout {puts "There was no password prompt!"}
}
```

The flip-side to listening to a child task is to send strings of data to the child. The exp_send sends string to the child process.

Syntax: exp_send *string*

string The string to be transmitted to the slave process. Note that a newline is not appended to this text.

A simple script to look for the password prompt and send a password looks like this:

```
expect {
     word: {exp_send "PASSWORD\n"}
}
```

Notice the \n at the end of PASSWORD. In order to interact with applications that key off of single keystrokes, the expect command sends exactly what you tell it to, and does not add any new characters (like newlines) to the string.

A script to log into a remote system, look for the shell prompt, and change to the /var/log directory would look like this:

```
spawn ssh 127.0.0.1

expect {
     word:    {exp_send "$password\n"}
     timeout {error "Timeout waiting for password"}
     eof       {error "Eof waiting for password"}
}

expect {
     "%>"     {exp_send "cd /var/log\n"}
     timeout {error "Timeout waiting for prompt"}
     eof           {error "Eof waiting for prompt"}
}
```

As with the switch statement's default option, you don't need to use the timeout and eof patterns. But, as with the switch (or C case statement), if you ignore catching the exception conditions, you will live to regret it.

This can lead to a lot of repeated code if your application has a lot of simple challenge/response interactions.

One solution to this problem is to create a procedure to maintain the process flow and provide the customization with the procedure arguments.

The syntax for the proc command is:

Syntax: proc *name args body*

The args and body parameters are generally grouped with braces when you define a procedure within your application.

This procedure generalizes the challenge/response nature of many conversations.

```
##############################################
# proc challResp {pattern response info}
# Hold a single interchange challenge/response
# interaction.
# Arguments
#    pattern:    The pattern to wait for as a challenge
#    response:  The response to this pattern
#    info:         Identifying information about this interac-
#                  tion for use with exception reporting
#
# Results
#
#
proc challResp {pattern response info} {
     expect {
        $pattern    {exp_send "$response\n"}
        timeout    {error "Timeout at $info"}
        eof          {error "Eof at $info"}
     }
}
```

Using the challResp procedure, automating this conversation:

```
%> ssh -l root 127.0.0.1
root@127.0.0.1's password:
Last login: Thu Jun 5 05:45:02 2003 from localhost
Have a lot of fun...
%> cd /var/log
%> tail -f messages
```

can be done with these four lines of code:

```
spawn ssh -l root 127.0.0.1

challResp "word:" $passWord "password prompt"
challResp "%>" "cd /var/log" "first shell prompt"
challResp "%>" "tail -f messages" "second shell prompt"
```

We can create a generic procedure for watching log files for a given pattern with this procedure.

```
##############################################
# proc watchFile {host passwd fileName pattern}—
#    Watch a file on a remote system for a given pattern
# Arguments
#    host:        IP Address of the host on which the file
#                 resides
#    passwd:    Password for accessing the host
#    fileName:  Full path to the file to be watched
#    pattern:    A pattern to watch for in the file
# Results
#    Throws an error if the expected pattern is not seen
#
proc watchFile {host passwd fileName pattern} {
    global spawn_id
    set stty_init -echo
    spawn ssh -l root 127.0.0.1

    challResp "word:" $passwd "password prompt"
    challResp "%>" "cd /var/log" "first shell prompt"
    challResp "%>" "tail -f messages" "second prompt"

    expect {
        "$pattern" {puts "ROOT LOGIN SUCCESSFUL"}
        timeout {error "timeout" "Timeout  \"$pattern\""}
        eof    {error "eof" "Eof waiting for \"$pattern\""}
    }
}
```

Note the global spawn_id at the beginning of this procedure. When spawn creates a new task, it creates a unique identifier for that task and saves that value in the variable spawn_id in the current scope. If the new task is spawned in a procedure, spawn_id is created in the procedure scope, and it vanishes when the procedure returns. Keeping the spawn_id in the global scope is the simplest way to deal with this for applications that only have a single child task.

This discussion of expect barely scratches the surface of what the package can do, but it's enough to automate accessing a remote system and scanning a log file for a pattern.

The next Tclsh Spot article will start looking at ways to coordinate the packet generators described previously with this log file scanner to generate attacks on a test system and confirm that the test system responds to the attacks correctly.

practical perl
Cleaning Up with Modules

by Adam Turoff

Adam is a consultant who specializes in using Perl to manage big data. He is a long-time Perl Monger, a technical editor for *The Perl Review*, and a frequent presenter at Perl conferences.

ziggy@panix.com

A friend of mine who occasionally writes some Perl was working on a CGI program to automate data entry into a database. He is not a programmer by trade, but he uses Perl to manage information critical to his job. Programming is a hobby for him, more intellectually stimulating than trainspotting and less important to him than his true calling, studying the English language.

My friend asked me to look at some of his CGI programs. His programs worked, but he felt they were "ugly." A Java programmer colleague of his recommended that he clean them up by creating a series of Java-like objects and classes. A better way to clean up a Perl program is through a mixture of objects, classes, and standard Perl data structures.

Every so often, we all write programs that just "feel wrong." A good indication is when it takes too much effort to do something that should be easy to do. Another indication is when some Perl feature is used in what feels like an "unclean" fashion. Such misuses can increase the cost of maintaining and extending a Perl program over time, and can lead to programs that work yet are difficult to fathom.

Creating Dynamic Web Pages

Perl is a great language for casual programmers who write code occasionally in service of a greater goal. These people often use Perl to write small a CGI program that automates some task. There are very many ways to write Web-based applications, including using a templating system, building XML-based systems, and using the venerable CGI.pm module. Using CGI.pm to generate dynamic HTML pages may not be the best system or the most elegant mechanism, but it works well and is quite easy to use.

Using CGI.pm can have its downsides, though. Its HTML-generation interface is an awkward way to replace HTML syntax with Perl syntax to create static HTML. This technique obscures the content of the document being created by hiding it within nested Perl subroutine calls:

```
#!/usr/bin/perl -Tw
use strict;
use CGI qw(:standard);
print header('text/html');
print start_html("Test Page");
print h1("Test Page");
print table(Tr(td("a"), td("b")), Tr(td("c"), td("d")));
print end_html();
```

A better use of CGI.pm's HTML-generation interface is to focus on wrapping repetitive Perl data inside HTML tags through simple loops. In this fashion, data can be abstracted into one of many places: a configuration file, a module, or a database. The structure of the HTML to be created is less likely to change frequently, so separating the content (data) from the presentation (HTML-generating code) will tend to simplify the program in the long run. After all, it is easier to update a database or change a configuration file than it is to modify and test a program for every little data fix.

This is the approach my friend used with his code. The goal in the following code samples is to create a simple HTML form with a series of fields. The fields are specified in the @fields array, and the code to generate the form simply iterates over all of the fields to be displayed. Each element in the @fields array is a hash reference that defines all of the aspects of an HTML form field to be displayed.

```
#!/usr/bin/perl -Tw
use strict;
use CGI qw(:standard);

my @fields = (
    {
      -name => "name",
      -label=> "Name",
      -size=>50,
      -maxlength=>50,
      -procedure=>\&CGI::textfield,
    },
    {
      -name => "subscribe",
      -label=> "Subscribe",
      -values=>[qw(Yes No)],
      -procedure=>\&CGI::radio_group,
    },
    {
      -name => "availability",
      -label=>"Availability",
      -values=> [qw(Mon Tue Wed Thu Fri)],
      -procedure=>\&CGI::checkbox_group,
    },
    {
      -name => "state",
      -label=>"State",
      -values=> [undef, qw(Maryland DC Virginia)],
```

```
        -procedure=>\&popup_menu,
    },
    …
);
…
```

For added simplicity, each field definition within @fields contains a reference to the CGI.pm function that will display that field. This has the effect of making the @fields array a dispatch table as well as a rudimentary data dictionary. The hash keys in the element definition are taken from the keys used by the CGI.pm HTML-generation subs that create each kind of input field. By combining these two features, creating the HTML form becomes very easy:

```
…
my @rows;
foreach my $row (@fields) {
    my $sub = $row->{'-procedure'};
    push (@rows, Tr(td($row->{-label}), td($sub->(%$row))));
}
print start_form(), table(@rows);
print submit(), reset(), end_form();
…
```

Designing with Modules

After working with this program for a while, my friend felt that the code was starting to get ugly. His colleague, a Java programmer, recommended creating a class called a FieldList object to represent a list of HTML form fields, represented by Field objects. Processing a FieldList would involve creating a FieldListIterator object to examine the elements one by one.

This is an area where Perl programmers and Java programmers have differing opinions. Perl programmers generally feel that creating artificial container objects like FieldList and FieldListIterator are a waste of time. Perl's array variables and the foreach loop exist to hold lists of values and examine them iteratively, thus obviating artificial container and iterator classes.

There is some wisdom in creating a Field class, or rather a hierarchy of field classes to handle different kinds of HTML form fields. The field definitions listed above in @fields are somewhat repetitive. Each field will contain very similar values for the -label and -name keys. The -label key specifies the text to appear to the left of a form field, and the -name key specifies the name of the form variable to appear in the HTML output. Note that there's a very simple relationship between the -name and -label values: In our sample program, the -name value can be derived from the -label value by lowercasing it and removing space characters.

There are other common behaviors shared by all field objects. In this program, all fields are displayed within a two-element HTML table row, with the text label appearing in the left cell and the HTML form input field appearing in the right cell. And then there are differences between individual field types. Setting the maxlength makes sense only with text fields.

A better way to structure this code would be to create a simple Field package (or class) that handles the common behaviors and then create specialized packages that describe the actual differences between text fields, radio groups, checkboxes, and other HTML form input fields. Here's the Field class that describes the common behaviors of HTML form input fields for this application. Note that the init will create a default CGI variable name from a field's label, a technique that reduces redundancy found in the older version of the program.

```
#!/usr/bin/perl -w
use strict;
package Field;
use CGI qw(:standard);
sub init {
    my $self = shift;
    my %params = @_;

    ## Assign the key/value pairs for this object
    while(my ($key, $value) = each %params) {
        $self->{$key} = $value;
    }
    ## Create the field name from the text label
    my $name = "\L$self->{-label}";
    $name =~ s/ /_/g;
    $self->{-name} = $name;
    return $self;
}
sub display_row {
    my $self = shift;
    my $sub  = $self->{-procedure};

    return Tr(td($self->{-label}), td($sub->(%$self)));
}
```

Creating a package to describe text fields is not difficult. The first step is to declare that this text field package inherits from the Field package (via a use base; declaration). This package needs a constructor (a method named new) that creates new text field objects with a minimal amount of information specified by the user. The only information that is absolutely necessary is the label for this input field. The CGI field name for this input field can be derived using the Field::init method. Attributes like the field size and the maximum input length can be defaulted to sensible values. Finally, all text fields will use the CGI::textfield subroutine to display themselves.

Here is the complete package definition this application needs to create CGI text input fields. The new method receives the label for a text field and uses sensible default values for the rest of the field's attributes. The size and maxlength methods exist to override the default values for the -size and -maxlength attributes.

```
package Field::Text;
use base 'Field';
use CGI;
sub new {
    my $class = shift;
    my $label = shift;
    ## Create an object
    my $self = bless {}, $class;
    ## Finish initialization
    $self->init(-label => $label,
        -size               => 50,
        -maxlength          => 50,
        -procedure          => \&CGI::textfield);
}
sub size {
    my $self = shift;
    $self->{-size} = shift;
}
sub maxlength {
    my $self = shift;
    $self->{-maxlength} = shift;
}
```

The other input field types (checkbox groups, pop-up menus, radio groups) are all similar in that they have multiple values. A good way to describe these similarities is to define a Field::Group package and derive specializations of that package for the specific input field types.

```
package Field::Group;
use base 'Field';

sub init_group {
    my $self = shift;
    my $procedure = shift;
    my $label = shift;
    my @values = @_;
    $self->init(-label => $label,
        -procedure          => $procedure,
        -values             => \@values);
}
```

The last few packages handle creation of radio groups, checkbox groups and pop-up menus. They are all very similar; only the procedure to display them varies:

```
package Field::RadioGroup;
use base 'Field::Group';
use CGI;

sub new {
    my $class = shift;

    my $self = bless {}, $class;
    $self->init_group(\&CGI::radio_group, @_);
}
package Field::CheckboxGroup;
use base 'Field::Group';
use CGI;

sub new {
    my $class = shift;

    my $self = bless {}, $class;
    $self->init_group(\&CGI::checkbox_group, @_);
}
package Field::PopupMenu;
use base 'Field::Group';
use CGI;

sub new {
    my $class = shift;

    my $self = bless {}, $class;
    $self->init_group(\&CGI::popup_menu, @_);
}
```

Cleaning Up with Modules

Now that we have a set of packages for creating CGI input fields, it's time to revisit the application. Recall that the first step was to create a list of field definitions, then create the HTML form. With the Field modules created for this application, it is easy to simply and succinctly declare what fields are used in this CGI application. There is no repetition in defining -label and -name attributes, and the common attributes of each type of field are defined once and only once.

```
#!/usr/bin/perl -Tw

use strict;
use Field;        ## pull in all the Field packages
use CGI qw(:standard);

my @fields = (
    new Field::Text('Name'),
    new Field::RadioGroup('Subscribe', 'Yes', 'No'),
    new Field::CheckboxGroup('Availability', 'Mon',
            'Tue', 'Wed', 'Thu', 'Fri'),
    new Field::PopupMenu('State', undef,
        'Maryland', 'DC', 'Virginia'),
);
...
my @rows;
foreach my $field (@fields) {
    push (@rows, $field->display_row());
}
print start_form(), table(@rows);
print submit(), reset(), end_form();
...
```

Conclusion

Over time, all software has a tendency to be extended beyond its original design and "get ugly." One technique for improving code that "feels wrong" is to restructure it and extract common behaviors into modules. In this example, the code for constructing CGI form fields was abstracted into a Field package and its subpackages. The main CGI program was simplified and now focuses on *what* fields need to be created, not the arcane details of *how* to create those fields.

where can I find a good investment?

by Ray Swartz

Ray Swartz ran his own computer training and consulting company for 20 years, until stepping away from gainful employment in 2000. Watching the stock market and his retirement portfolio move up and down like a yo-yo has given him a new appreciation of his tolerance for risk.

raybo@idiom.com

Let's say you've added up your income and expenses, crafted your ideal retirement, and know what kind of return you need on your retirement assets to make it happen just the way you want. Now, the only issue left is where to invest your money.

Library shelves are loaded with books identifying the best way to invest. Every day, television and radio programs offer financial advice from trained professionals. Emails offer hot stock tips. Even, friends and colleagues have ideas for you. Given the thousands of investment options available, how is someone supposed to pick a good place to put one's money?

For most people, putting their hard-earned dollars into specific investments is one of the scariest things they do. This can be so stressful that people often keep their retirement funds in cash to put off making such choices. It is easy to dismiss this investment-decision avoidance as an irrational fear of what might go wrong. However, I believe that most of us experience this concern when our own money is being put on the line.

As I discussed in my last column ("Isn't That a Little Risky?" April 2003), you are taking some kind of risk no matter what you do with your money. We aren't trying to avoid risk; we simply want to use risk to our advantage. The key is to make investments that offer the return we need at the smallest risk.

You Bet Your Sweet Assets

The kinds of investments you can make may be limited by the kind of retirement accounts you own. Different types of accounts provide varying degrees of choice. Some retirement plans are completely self-directed and allow you to move your money when and where you want. Some programs have a limited set of mutual fund families to choose from. Other accounts are limited to a single family of mutual funds. Many people have more than one kind of account, which often provides more flexibility but increases complexity.

Regardless of the choices you do have, you still need to decide on exactly which investments you are going to make. How might you go about doing this?

The place to start is with the amount of return you need to meet your retirement goals. That number, alone, will begin to winnow down your options. Put another way, each possible investment has an expected return and risk level associated with it. Since we want to minimize our risk while earning a specified return, we can begin eliminating those investments that don't fit these criteria.

Recall that the key to minimizing risk is diversification. Thus, the question we want to answer is, What collection of available investments will minimize the risk for our desired return? In financial parlance, we are trying to set our "asset allocation." That is, we want to determine what percentage (allocation) of our investment dollars (assets) we want to put into each chosen investment to meet our goals.

The first step in asset allocation is deciding how many asset classes to use. The basic allocation is among cash, stocks, and bonds. In fact, investment advisors often identify what percentage of each they recommend for their clients. A typical allocation might be 5% cash, 55% stocks, 40% bonds. Using historical data, we could calculate the expected return and risk of this allocation.

In reality, most asset allocations add detail by further sub-classing stocks and bonds. Thus, you might divvy stocks into domestic and international and then into large-cap, mid-cap, and small-cap. Bonds, too, might be separated by location (domestic and international), length (short-, intermediate-, and long-term), and "quality" (government, corporate, or junk).

Even cash has distinctions. There are CDs, money market accounts, and saving certificates. In addition, short-term bonds, government or corporate, are usually identified as cash investments.

If we used just these categories, we'd have 18. Let's simplify it a bit by representing all international stocks and bonds as separate, single categories; that gets us down to 13.

Stocks
Domestic large-cap
Domestic mid-cap
Domestic small-cap
International stocks

Bonds
Domestic government intermediate-term
Domestic government long-term
Domestic corporate intermediate-term
Domestic corporate long-term
International bonds
Junk bonds

Cash
CDs
Money market account
Short-term bonds

While this is a fairly detailed list, it is only a simplified example. It doesn't include one of the biggest investments most people make, real estate, and lumps stocks by size but not by type, such as growth or value.

Let's suppose we decided to use these 13 categories. By assembling historical data into expected returns and risk for each category and calculating the covariances for each pair of asset classes, we can determine which combination of these possible investments would yield the least risk for the return we are seeking. That result would be our asset allocation.

As I said in my last column, this data manipulation is done by a financial advisor using one of several available software packages. One drawback to these asset allocation packages is that they often use different asset classes, making it hard to compare results. But, for the sake of demonstration, let's say our (or our advisor's) software uses just the categories listed above.

The process would work something like this: We would specify the return we want to achieve and the program would spit out a set of recommended percentages for each asset class. Then we would compare these percentages to our current investment mix to determine what changes we need to make in order to conform to our recommended asset allocation. In virtually all cases, this will involve selling some investments and buying others.

Choosing Specific Investments

Above, I described a sample allocation as 5% cash, 55% stocks, 40% bonds. Let's suppose that our more detailed allocation broke it down to:

Cash (5%)
3% Money market account
2% CDs

Stocks (55%)
22% Domestic large-cap stocks
17% Domestic small-cap stocks
16% International stocks

Bonds (40%)
15% Domestic corporate long-term bonds
8% Junk bonds
7% Domestic corporate intermediate-term bonds
5% International bonds
5% Domestic government long-term bonds

The next step is to find specific investments that fit the categories listed. We could buy individual stocks and bonds. However, my preference is for mutual funds, even though they have much higher costs. I prefer to let the fund managers do the work of buying, selling, and researching actual stocks. Here is where your choices might be limited due to the type of retirement account you hold.

If you are in a company 401(k) plan or have your accounts with specific investment companies, your selection of funds may be limited. Since mutual funds are required to disclose how they invest money, it should be a straightforward task to assemble a list of mutual funds that you can invest in and the asset class they belong to.

If you make your own investment decisions, getting a list of available mutual funds that meet your asset-class definitions can be done online at a site like *http://www.morningstar.com* (though they break things down even finer than we did here). You also can take advantage of other specialized financial planning software that allows you to search though all the available mutual funds by many different criteria.

Once we have determined, using our allocation, that we want to invest 22% of our retirement money in domestic large-cap stocks, we can narrow our options in that class by ratings, costs, fees, returns, risks, or whatever other performance categories we come up with. In my experience, once the asset class is identified, it is relatively easy to find several acceptable funds that cover that class.

By way of example, I went to Morningstar's Web site and, using their free mutual fund selector, I did a search for domestic large-cap growth stock mutual funds. Here are the criteria I used (all pull-down selections on the search page):

Minimum purchase $10,000 or less
3-year return greater than category average
5-year return greater than category average
10-year return greater than category average
5-star rating only (Morningstar rating)

In less than a minute, I had a list of 13 mutual funds meeting these qualifications. While it will take much longer to pick the fund(s) we actually want to sink money into, it isn't an unbounded search.

Discipline, Not Emotion

If picking a list of possible investments off the Internet in under a minute makes your knees wobble, then that is all the more reason to follow an asset allocation plan. There is a great deal of mystique surrounding the entire investment field, and it is easy to feel overwhelmed by the available choices and the consequences of making a serious mistake.

In truth, such feelings are based on emotion: the desire to make money, the fear of losing money, the worry of mismanaging retirement funds, the dread of making monetary decisions. Emotional investing causes people to buy at the top of the market, hold losing investments, sell at the wrong time, and make other bad choices.

It is much easier to follow a plan, based on our specific needs, that tells us what to do and when to do it. If investing is a hard thing for you to do, then removing the emotion from the decision process should make your investing life much easier.

Trying to find a good mutual fund from the thousands in the investment haystack is hard. An asset allocation plan simplifies this by dividing the entire market into clear asset classes. By listing a dollar amount for each class, you are given an easy-to-follow financial guide. From there, it quickly gets down to picking one out of a selected list of mutual funds. I find this task much more manageable, especially if all I have to do is to make my choice from a set of highly rated funds.

Selling Your Winners and Buying Your Losers

Not counting the CDs and money market account, our sample asset allocation calls for investment in eight different asset classes. After we make our investment decisions, we might actually put money into 10 or more different mutual funds. We don't know which of these will make us money and which ones won't. If our assumptions are correct (namely, that historical trends will continue on into the future), it won't matter because, over time, this mix of investments will earn us the return we are seeking.

To better demonstrate how this works, let's suppose we need 9% to meet our retirement goals. Using the asset allocation described above, our return might break down like this:

Cash (5%)
3% Money market account (4% return)
2% CDs (5% return)

Stocks (55%)
22% Domestic large-cap stocks (10% return)
17% Domestic small-cap stocks (12% return)
16% International stocks (14% return)

Bonds (40%)
15% Domestic corporate long-term bonds (7% return)
8% Junk bonds (12% return)
7% Domestic corporate intermediate-term bonds (5% return)
5% International bonds (11% return)
5% Domestic government long-term bonds (6% return)

One of the hardest things about investing is handling the emotional issues of committing money to an uncertain future.

Any deviation from the returns assumed here will alter the percentages called for by our asset allocation. Since our asset allocation's percentages are designed to minimize risk for our selected return, we will either begin to take more risk than necessary or get a lower expected return. In order to keep on using our allocation, we need to rebalance our investments back to the allocation's percentages every so often. This means selling some shares that did well and buying those that did poorly.

A numerical example may help here. Our asset allocation calls for 22% of our retirement assets in domestic large-cap stocks. One year after putting our money into the market, it turns out that large-cap stocks had a better than 10% return and this class now accounts for 27% of our retirement assets. Further, let's say that it was a bad year for long-term bonds and we now have only 10% of our assets in domestic corporate long-term bonds. To get back into balance, we would sell our large-cap stock funds and buy our domestic corporate long-term bond funds, so that our portfolio again holds the percentages listed in our allocation.

Here is where emotion can get in the way. Most people would advise you to hold those funds that are doing well and get rid of those that are doing poorly. The goal of such a strategy is to maximize return regardless of risk. This is not what we are trying to do.

In order to maintain our asset allocation and minimize risk, we need to do exactly the opposite. We sell those asset classes that have done well and, therefore, now constitute a larger percentage than they should. At the same time, we buy more of our losing investments. This not only requires discipline but the fortitude to withstand the jibes of others who, no doubt, have made a killing in the stock market following their own advice!

How often should you rebalance? There is no generally accepted amount of time. Some people suggest every three months; others say once a year is plenty. I prefer balancing once a year, on my birthday. I do this simply to limit the time and effort I spend worrying about and fussing with my investments. A different time frame might work better for you.

One of the hardest things about investing is handling the emotional issues of committing money to an uncertain future and the roller-coaster of watching your retirement funds rise and fall in value. Asset allocation investing aims to replace emotion with a disciplined strategy based on the assumption that, in the long run, things will move along historical trends.

Sometimes it is hard to focus on the long-term when your balance is reduced by thousands of dollars in a single day. In the short-term, things might be great (dot-com boom) or horrible (dot-com crash). However, by following an investment discipline, you should be able to avoid making bad investments in the heat of the moment.

Following an asset allocation is no guarantee that you will meet your retirement goals or that everything will go according to plan. After all, risk is involved and things might not work out. The good news is that you can always choose to alter your allocation to earn more return or take less risk if the future begins to turn out differently than you hoped. The best reason to follow an asset allocation is that you have a specific plan to implement and a disciplined way to handle the risk of investing. In my experience, this not only reduces the stress of investing but also provides specific directions and dollar amounts to guide one's decisions.

the bookworm

by Peter H. Salus

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

peter@netpedant.com

This will be an "all over the map" column. I've been reading and thinking about a variety of things.

For example, back in 1985 Peter Capek organized a session on "UNIX on Big Iron." At that time, UNIX had moved from being DEC-only to the Interdata and the early SUN machines. But I certainly never thought about UNIX at IBM – even though I worked at IBM Research and knew Peter.

More recently, IBM has moved into Linux, and so here's Eilert et al. giving us a fine volume concerning Linux on the zSeries and the S/390.

Most interestingly, Linux runs on the zSeries as z/VM, taking us back to the late '70s as well. Let's try to recall that the various software tools ran on Joe Sventek's VM. And, of course, those of us running on IBM iron in the '70s and '80s ran on virtual machines.

Nostalgia aside, *Linux on the Mainframe* is an interesting and valuable book produced by a group at Boblingen and in Poughkeepsie. There is a good bibliography of IBM "red books" and Web-available papers. The references to actual books are the sole notable weak spot.

Security

In a sudden burst, I received several more books on security. Outside of the second edition of Kaufman et al. (*Network Security: Private Communication in a Public World*, 2nd Ed., Prentice-Hall), I really don't think they're worth a great deal.

For example, Yeo's *Personal Firewalls* cites the 1995 [!] edition of Chapman and Zwicky, but nowhere even mentions Cheswick and Bellovin or Garfinkel and Spafford.

Day's book is "written in a manner that anyone with the most basic IT knowledge will be able to read it." Possibly so. It was so simplistic that I gave up on it. Garfinkel and Spafford is the sole technical book in the "Recommended Reading."

Linda McCarthy's book is supposed to give the "big picture" of IT security. It's so big, I missed the trees. The best parts of the book are Spafford's "Foreword" and the 40-page Appendix.

Java

On the other hand, there is a trio of really good Java books from O'Reilly, *Java Data Objects*, *Head-First Java*, and *Java Database Best Practices*. I've been coming across instances where Oracle-fits-all just isn't an appropriate solution. JDO lets you manage data without concerning yourself with db software or db query languages. Hey, you don't need SQL and you don't need to copy stuff using JDBC calls. It looks (to me) like a fine API; Jordan and Russell have written a fine book.

Sierra and Bates have written an exciting book that won't be for everyone. It's a quick read, a "contemporary" sort of learning experience, that incorporates humor and puzzles into instruction. This may be the best book on learning Java I've seen.

In *Java Database Best Practices,* Reese runs through the myriad APIs and technologies – EJB, JDO, JDBC, SQL, RDBMS, OODBMS, and more. He explains the various approaches and

BOOKS REVIEWED IN THIS COLUMN

LINUX ON THE MAINFRAME
John Eilert, et al.
Upper Saddle River, NJ: Prentice Hall, 2003. Pp. 434. ISBN 0-13-101415-3.

PERSONAL FIREWALLS
Lisa Yeo
Upper Saddle River, NJ: Prentice Hall, 2003. Pp. 216. ISBN 0-13-046222-5.

INSIDE THE SECURITY MIND
Kevin Day
Upper Saddle River, NJ: Prentice Hall, 2003. Pp. 309. ISBN 0-13-111829-3.

IT SECURITY
Linda McCarthy
Upper Saddle River, NJ: Prentice Hall, 2003. Pp. 246. ISBN 0-13-101112-X.

JAVA DATA OBJECTS
David Jordan & Craig Russell
Sebastopol, CA: O'Reilly, 2003. Pp. 356. ISBN 0-596-00276-9.

HEAD FIRST JAVA
Kathy Sierra and Bert Bates
Sebastopol, CA: O'Reilly, 2003. Pp. 619. ISBN 0-596-00465-6.

JAVA DATABASE BEST PRACTICES
George Reese
Sebastopol, CA: 2003. Pp. 267. ISBN 0-596-00522-9.

IP STORAGE NETWORKING
Gary Orenstein
Boston: Addison-Wesley, 2003. ISBN 0-321-15960-8.

UNIX SYSTEMS PROGRAMMING: COMMUNICATIONS, CONCURRENCY AND THREADS, SECOND EDITION
Kay A. Robbins and Steve Robbins
Upper Saddle River, NJ: Prentice Hall, 2003. Pp. 912. ISBN 0-130-42411-0

gives all the information necessary to assess just which approach should be the most effective.

Storage

Orenstein's *IP Storage Networking* is a fascinating, lucid exposition of a very complex problem. I'm confident that more and more of us will be employing iSCSI. This book will be leading us in the developing processes.

A Reappearance

Robbins and Robbins is far more than merely a new edition of *Practical UNIX Programming* (1995). In addition to putting on pounds, it has gained in topics. It's now *UNIX Systems Programming*. As I was sent galleys, I can't tell exactly how many pages the finished book will have (nor its ISBN). And while it may never replace Rich Stevens' book in my affections, it is certainly more up-to-date.

Twenty-Five Years Ago in UNIX

by Peter H. Salus

Some of you may recall reading in *UNIX NEWS* that the long-announced UNIX issue of the *BSTJ* was available. Only $2!

It was the July-August 1978 issue. A full quarter-century ago.

I still consider the pale blue issue of the *BSTJ* the very best "book" on UNIX. I think the 21 papers are outstanding. Doug McIlroy on the beginnings; Dennis and Ken on "The UNIX Time-Sharing System"; Ken on implementation; Steve Bourne on the shell; Dennis, Brian, Steve Johnson, and Mike Lesk on C; Johnson and Ritchie on C portability; Heinz Lyclama on MERT; Ted Dolotta, Dick Haight, and John Mashey on "The Programmer's Workbench"; and so on.

Prentice-Hall reprinted the issue in 1987, together with the second *BSTJ* "UNIX" issue. I saw a two-volume set in a bookstore for nearly $300 recently. I presume Lucent now owns the copyright. Republish this, someone!

Browsing this old friend, I was repeatedly astounded at how relevant the articles still were.

No, Heinz, I'm not going to use MERT; no, Ted, not PWB either.

But there are about 400 pages of really first-rate ideas here. They are ideas, for the most part, equally applicable to OS X or any flavor of Linux.

(Maybe that's SCO's point in their silly lawsuit: Those ideas which were important to UNIX are the same ideas now used in the BSDs, in Linux, in OS X. Look out, Shakespeare! Your sonnets have the look and feel of Petrarch's. If history repeats itself as farce, then SCO is really into clown-paint. For a fine analysis, I recommend Eric Raymond's: *http://www.opensource.org/sco-vs-ibm.html*.

And a 20-year note . . .

In January 1983, in San Diego, Armando Stettner introduced the "UNIX" license plate. Armando, it's still up on my wall. Hey! SCO! "Live Free or Die!"

Standards Reports

What's New in Technical Corrigendum Number 1 for IEEE Std 1003.1-2001

by Andrew Josey
Andrew Josey is the director, server platforms, for The Open Group in Reading, England, and the chair of the Austin Group, Inc.

a.josey@opengroup.org

The IEEE-SA Standards Board approved Technical Corrigendum Number 1 (TC1) on 10th December 2002. The governing board of The Open Group approved the document on 7 February 2003.

The following article presents an overview of what has been changed by this corrigendum.

Issues Related to the Base Definitions

GLOB.H

Technical Corrigendum Number 1 item XBD/TC1/D6/8 is applied, correcting the glob() prototype definition by removing the restrict qualifier from the function pointer argument.

LANGINFO.H

Technical Corrigendum Number 1 item XBD/TC1/D6/9 is applied, adding a sentence to the "Meaning" column entry for the CRNCYSTR constant. This change is to accommodate historic practice.

LIMITS.H

Technical Corrigendum Number 1 item XBD/TC1/D6/10 is applied, updating the value of _POSIX_CHILD_MAX from 6 to 25. This corrects an editorial error and is for FIPS 151-2 alignment.

NETDB.H

Technical Corrigendum Number 1 item XBD/TC1/D6/11 is applied, adding a description of the NI_NUMERICSCOPE macro and correcting the getnameinfo() function prototype. These changes are for alignment with the IETF IPv6 specification.

NETINET/IN.H

Technical Corrigendum Number 1 item XBD/TC1/D6/12 is applied, adding "const" qualifiers to the in6addr_any and in6addr_loopback external variables.

PTHREAD.H

Technical Corrigendum Number 1 item XBD/TC1/D6/13 is applied, correcting shading errors that were in contradiction with the System Interfaces Volume.

SIGNAL.H

Technical Corrigendum Number 1 item XBD/TC1/D6/14 is applied, changing the descriptive text for members of struct sigaction. Technical Corrigendum Number 1 item XBD/TC1/D6/15 is applied, correcting the definition of the sa_sigaction member of struct sigaction.

SYS/MMAN.H

Technical Corrigendum Number 1 item XBD/TC1/D6/16 is applied, correcting margin code and shading errors for the mlock() and munlock() functions.

SYS/STAT.H

Technical Corrigendum Number 1 item XBD/TC1/D6/17 is applied, adding text regarding the st_blocks members of the stat structure to the RATIONALE.

SYS/STATVFS.H

Technical Corrigendum Number 1 item XBD/TC1/D6/18 is applied, changing the description of ST_NOSUID from "Does not support setuid()/setgid() semantics" to "Does not support the semantics of the ST_ISUID and ST_ISGID file mode bits".

TERMIOS.H

Technical Corrigendum Number 1 item XBD/TC1/D6/19 is applied, changing ECHOK to ECHOKE in the APPLICATION USAGE section.

UNISTD.H

Technical Corrigendum Number 1 item XBD/TC1/D6/2 is applied, changing "Thread Stack Address Size" to "Thread Stack Size Attribute".

Technical Corrigendum Number 1 item XBD/TC1/D6/20 is applied, adding the _POSIX_IPV6, _SC_V6, and _SC_RAW_SOCKETS symbols.

Technical Corrigendum Number 1 item XBD/TC1/D6/21 is applied, correcting the description in "Constants for Functions" for the _CS_POSIX_V6_LP64_OFF64_CFLAGS, _CS_POSIX_V6_LP64_OFF64_LDFLAGS, and _CS_POSIX_V6_LP64_OFF64_LIBS symbols.

Technical Corrigendum Number 1 item XBD/TC1/D6/22 is applied, removing the shading for the _PC* and _SC* constants, since these are mandatory upon all implementations.

Technical Corrigendum Number 1 item XBD/TC1/D6/23 is applied, adding the _PC_SYMLINK_MAX and _SC_SYMLOOP_MAX constants.

Technical Corrigendum Number 1 item XBD/TC1/D6/24 is applied, correcting the shading and margin code for the fsync() function.

Technical Corrigendum Number 1 item XBD/TC1/D6/25 is applied, adding the following text to APPLICATION USAGE: "New applications should not use _XOPEN_SHM or _XOPEN_ENH_I18N".

WCHAR.H

Technical Corrigendum Number 1 item XBD/TC1/D6/26 is applied, adding the APPLICATION USAGE section.

Rationale Changes Related to the Base Definitions

A.4.10

Add to end of A.4.10, Memory Synchronization p37 l 1465.

Technical Corrigendum Number 1 item XBD/TC1/D6/4 is applied, adding a new paragraph beneath the table of functions: "The pthread_once() function shall synchronize memory for the first call in each thread for a given pthread_once_t object."

A.7.3.3 LC_MONETARY

Line 1986, add after "Technical Corrigendum No. 1", "item XBD/TC1/D6/6."

Add another paragraph at end of this section:

Technical Corrigendum Number 1 item XBD/TC1/D6/5 is applied, adding the int_[np]_* values to the POSIX locale definition of LC_MONETARY.

A.8.3 TZ

Add to the end of the TZ section (line 2339).

Technical Corrigendum Number 1 item XBD/TC1/D6/7 is applied, adding the ctime_r() and localtime_r() functions to the list of functions that use the TZ environment variable.

Issues Related to the System Interfaces

ABORT

Technical Corrigendum Number 1 item XSH/TC1/D6/10 is applied, changing the DESCRIPTION of abnormal termination processing and adding to the RATIONALE section.

BSEARCH

Technical Corrigendum Number 1 item XSH/TC1/D6/11 is applied, adding to the last sentence at the end of the first non-shaded paragraph in the DESCRIPTION and adding the three following paragraphs. The RATIONALE section is also updated. These changes are for alignment with the ISO C standard.

CLOSE

Technical Corrigendum Number 1 item XSH/TC1/D6/12 is applied, correcting the XSI shaded text relating to the master side of a pseudo-terminal. The reason for the change is that the behavior of pseudo-terminals and regular terminals should be as much alike as possible in this case; the change achieves that and matches historical behavior.

CLOSELOG

Technical Corrigendum Number 1 item XSH/TC1/D6/13 is applied, correcting the EXAMPLES.

DLSYM

Technical Corrigendum Number 1 item XSH/TC1/D6/14 is applied, correcting an example and adding text to the RATIONALE describing issues related to conversion of pointers to functions and back again.

EXEC

Technical Corrigendum Number 1 item XSH/TC1/D6/15 is applied, adding a new paragraph to the DESCRIPTION and text to the end of the APPLICATION USAGE section. This change addresses a security concern, where implementations may want to reopen file descriptors 0, 1, and 2 for programs with the set-user-id or set-group-id file mode bits calling the exec family of functions.

EXIT

Technical Corrigendum Number 1 item XSH/TC1/D6/16 is applied, correcting grammar in the description.

FORK

Technical Corrigendum Number 1 item XSH/TC1/D6/17 is applied, adding text to the DESCRIPTION and RATIONALE relating to fork handlers registered by the pthread_atfork() function and async-signal safety.

FPATHCONF

Technical Corrigendum Number 1 item XSH/TC1/D6/18 is applied, changing the fourth paragraph of the DESCRIPTION and removing shading and margin markers from the table. This change is needed since implementations are required to support all these symbols.

FREEADDRINFO

Technical Corrigendum Number 1 item XSH/TC1/D6/19 is applied, adding three notes to the DESCRIPTION and adding text to the APPLICATION USAGE related to the term canonical name. A reference to RFC 2181 is also added to the informative references front matter.

Technical Corrigendum Number 1 item XSH/TC1/D6/20 is applied, making changes for alignment with the IETF IPv6 API specification. These include the following: adding AI_V4MAPPED, AI_ALL, and AI_ADDRCONFIG to the allowed values for the ai_flags field; adding a description of AI_ADDRCONFIG; and adding a description of the consequences of ignoring the AI_PASSIVE flag.

FSETPOS

Technical Corrigendum Number 1 item XSH/TC1/D6/21 is applied, deleting an erroneous EINVAL error case from the ERRORS section.

GAI_STRERROR

Technical Corrigendum Number 1 item XSH/TC1/D6/22 is applied, adding the EAI_OVERFLOW error code.

GETNAMEINFO

Technical Corrigendum Number 1 item XSH/TC1/D6/23 is applied, making various changes in the SYNOPSIS and DESCRIPTION for alignment with the IETF IPv6 specification.

Technical Corrigendum Number 1 item XSH/TC1/D6/24 is applied, adding the EAI_OVERFLOW error to the ERRORS section.

GETRLIMIT

Technical Corrigendum Number 1 item XSH/TC1/D6/25 is applied, changing wording for RLIMIT_NOFILE in the DESCRIPTION related to functions that allocate a file descriptor failing with [EMFILE]. Text is added to the APPLICATION USAGE section noting the consequences of a process attempting to set the hard or soft limit for RLIMIT_NOFILE less than the highest currently open file descriptor+1.

GETSUBOPT

Technical Corrigendum Number 1 item XSH/TC1/D6/26 is applied, correcting an editorial error in the SYNOPSIS.

GMTIME

Technical Corrigendum Number 1 item XSH/TC1/D6/27 is applied, adding the EOVERFLOW error case.

IF_INDEXTONAME

Technical Corrigendum Number 1 item XSH/TC1/D6/28 is applied, changing {IFNAMSIZ} to {IF_NAMESIZ} in the DESCRIPTION.

INET_NTOP

Technical Corrigendum Number 1 item XSH/TC1/D6/29 is applied, adding "the address must be in network byte order" to the end of the fourth sentence of the first paragraph in the DESCRIPTION.

INITSTATE

Technical Corrigendum Number 1 item XSH/TC1/D6/30 is applied, removing rand_r() from the list of suggested functions in the APPLICATION USAGE section.

LOCALECONV

Technical Corrigendum Number 1 item XSH/TC1/D6/31 is applied, removing references to "int_curr_symbol" and updating the descriptions of p_sep_by_space and n_sep_by_space. These changes are for alignment with the ISO C standard.

LOCALTIME

Technical Corrigendum Number 1 item XSH/TC1/D6/32 is applied, adding the EOVERFLOW error case.

MAKECONTEXT

Technical Corrigendum Number 1 item XSH/TC1/D6/33 is applied, clarifying that the arguments passed to func are of type int.

MMAP
Technical Corrigendum Number 1 item XSH/TC1/D6/34 is applied, changing the margin code in the SYNOPSIS from MF|SHM to MC3 (notation for MF|SHM|TYM).

MODF
Technical Corrigendum Number 1 item XSH/TC1/D6/35 is applied, correcting the code example in the APPLICATION USAGE.

MUNMAP
Technical Corrigendum Number 1 item XSH/TC1/D6/36 is applied, changing the margin code in the SYNOPSIS from MF|SHM to MC3 (notation for MF|SHM|TYM).

NANOSLEEP
Technical Corrigendum Number 1 item XSH/TC1/D6/37 is applied, updating the SEE ALSO to include the clock_nanosleep() function.

POW
Technical Corrigendum Number 1 item XSH/TC1/D6/42 is applied, correcting the third paragraph in the RETURN VALUE section.

PTHREAD_ATTR_GETSTACKSIZE
Technical Corrigendum Number 1 item XSH/TC1/D6/43 is applied, correcting the margin code in the SYNOPSIS from TSA to TSS and updating the CHANGE HISTORY from "Thread Stack Address Attribute option" to "Thread Stack Size Attribute option."

PTHREAD_CREATE
Technical Corrigendum Number 1 item XSH/TC1/D6/44 is applied, adding text that the alternate stack is not inherited.

PTHREAD_RWLOCK_DESTROY
Technical Corrigendum Number 1 item XSH/TC1/D6/45 is applied, adding APPLICATION USAGE relating to priority inversion.

PUTENV
Technical Corrigendum Number 1 item XSH/TC1/D6/48 is applied, clarifying wording in the DESCRIPTION and adding a new paragraph into APPLICATION USAGE referring readers to exec.

QSORT
Technical Corrigendum Number 1 item XSH/TC1/D6/49 is applied, adding to the last sentence to the end of the first non-shaded paragraph in the DESCRIPTION and adding the two following paragraphs. The RATIONALE section is also updated. These changes are for alignment with the ISO C standard.

READDIR
Technical Corrigendum Number 1 item XSH/TC1/D6/50 is applied, replacing the EXAMPLES section with a new example.

REALPATH
Technical Corrigendum Number 1 item XSH/TC1/D6/51 is applied, adding new text to the DESCRIPTION for the case when resolved_name is a null pointer, changing the EINVAL error case text, adding RATIONALE text, and the FUTURE DIRECTIONS text.

SCHED_GET_PRIORITY_MAX
Technical Corrigendum Number 1 item XSH/TC1/D6/52 is applied, changing the PS margin code in the SYNOPSIS to PS|TPS.

SCHED_RR_GET_INTERVAL
Technical Corrigendum Number 1 item XSH/TC1/D6/53 is applied, changing the PS margin code in the SYNOPSIS to PS|TPS.

SEM_GETVALUE
Technical Corrigendum Number 1 item XSH/TC1/D6/54 is applied.

SETENV
Technical Corrigendum Number 1 item XSH/TC1/D6/55 is applied, adding references to exec in the APPLICATION USAGE and SEE ALSO sections.

SETPGID
Technical Corrigendum Number 1 item XSH/TC1/D6/56 is applied, changing the wording in the DESCRIPTION from "the process group ID of the indicated process shall be used" to "the process ID of the indicated process shall be used." This change reverts the wording to as in IEEE Std 1003.1-1996; it appeared to be an unintentional change.

SIGACTION
Technical Corrigendum Number 1 item XSH/TC1/D6/57 is applied, changing descriptive text in the table describing the sigaction structure.

SIGALTSTACK
Technical Corrigendum Number 1 item XSH/TC1/D6/58 is applied, updating the first sentence to include "<Q>for the current thread</Q>" at the end.

SIGINTERRUPT
Technical Corrigendum Number 1 item XSH/TC1/D6/59 is applied, correcting the declaration in the sample implementation given in the DESCRIPTION section.

STRFTIME
Technical Corrigendum Number 1 item XSH/TC1/D6/60 is applied.

STRTOD
Technical Corrigendum Number 1 item XSH/TC1/D6/61 is applied, correcting the second paragraph in the RETURN VALUE section. This change makes it clear the sign of the return value.

SYSCONF
Technical Corrigendum Number 1 item XSH/TC1/D6/62 is applied, updating the DESCRIPTION to denote that the _PC* and _SC* symbols are now required to be supported. A corresponding change has been made in the Base Definitions volume. The deletion in the second paragraph removes some duplicated text. The additions add some symbols drawn from the standard that were accidentally omitted from this page.

Technical Corrigendum Number 1 item XSH/TC1/D6/63 is applied, making it clear in the RETURN VALUE that the value returned for sysconf(_SC_OPEN_MAX) may change if a call to setrlimit() adjusts the RLIMIT_NOFILE soft limit.

TAN
Technical Corrigendum Number 1 item XSH/TC1/D6/64 is applied, correcting the last paragraph in the RETURN VALUE section.

TGAMMA
Technical Corrigendum Number 1 item XSH/TC1/D6/65 is applied, correcting the third paragraph in the RETURN VALUE section.

WCSTOD
Technical Corrigendum Number 1 item XSH/TC1/D6/66 is applied, correcting the second paragraph in the RETURN VALUE section.

Rationale Changes Related to the System Interfaces

B.2.2.2
Add to end of B.2.2.2

Technical Corrigendum Number 1 item XSH/TC1/D6/2 is applied, deleting the entries POSIX_, _POSIX_, and posix_ from the column of allowed namespace prefixes for use by an implementation in the first table. The presence of these prefixes was contradicting later text that states "The pre-

fixes posix_, POSIX_, and _POSIX are reserved for use by IEEE Std 1003.1-2001 and other POSIX standards. Implementations may add symbols to the headers shown in the following table, provided the identifiers . . . do not use the reserved prefixes posix_, POSIX_, or _POSIX."

Technical Corrigendum Number 1 item XSH/TC1/D6/3 is applied, correcting the reserved macro prefix from "PRI[a-z],SCN[a-z]" to "PRI[Xa-z],SCN[Xa-z]" in the second table. The change was needed since the C Standard allows implementations to define macros of the form "PRI" or "SCN" followed by any lowercase letter or "X" in <inttypes.h> (ISO/IEC 9899:1999 P400, Sub-clause 7.26.4.).

Technical Corrigendum Number 1 item XSH/TC1/D6/4 is applied, adding a new section listing reserved names for the <stdint.h> header. This change was for alignment with the C standard.

B.2.4.3
Add to the end of B2.4.3.

Technical Corrigendum Number 1 item XSH/TC1/D6/5 is applied, reordering the RTS shaded text under the third and fourth paragraphs of the SIG_DFL description. This corrects an earlier editorial error in this section.

Technical Corrigendum Number 1 item XSH/TC1/D6/6 is applied, adding the abort() function to the list of async-cancel-safe functions.

B.2.8.3
Add new paragraph 2 before "Memory Locking" in 2.8.3.

Technical Corrigendum Number 1 item XSH/TC1/D6/7 is applied, correcting the shading and margin markers in the introduction to section 2.8.3.1.

B.2.9.5
Add to the end of B.2.9.5.

Technical Corrigendum Number 1 item XSH/TC1/D6/8 is applied, adding the pselect() function to the list of functions with Cancellation points.

Issues Related to Shell and Utilities

BREAK, COLON, CONTINUE, DOT, EVAL, EXEC, EXIT, EXPORT, READONLY, RETURN, SET, SHIFT, TRAP, UNSET
Technical Corrigendum Number 1 item XCU/TC1/D6/5 is applied, so that the manual page sections use terms as described in the Utility Description Defaults. No change in behavior is intended.

EXPORT
Technical Corrigendum Number 1 item XCU/TC1/D6/6 is applied, adding the following text to the end of the first paragraph of the DESCRIPTION:

> "If the name of a variable is followed by =word, then the value of that variable shall be set to *word*."

The reason for this change was that the SYNOPSIS for export includes export name[=word]. . . but the meaning of the optional "=word" is never explained in the text.

READONLY
Technical Corrigendum Number 1 item XCU/TC1/D6/7 is applied, adding the following text to the end of the first paragraph of the DESCRIPTION:

> "If the name of a variable is followed by =word, then the value of that variable shall be set to *word*."

The reason for this change was that the SYNOPSIS for readonly includes readonly name[=word]. . . but the meaning of the optional "=word" is never explained in the text.

SET
Technical Corrigendum Number 1 item XCU/TC1/D6/8 is applied, changing the square brackets in the example in RATIONALE to be in bold which is the typeface used for optional items.

TIMES
Technical Corrigendum Number 1 item XCU/TC1/D6/9 is applied,

changing text in the DESCRIPTION from:

> "Write the accumulated user and system times for the shell and for all of its child processes . . ."

to:

> "The times utility shall write the accumulated user and system times for the shell and for all of its child processes . . ."

AR
Technical Corrigendum Number 1 item XCU/TC1/D6/10 is applied, making corrections to the SYNOPSIS. The change was needed since the -a, -b, and -i options are mutually exclusive, and posname is required if any of these options is specified.

Technical Corrigendum Number 1 item XCU/TC1/D6/11 is applied, correcting the description of the two-byte trailer in RATIONALE that had missed out a back quote. The correct trailer is a back quote followed by a <newline>.

C99
Technical Corrigendum Number 1 item XCU/TC1/D6/12 is applied, correcting the EXTENDED DESCRIPTION section of -l c and -l m. Previously the text did not take into account the presence of the c99 math headers.

Technical Corrigendum Number 1 item XCU/TC1/D6/13 is applied, changing the reference to the libxnet library to libxnet.a.

CD
Technical Corrigendum Number 1 item XCU/TC1/D6/14 is applied, changing the SYNOPSIS to make it clear that the -L and -P options are mutually exclusive.

CHGRP
Technical Corrigendum Number 1 item XCU/TC1/D6/15 is applied, changing the SYNOPSIS to make it clear that the -h and -R options are optional.

CHMOD
Technical Corrigendum Number 1 item XCU/TC1/D6/16 is applied, changing XSI shaded text in the EXTENDED DESCRIPTION from:

> "The perm symbol t shall specify the S_ISVTX bit and shall apply to directories only. The effect when using it with any other file type is unspecified. It can be used with the who symbols o, a, or with no who symbol. It shall not be an error to specify a who symbol of u or g in conjunction with the perm symbol t; it shall be ignored for u and g."

to:

> "The perm symbol t shall specify the S_ISVTX bit. When used with a file of type directory, it can be used with the who symbol a, or with no who symbol. It shall not be an error to specify a who symbol of u, g, or o in conjunction with the perm symbol t, but the meaning of these combinations is unspecified. The effect when using the perm symbol t with any file type other than directory is unspecified."

This change is to permit historical behavior.

CHOWN
Technical Corrigendum Number 1 item XCU/TC1/D6/17 is applied, changing the SYNOPSIS to make it clear that the -h and -R options are optional.

CP
Technical Corrigendum Number 1 item XCU/TC1/D6/18 is applied, correcting an error in the SEE ALSO section.

DATE
Technical Corrigendum Number 1 item XCU/TC1/D6/19 is applied, correcting the CHANGE HISTORY section.

DIFF
Technical Corrigendum Number 1 item XCU/TC1/D6/20 is applied, changing the STDOUT section. This changes the specification of "diff -c" so it agrees with existing practice when contexts contain zero lines or one line.

ECHO
Technical Corrigendum Number 1 item XCU/TC1/D6/21 is applied, so that the echo utility can accommodate historical BSD behavior.

ED
Technical Corrigendum Number 1 item XCU/TC1/D6/22 is applied, adding the text "Any line modified by the command list shall be unmarked." to the G command. This change corresponds to a similar change made to the g command in the 2001 revision.

EX
Technical Corrigendum Number 1 item XCU/TC1/D6/23 is applied, correcting a URL.

FALSE
Technical Corrigendum Number 1 item XCU/TC1/D6/24 is applied, changing the STDERR section from "None" to "Not Used" for alignment with the Utility Description Defaults.

FILE
Technical Corrigendum Number 1 item XCU/TC1/D6/25 is applied, making major changes to address ambiguities raised in defect reports.

Technical Corrigendum Number 1 item XCU/TC1/D6/26 is applied, making it clear in the OPTIONS section that the -m, -d, and -M options do not comply with Guideline 11 of the utility Syntax Guidelines.

GETCONF
Technical Corrigendum Number 1 item XCU/TC1/D6/27 is applied, correcting the descriptions of path_var and system_var in the OPERANDS section.

GREP
Technical Corrigendum Number 1 item XCU/TC1/D6/28 is applied, correcting the examples using the grep -F option that did not match the normative description of the -F option.

ICONV
Technical Corrigendum Number 1 item XCU/TC1/D6/29 is applied, making changes to address inconsistencies with the iconv() function in the System Interfaces Volume.

LOCALE
Technical Corrigendum Number 1 item XCU/TC1/D6/30 is applied, correcting an editorial error in the STDOUT section.

M4
Technical Corrigendum Number 1 item XCU/TC1/D6/31 is applied, replacing the EXAMPLES section.

MAILX
Technical Corrigendum Number 1 item XCU/TC1/D6/32 is applied, applying a change to the EXTENDED DESCRIPTION, raised by IEEE PASC Interpretation 1003.2-1992 #122, which was overlooked in the revision.

OD
Technical Corrigendum Number 1 item XCU/TC1/D6/33 is applied, correcting the examples, which were using an undefined "-n" option that should have been "-N."

PATCH
Technical Corrigendum Number 1 item XCU/TC1/D6/34 is applied, clarifying the way that the patch utility performs ifdef selection for the -D option.

PAX
Technical Corrigendum Number 1 item XCU/TC1/D6/35 is applied. This change, which adds the process ID of the pax process into certain fields, provides a method for the implementation to ensure that different instances of pax extracting a file named "/a/b/foo" will not collide when processing the extended header information associated with "foo."

Technical Corrigendum Number 1 item XCU/TC1/D6/36 is applied, changing "-x B" to "-x pax" in the OPTIONS section.

STTY
Technical Corrigendum Number 1 item XCU/TC1/D6/37 is applied, applying IEEE PASC Interpretation 1003.2-1992 #133, fixing an error in the description of "stty nl.".

TEST
Technical Corrigendum Number 1 item XCU/TC1/D6/38 is applied, XSI margin marking and shading a line in the OPERANDS section referring to the use of parentheses as arguments to the test utility.

TRUE
Technical Corrigendum Number 1 item XCU/TC1/D6/39 is applied, replacing the terms "None" and "Default" from the STDERR and EXIT STATUS section with terms as defined in the Utility Description Defaults section.

UNIQ
Technical Corrigendum Number 1 item XCU/TC1/D6/40 is applied, adding LC_COLLATE to the ENVIRONMENT VARIABLES section, and changing "the application shall ensure that" in the OUTPUT FILES section.

VI
Technical Corrigendum Number 1 item XCU/TC1/D6/41 is applied, adding "[count]" to the Synopsis for "[[."

Technical Corrigendum Number 1 item XCU/TC1/D6/42 is applied, adding "[count]" to the Synopsis for "]]."

Rationale Changes Related to the Shell and Utilities

XRAT SECTION C.1.9 UTILITY LIMITS
Add to the end of C.1.9.

Technical Corrigendum Number 1 item XCU/TC1/D6/2 is applied, deleting the entry for {POSIX2_VERSION} since it is not a Utility Limit Minimum Value.

Technical Corrigendum Number 1 item XCU/TC1/D6/3 is applied, changing the text in Utility Limits from:

> "utility (see getconf (on page 481)) and through the sysconf() function defined in the System Interfaces volume of IEEE Std 1003.1-2001. The literal names shown in Table 1-3 (on page 17) apply only to the getconf utility; the high-level language binding describes the

exact form of each name to be used by the interfaces in that binding."

to:

"utility (see getconf (on page 481))."

c.

Add to the end of C.2.6.3

Technical Corrigendum Number 1 item XCU/TC1/D6/4 is applied, changing the text from:

"If a command substitution occurs inside double-quotes, it shall not be performed on the results of the substitution."

to:

"If a command substitution occurs inside double-quotes, field splitting and pathname expansion shall not be performed on the results of the substitution."

The replacement text taken from POSIX.2-1992 is clearer about the items that are not performed.

Austin Group Status Update
April 15, 2003

by Andrew Josey

a.josey@opengroup.org

Since the last status update, we are pleased to report Technical Corrigendum 1 to the Austin Group Specifications has been approved by all the sponsoring bodies – the IEEE-SA, The Open Group, and ISO/IEC.

The Austin Group has recently published the 2003 edition of its specifications incorporating Technical Corrigendum 1. The designation for this edition is IEEE Std 1003.1, 2003 Edition, ISO/IEC 9945:2003 and form the core of the 2003 edition of the Single UNIX Specification Version 3.

HTML copies of the specification can be freely downloaded or read online at *http://www.unix-systems.org/version3/*. USENIX members who would like a PDF copy should send an email request to Andrew Josey.

Text of Technical Corrigendum 1 (the list of changes to the 2001 edition of the Austin Group specification) is freely available from *http://www.opengroup.org/corrigenda/*.

LSB Certification News 1Q2003
April 15, 2003

by Andrew Josey

a.josey@opengroup.org

The Open Group has certified the following products to the LSB Specifications during 1Q 2003:

Date	Company	Product
06-Jan-03	Red Hat, Inc,	Red Hat Linux Advanced Server 2.1 with updates
07-Jan-03	Sun Wah Linux Ltd	Sun Wah Linux Desktop 3.0
07-Jan-03	Turbo Linux Inc	Turbolinux Enterprise Server 8 powered by UnitedLinux
15-Jan-03	Conectiva Inc.	Conectiva Linux Enterprise Edition Powered by UnitedLinux v1.0
24-Mar-03	SuSE Linux AG	UnitedLinux 1.0
24-Mar-03	SuSE Linux AG	SuSE Linux 8.2
28-Mar-03	SuSE Linux AG	SuSE Linux Enterprise Server 8 for IPF powered by UnitedLinux
01-Apr-03	Red Hat, Inc.	Red Hat Linux 9

As of April 15, 2003, there are nineteen LSB certified products.

The full register of certified products is available at *http://www.opengroup.org/lsb/cert/register.html.*

For more information LSB Certification, please see *http://www.opengroup.org/lsb/cert/*.

Standards Briefing: The Linux Standard Base (LSB)
April 15, 2003

by Andrew Josey

a.josey@opengroup.org

In this article we introduce the Linux Standard Base, the specification, and certification programs.

The LSB Specification

The Linux Standard Base (LSB) Specification is an application binary interface standard for shrink-wrapped applications. The LSB draws on the source standards of IEEE POSIX 1003.1-1990 and The Open Group Single UNIX Specification Version 2 for

many of its interfaces, although it does not formally defer to them, preferring to document any differences where they exist. It also extends the source standards in other areas (such as graphics) and includes the necessary details such as the binary execution file formats to support a high-volume application platform.

Although in theory the LSB is not tied to the GNU/Linux operating system, in practice the binary definitions are tightly coupled to Linux and the GNU C compiler.

The LSB is available as a family of specifications supporting a number of processor architectures, including IA32, PPC32, and IA64. There is a generic specification, common to all the processor architectures, known as the "generic LSB" (or gLSB), and for each processor architecture an architecture-specific specification ("archLSB") describing the details that vary by processor architecture.

The specification is evolving quite rapidly. LSB 1.3, introduced in January 2003, adds internationalization, PAM, packaging, static C++ linking, bug fixes, plus IA64, PPC32, and soon PPC64, S390, S390X, and maybe Hammer. LSB 2.0 is planned for January 2004.

To support the specification, the LSB includes a number of development tools, including test suites, and a set of reference conforming applications. Binary versions of the test suites and reference applications are used for formal LSB certification of runtime environments. All the major Linux vendors today have certified LSB systems.

LSB Certification

The LSB certification program is a voluntary program of the Free Standards Group, open to any product meeting the conformance requirements. It is not restricted to Linux-based systems or Linux-based applications, although in practice it does lean toward requiring use of glibc.

It is a formal process built around a policy document and a trademark license agreement. Suppliers of certified products, warrant and represent that the product meets all the conformance requirements applicable to the class of LSB Certification being certified.

LSB certification currently covers the following specifications:*

- The Linux Standard Base Specification 1.3
- The Linux Standard Base Specification for IA32 1.3
- The Linux Standard Base Specification for PPC32 1.3
- The Linux Standard Base Specification for IA64 1.3
- The OpenI18N Specification (formally the Li18nux 2000 Globalization Specification Version 1.0 with Amendment 4)

*Note that LSB 1.2 certification was withdrawn on April 18 2003.

LSB 1.2, introduced in January 2002, was the first version of the specification to have an equivalent LSB certification program. LSB 1.2 certification, which commenced in July 2002, is limited to the IA32 ABI. LSB 1.3 certification was introduced in January 2003 and adds support for PPC32 and IA64. At the time of writing, there are nineteen runtime environments from nine vendors.

More Information

Detailed information on the LSB is available from *http://www.linuxbase.org*.

Detailed information on the LSB Certification Program is available from the LSB Certification Authority at *http://www.opengroup.org/lsb/cert/*.

The Guide to LSB Certification is available at *http://www.opengroup.org/lsb/cert/docs/LSB_Certification_Guide.html*.

The LSB Certification Register can be viewed at *http://www.opengroup.org/lsb/cert/register.html*.

USENIX news

USENIX MEMBER BENEFITS

As a member of the USENIX Association, you receive the following benefits:

FREE SUBSCRIPTION TO *;login:*, the Association's magazine, published six times a year, featuring technical articles, system administration articles, tips and techniques, practical columns on such topics as security, Tcl, Perl, Java, and operating systems, book reviews, and summaries of sessions at USENIX conferences.

ACCESS TO *;login:* online from October 1997 to last month <*www.usenix.org/publications/login/login.html*>.

ACCESS TO PAPERS from the USENIX Conferences online starting with 1993 <*www.usenix.org/publications/library/index.html*>.

THE RIGHT TO VOTE on matters affecting the Association, its bylaws, election of its directors and officers.

DISCOUNTS on registration fees for all USENIX conferences.

DISCOUNTS on the purchase of proceedings and CD-ROMS from USENIX conferences.

SPECIAL DISCOUNTS on a variety of products, books, software, and periodicals. See <*http://www.usenix.org/membership/specialdisc.html*> for details.

FOR MORE INFORMATION REGARDING MEMBERSHIP OR BENEFITS, PLEASE SEE

<*http://www.usenix.org/membership/membership.html*>

OR CONTACT

<*office@usenix.org*>

Phone: 510 528 8649

2003 USENIX Nominating Committee

The biennial elections of USENIX's Board of Directors will be held in the Spring of 2004. The USENIX Board has appointed Dan Geer to serve as chairman of the Nominating Committee. The composition of this committee and instructions on how to nominate individuals will be published in the October issue of *;login:*.

Summary of the USENIX Board of Directors Meetings

by Ellie Young and Tara Mulligan

ellie@usenix.org
tara@usenix.org

The following is a summary of the actions taken by the USENIX Board of Directors from November 6, 2002 through June 30, 2003.

FINANCES

The Board voted to accept the budget for 2003, which reflected the changes made at the previous meeting as well as revised projections for conference attendance and membership dues.

REGISTRATION FEES

It was agreed that USENIX would offer a discounted conference registration fee for the unemployed. The discount amount will be $195, available only during the pre-registration period. A limited number of discounts would be granted for each conference, on a first come, first serve basis. Attendees should contact *conference@usenix.org* regarding this offer.

STANDARDS ACTIVITIES

The Board voted to allocate $2,500 to renew USENIX's membership in The Open Group, at a reduced rate that allows us to retain a vote on standards issues.

GRANTS

It was agreed to grant $3,000 to the Middleware 2003 conference to support student travel.

It was agreed to fund $10,100 to support the request from the Computing Research Association's Committee on the Status of Women in Computing to support one student in the Distributed Mentoring Project ($7,100) and one student in the Collaborative Research Experiences for Women and Minorities program ($3,000).

CONFERENCES

Symposium on Networked Systems Design & Implementation (NSDI '04). It was agreed that USENIX would put together an agreement to co-sponsor NSDI with ACM Sigcomm and Sigops.

It was agreed to accept ACM Sigmobile's proposal that USENIX again co-sponsor **MobiSys** in 2004 with a different agreement. ACM Sigmobile will finance and organize MobisSys in 2004.

Internet Measurement Conference. USENIX will co-sponsor this conference once again in 2003 with ACM Sigcomm, and provide $5000 in student stipend funding, help with event marketing, and post the Proceedings on the USENIX Web site.

USENIX Security Symposium 2004. Matt Blaze will serve as program chair, with Vern Paxson and Avi Rubin coordinating the Invited Talks program.

Future USENIX Annual Technical Conferences. For 2004, it was decided to embrace a new format for a 6 day conference that would feature a 5 day single track of general and freenix sessions; a 5 day track of sessions designed for SIGs and focussed topics (i.e., Uselinux, UseBSD, Beowulf/clusters, client computing, security, sysadmin, etc.); tutorials on all 6 days; invited/plenary talks on 5 days; Gurus and BOFs as usual; and no vendor exhibition. A per-day registration fee will also be offered.

It was also agreed to invite Andrea and Remzi Apraci-Dusseau to serve as program co-chairs for the general sessions, and Keith Packard and Bart Massey will serve as co-chairs for the Freenix track. The committees for the special interest groups and topics will be pulled together in the near future.

Beginning 2005, it was decided to move the Annual Technical Conference into an early Spring timeframe, e.g., March/early April.

SAGE

The USENIX Board voted to implement a proposal from the SAGE Executive Committee to have a SAGE only membership category (which does not require SAGE members to become full members of USENIX with voting rights), the details to be worked out by the staff.

NOMINATING COMMITTEE

Dan Geer was appointed to seve as the Chairman of the Nominating Committee for the USENIX Board of Directors election in 2004.

BOARD MEETINGS

The next meeting of the USENIX Board of Directors is scheduled for Tuesday, October 28, 2003, in conjunction with LISA 2003 conference in San Diego, CA.

by Ellie Young

Executive Director

ellie@usenix.org

USENIX Association Financial Report for 2002

The following information is provided as an annual report of the USENIX Association's finances and represents the Association's statement of revenue and expenses for the year. Accompanying the statements are several charts on specific aspects of the Association's financial position.

FINANCIAL STATEMENTS SUMMARY. The downturn in the economy continues to be a big factor in decreased revenues in 2001 and 2002. Conference attendance at the USENIX Annual Technical Conference was down 32%, but revenue from other conferences and dues was stable. In 2002, conferences contributed a net $282K, in large part due to cost control. The Association suspended most of its Good Works programs. SAGE, and SAGE Certification incurred expenses but did not generate sufficient revenue from dues or tests to cover them. The net operating deficit for all USENIX programs was $831K. The Reserve Fund declined by $734K.

USENIX Membership Dues and Expenses USENIX averaged 7,500 members in 2002, which is a 10% drop from 2001. Of these, 56% opted for SAGE membership as well. Chart I shows the total USENIX dues income ($742K) for 2002, divided into membership types. Chart 2 shows how those dues were spent. Note that all costs for producing conferences, including staff, marketing, and exhibitions are covered by revenue generated by the conferences.

SAGE

Chart 3 shows SAGE revenue sources for 2002 (Dues accounts for $138K, USENIX subsidy to cover losses is $232K, and the share of LISA conference revenue allocated to SAGE is $131K). Chart 4 provides a breakout of SAGE expenses which total $501K.

Student Programs, International Conferences, and Good Works. Chart 5 describes how the money allocated ($300K) was spent in 2002.

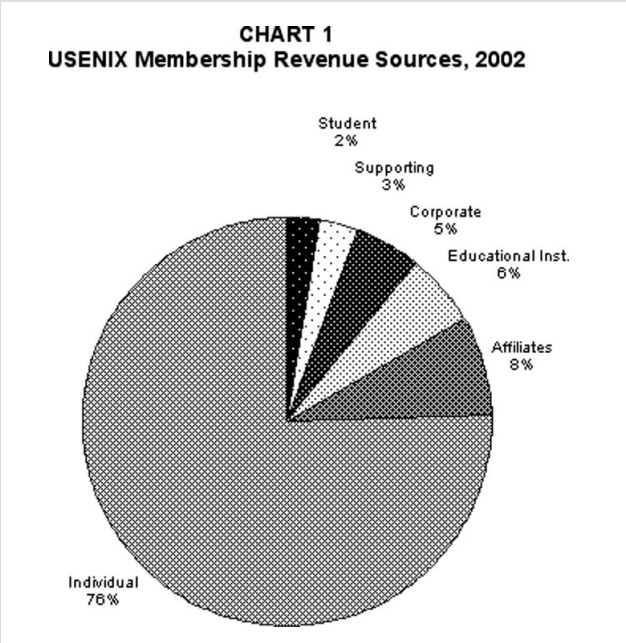


CHART 2
Where Your 2002 Membership Dues Went

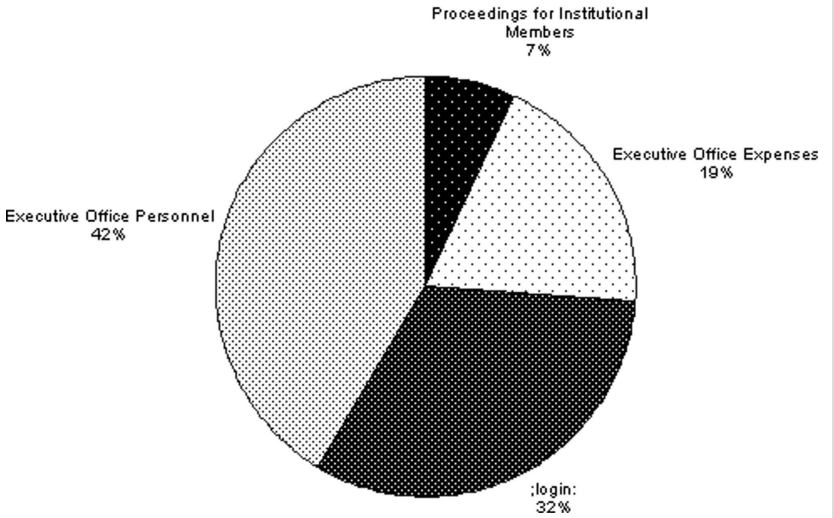


CHART 3 SAGE Revenue Sources, 2002

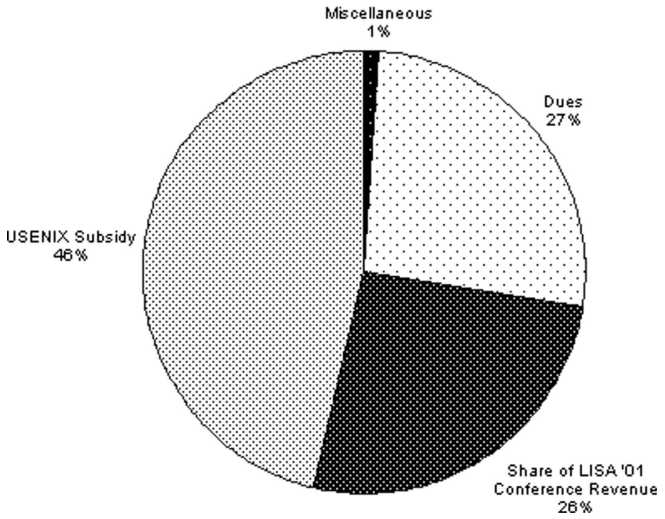


Chart 4
SAGE Expenses, 2002

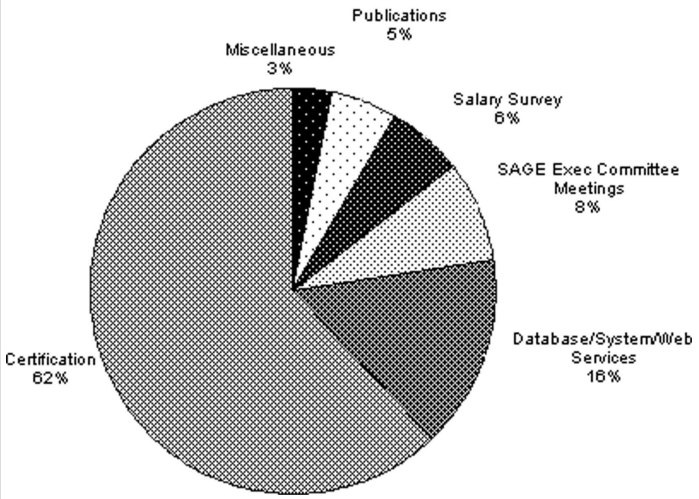
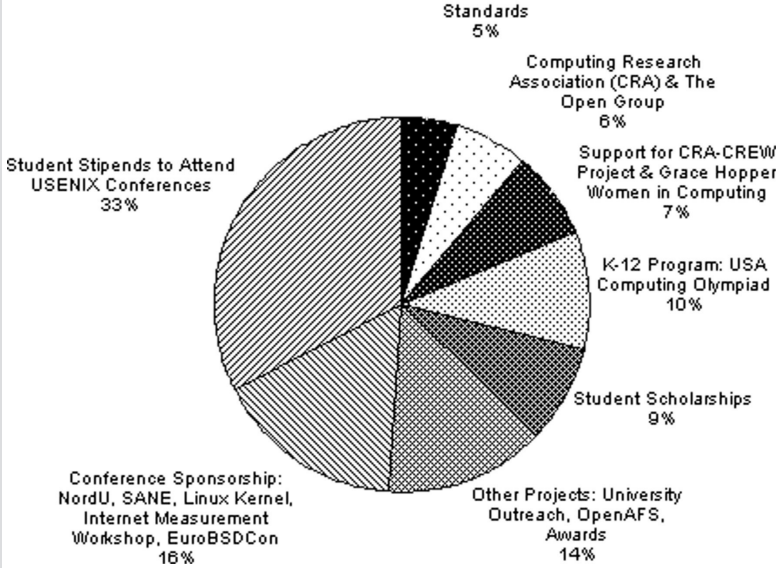


Chart 5
Programs & Good Works Projects, 2002



USENIX ASSOCIATION
STATEMENT OF FUNCTIONAL EXPENSES
For the Years Ended December 31, 2002 and 2001

	Conferences and Workshops	Programs and Membership	Student Programs, Good Works and Projects	SAGE	Sage Certification	Total Program	Management and general	Fund Raising	Total Support	2002 Total	2001 Total
Operating Expenses											
Conference & workshop-direct	$ 1,840,022	$	$	$	$	$ **1,840,022**	$ 10,408	$ 15,316	$ **25,724**	$ **1,865,746**	$ 2,666,136
Personnel and related benefits:											
Salaries	778,823	107,384	8,356	153,668		**1,048,231**	118,507		**118,507**	**1,166,738**	1,110,550
Payroll taxes	57,518	7,855	617	11,349		**77,339**	8,752		**8,752**	**86,091**	82,498
Employee benefits	140,794	19,229	1,511	27,780		**189,314**	21,424		**21,424**	**210,738**	203,789
Membership/proceedings		70,683				**70,683**			**0**	**70,683**	40,102
Membership/login:		216,016				**216,016**			**0**	**216,016**	343,088
Membership/e-learning:		53,015				**53,015**				**53,015**	
SAGE expenses				190,885		**190,885**			**0**	**190,885**	184,797
SAGE Certification expenses					309,885	**309,885**			**0**	**309,885**	287,793
Student programs, Good Works, and projects			285,583			**285,583**			**0**	**285,583**	967,193
General and administrative	266,204	79,046	4,214	58,745	6,351	**414,560**	208,587	23,691	**232,278**	**646,838**	803,935
	$ 3,083,361	$ 553,228	$ 300,281	$ 442,427	$ 316,236	$ **4,695,533**	$ 367,678	$ 39,007	$ **406,685**	$ **5,102,218**	$ **6,689,882**

USENIX ASSOCIATION
STATEMENT OF FINANCIAL POSITION
December 31, 2002 and 2001

ASSETS	2002	2001
Current assets:		
Cash & cash equivalents	$ 1,049,294	$ 476,185
Accounts receivable	44,919	66,936
Prepaid expenses	27,824	108,977
Inventory	22,045	31,225
Total current assets	1,144,082	683,323
Investments at fair market value -reserve fund	4,346,762	6,638,588
Property and equipment:		
Office furniture and equipment	433,538	422,576
Less: accumulated depreciation	(256,939)	(183,204)
Net property and equipment	176,599	239,372
Total assets	$ 5,667,443	$ 7,561,283
LIABILITIES AND NET ASSETS		
Current liabilities:		
Accounts payable and accrued expenses	$ 355,568	$ 633,503
Deferred revenue	12,390	63,045
Total liabilities	367,958	696,548
Net assets:		
Unrestricted net assets:		
Board designated	4,346,762	6,638,588
Undesignated	952,723	226,147
Total unrestricted net assets		
Temporarily restricted net assets		
Total net assets	5,299,485	6,864,735
Total liabilitites and net assets	$ 5,667,443	$ 7,561,283

USENIX ASSOCIATION
STATEMENTS OF ACTIVITIES
For the Years Ended December 31, 2002 and 2001

	2002	2001
Operating revenues:		
Conference and workshop revenue	$ 3,371,062	$ 3,506,275
Membership dues	741,587	739,856
SAGE dues & other revenue	137,182	151,820
Product sales	20,129	20,676
SAGE Certification	847	10,750
Total operating revenues	4,270,807	4,429,377
Operating expenses:		
Program services:		
Conference and workshop revenue	3,083,362	4,063,800
Programs and membership	553,228	629,833
Student programs, Good Works, and projects	300,281	981,806
SAGE	442,427	349,713
SAGE Certification	316,236	287,793
Total program services	4,695,534	6,312,945
Suppport services:		
Management and general	367,678	349,870
Fund raising	39,006	27,067
Total support services	406,684	376,937
Total operating expenses	5,102,218	6,689,882
Net operating (deficit) surplus	(831,411)	(2,260,505)
Net investment income and nonoperating activities		
Donations	45,560	532
Interest and dividend income	156,327	240,445
Net realized and unrealized losses on investments	(865,941)	(1,185,139)
Investment fees and costs	(69,785)	(94,171)
Net investment income and nonoperating activities	(733,839)	(1,038,333)
Change in net assets	(1,565,250)	(3,298,838)
Net assets, beginning of year	6,864,735	10,163,573
Net assets, end of year	$ 5,299,485	$ 6,864,735

USENIX ASSOCIATION
STATEMENTS OF CASH FLOWS
For the Years Ended December 31, 2002 and 2001

	2002	2001
Cash flows from operating activities:		
Change in net assets	$ (1,565,250)	$ (3,298,838)
Adjustments to reconcile change in net assets to net cash (used in)/provided by operating activities:		
Depreciation	73,735	77,455
Net investment income designated for long-term purposes	(73,987)	(94,289)
Realized and unrealized losses on investments	865,941	1,185,139
(Increase) decrease in assets:		
Accounts receivable	22,017	298,046
Prepaid expenses	81,153	(14,854)
Inventory	9,180	(11,076)
Increase (decrease) in liabilities:		
Accounts payable and accrued expenses	(277,935)	(226,723)
Deferred revenue	(50,655)	23,695
Net cash (used in) provided by operating activities	(915,801)	(2,061,445)
Cash flows from investing activities:		
Purchases of investments	(3,765,885)	(5,646,360)
Proceeds from sale of investments	3,765,885	5,646,360
Withdrawals from reserve fund	1,499,872	355,000
Additions to reserve fund		
Purchases of property and equipment	(10,962)	(29,433)
Net cash provided by (used in) investing activities	1,488,910	325,567
Net (decrease) increase in cash and cash equivalents	573,109	(1,735,878)
Cash and cash equivalents, beginning of year	476,185	2,212,063
Cash and cash equivalents, end of year	$ 1,049,294	$ 476,185

conference reports

This issue's reports focus on MobiSys 2003, HoTOS-IX, and the 2003 European Tcl/Tk User Meeting

Our thanks to the summarizers:

for MobiSys 2003
James Mickens

for HoTOS
Ranjita Bhagwan
David Oppenheimer
Amit Purohit
Matt Welsh

for Tcl/Tk
Clif Flynt

Mobisys 2003

The First International Conference on Mobile Systems, Applications, and Services

San Francisco, California
May 5–8, 2003

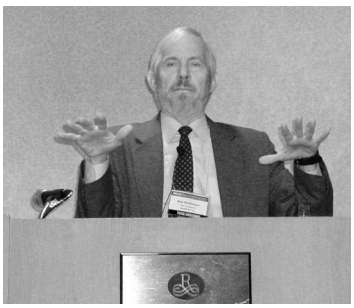
Summarized by James Mickens

KEYNOTE ADDRESS

Design of Wireless Systems-on-a-Chip

Bob Brodersen, Berkeley Wireless Research Center, University of California, Berkeley

A "system-on-a-chip" (SoC) provides integrated components for computation and network communication in a single piece of hardware. SoCs are particularly well suited for small mobile devices which must execute user tasks and transfer data over the Internet. Bob Brodersen argued that the key question for any SoC design is, "What is the cost of flexibility?" There is a fundamental tension between simple designs that do one thing well and larger, more complex designs that provide many features. More flexible designs are often attractive for business reasons (e.g., backward compatibility or the ability to sell one core design for multiple operating environments). However, Brodersen showed that flexible chips are much less efficient than specialized chips, often by several orders of magnitude.



Bob Brodersen

Brodersen examined three types of chips: microprocessors, general-purpose DSPs, and dedicated chips for specialized environments. He noted that supply voltage is relatively constant across all three chip types. However, microprocessors have much higher switched capacitances – in their quest to support high clock rates, they introduce a lot of logic to drive and multiplex the clock signal.

Brodersen then discussed the key relationship between chip area and operations per unit time. Microprocessors have the highest ratio of chip area per operation, whereas dedicated processors have the lowest. Dedicated processors are more energy-efficient because they can extract application-specific parallelism. At any given moment, the vast majority of a dedicated chip's area is actively utilized. In contrast, flexible designs must support many general-purpose operations whose corresponding chip areas are often irregularly utilized. Brodersen drew several conclusions from these results. First, he observed that from an energy perspective, it is better to increase parallelism than to increase the clock rate. He then forcefully noted that, for any given problem, a dedicated hardware solution always has better performance and energy efficiency than a software solution running on a flexible chip. In fact, the popular notion of a "hardware/software" tradeoff is imaginary, because the dedicated hardware solution is always the best! Once again, he stressed that flexible-chip designs are often motivated by business concerns as opposed to efficiency concerns. In general, dedicated parallel processors are the best solution.

Brodersen then described his methodology for rapid prototyping of parallel SoCs. Instead of using a low-level design language like VHDL, Brodersen uses the Simulink and Stateflow programs provided by Mathworks. A chip design is decomposed into timing dataflows and finite-state machines. The design can then be instantiated in an FPGA or ASIC

chip. The FPGA solution is particularly attractive, for several reasons. First, it is very easy to connect multiple FPGAs in parallel. Second, even though the clock rates of FPGAs are not exceptionally fast, FPGAs take full advantage of technological advances in hardware density. Thus, as time progresses, an individual FPGA can contain more and more parallel units.

An audience member asked Brodersen, "How much parallelism can we expect from the real applications that users want to run?" This was an insightful question – if users typically run serial applications, then explicitly parallel chips would not be commonly optimized. Brodersen quickly replied that the current computational paradigm is wrong: Instead of writing serial applications and then trying to run them on parallel chips, we should write explicitly parallel programs for explicitly parallel processors. Brodersen said that the computer community was committing a "great injustice" upon the current generation of programmers by forcing them to learn C. Instead, the community should focus on generating natural methods for describing concurrent phenomenon. Brodersen admitted that his current Mathworks prototyping system is not optimal. However, it is much better adapted to the creation of explicitly parallel applications than many other prototyping systems.

PANEL

How Should We Evaluate Systems Contributions to Pervasive/Ubiquitous Computing?

Keith Edwards, Palo Alto Research Center; Armando Fox, Stanford University (moderator); Anthony LaMarca, Intel Research; Brian Noble, University of Michigan; Yi-Min Wang, Microsoft Research

Armando Fox opened the panel discussion by observing that ubiquitous-computing (ubicomp) research is a combination of two seemingly disparate fields: HCI work and systems research. Some ubicomp research focuses too heavily on the HCI aspect, leading to interesting proof-of-concept projects that cannot handle real-life workloads. Other ubicomp research emphasizes the lower-level systems aspects; unfortunately, this often results in "solutions" that do not truly address the needs of typical users. How can the ubicomp community find a satisfactory methodology for designing and evaluating ubicomp systems?

Brian Noble argued that ubicomp researchers must create metrics that quantitatively describe users' subjective experiences. Noble said that current metrics are often chosen for their mathematical tractability and are only indirectly related to actual user preferences. Keith Edwards echoed these sentiments. He reminded the audience that ubiquitous computing is inherently user-centric, so the ultimate evaluation metric for any such system must be end-user utility.

Yi-Min Wang and Anthony LaMarca agreed that the subjective end-user experience is important. LaMarca's three evaluation criteria for ubicomp systems were robustness, programmability, and manageability. Yi-Min argued that improved fault modeling is critical to the acceptance of ubiquitous computing. For example, people will not buy a house filled with pervasive computers unless they believe that they can fix most errors without professional help, and that a total system failure would not result in personal danger.

The audience members asked many thought-provoking questions. One attendee wondered why system designers and HCI experts should communicate at all – shouldn't they both stick to their respective areas of expertise? Noble responded that each discipline must have an understanding of the problems in the other. Without synergistic interaction between the systems and HCI communities, it is impossible to create ubicomp systems that excel in both aspects.

Another audience member proposed that ubiquitous computing is not about "killer demos." Instead, it is about a "killer lifestyle." The ubicomp community has many vignettes that illustrate simple ways in which pervasive computing is useful, but it has not effectively demonstrated the full power of the ubicomp model. The panel found no easy solutions to this challenge. LaMarca noted that pervasive computing is a new computational paradigm; it forces researchers to revisit traditional notions of what a computer can and cannot do. As the field matures, researchers will discover new ubicomp applications that can change popular perceptions of the technology.

The audience loved an attendee suggestion that conferences like MobiSys offer tutorials on conducting user studies. Many felt that a gentle introduction to the user evaluation process would help systems researchers better understand the role of the end-user in a ubicomp environment.

DEMO/POSTER SESSION

E-textiles

Traditional sensor-node systems use wireless communication protocols and provide each node with an individual power supply. An e-textile is a piece of fabric that contains embedded processors, sensors, and actuators. Unlike a traditional sensor node, an e-textile node communicates via wires that are threaded along the surrounding fabric. E-textile nodes also draw their energy from embedded power lines. Thus, unlike more common sensor networks, an e-textile grid allows nodes to share energy resources as easily as they share information. An obvious application of e-textiles is in the domain of wearable computers. For example, one could create an e-textile glove that sensed the movements of the user's fingers, providing a virtual keyboard or musical instrument. One could also take advantage of the ease with which e-textile sensors can

be deployed and later recovered. Building inspectors could test for asbestos by rolling an e-textile carpet onto a floor or crawlspace; once the sensors had finished their sampling, the inspectors could simply roll up the carpet and move to the next room.

SmartView: Enhanced Document Viewer for Mobile Devices

Small mobile devices like PDAs have difficulty displaying Web pages that are large and have complex layouts. The PDA is often forced to break these pages into screen-sized chunks. These chunks do not have semantically meaningful boundaries and are usually poorly formatted; the user must engage in extensive scrolling to fully understand the document. SmartView analyzes the source of a Web page and breaks it into properly formatted segments. It then creates thumbnails for each segment, allowing the user to preview a segment before expanding it to its full size. SmartView also provides annotations for pages returned by Google searches. Each segment of a returned page will contain a certain number of the total keyword hits. SmartView graphically depicts the number of hits that each segment has in its thumbnail. Users can then directly jump to segments with the most hits.

MagnetOS: An Operating System for Mobile Ad Hoc Networks

MagnetOS is an operating system for ad hoc networks. It provides a collection of nodes with the abstraction of a single Java Virtual Machine. MagnetOS also automatically splits programs into mobile partitions, and it dynamically migrates partitions in an effort to minimize energy consumption. MagnetOS uses two object-placement strategies: netCenter and netPull. In the netCenter approach, a mobile code object moves directly to the node that generates the largest percentage of the packets that it receives. In the netPull strategy, mobile code is moved in the general direction where most packets are generated. In other words, given an object receiving packets from multiple sources, the object will move to the "center of gravity" of the aggregate packet flow. Experiments show that both techniques result in large energy savings when compared to static or random migration models.

Mobile Web Services

IBM Research Technologies had several interesting demos. One dealt with providing Web services via mobile devices. Suppose that you own an IBM Linux watch with embedded Bluetooth networking. Your watch can export multiple services. For example, IBM demonstrated a watch that exported a payment protocol; if you go to a store that is Bluetooth-enabled, you can purchase your items via your watch. Your watch could export a time service that allows it to synchronize its clock with other network devices. Your watch could also exchange electronic business cards with other watches.

Constraints: An Abstraction to Express Semantics for Reconciliation

Mobile devices often have intermittent network connectivity. This means that if several devices want to modify a shared database, there will be periods when a node's updates cannot be immediately propagated to its peers or the central repository. When devices regain connectivity and submit their updates, the database must reconcile all of the updates to ensure that the final database state is "sensible." Individual applications often have specific reconciliation semantics, but the database would prefer to support multiple applications in a generic fashion. When mobile devices in the IceCube system emerge from disconnected operation, they do not transmit their disconnected updates directly to the database. Instead, they send these transactions to an automatic inference module. This module outputs generic constraints for a device's application-specific updates. For example, one set of operations may need to commit in sequential order; in another update group, perhaps all of the actions must commit or none of them must commit. The automatic inference module delivers a log of actions and a constraint list to the IceCube Generic Reconciler. If the reconciler can generate a feasible reconciliation schedule, it delivers the associated transactions to the central database, which is oblivious to application-specific reconciliation semantics.

LOCATION MANAGEMENT

Single Reflection Spatial Voting: A Novel Method for Discovering Reflective Surfaces Using Indoor Positioning Systems

Robert Harle and Andy Hopper, University of Cambridge; Andy Ward, Ubiquitous Systems Limited

A key goal of many pervasive computing systems is to generate a map of their surroundings. In the Single Reflection Spatial Voting system, people wear tags that emit ultrasonic pulses. As users walk through a room, the reflections from their tags' pulses are detected by sensors in the ceiling. By observing the intersections of these reflections, the sensors can determine the locations of walls and furniture in the room.

The Lighthouse Location System for Smart Dust

Kay Römer, ETH Zurich

"Smart dust" networks consist of millimeter-scale autonomous devices with integrated computing, sensing, and wireless communication capabilities. A base station acts as a data sink for information collected by the nodes. To impose a geographic ordering over this data, the nodes must have a sense of their relative spatial orientations. How can we provide this topological information without consuming an excessive amount of power? Nodes cannot use active radio communication, because the required antennas are too large and require too much power. In the Lighthouse approach, the nodes detect their location in a passive fashion. The base station emits a continually rotating light

stream. During each rotation, a node can measure the amount of time that it is illuminated by the beam. If the node knows the rotation rate of the lighthouse, it can use simple trigonometric formulas to determine its distance from the base station. If we introduce a second and third lighthouse, all with perpendicular beam sweeps, then a node can determine its location in two and three dimensions, respectively.

The primary advantage of the lighthouse protocol is that a node does not expend energy talking to other nodes or to the base station. Furthermore, the code that performs the trigonometric calculations has small CPU and memory requirements.

A member of the audience observed that the sensor nodes are very small and thus can be jostled by the wind or other vibrations. These movements could disturb a node's observation of the lighthouse beam and thus upset its location calculations. The speaker said that this problem can be solved by equipping nodes with accelerometers. The accelerometers would measure any unexpected movement, and the trigonometric calculations could be adjusted by the necessary amount.

Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking

Marco Gruteser and Dirk Grunwald, University of Colorado, Boulder

Fifteen years ago, Tim McCarthy of Motorola's GPS business noticed that many devices suddenly had embedded clocks. McCarthy now predicts that every device will soon have an embedded location sensor. As these sensors become ubiquitous, they will introduce new threats to location privacy. For example, if a malicious party has access to accurate location information about you, he can infer whether you have recently visited a hospital or a political organization. The key idea underlying cloaking is k-anonymity. A subject is k-anonymous if its associated location data is indistinguishable from that of k-1 other subjects. In other words, given a rectangular bounding area and a time interval, this data must describe at least k unique subjects. To achieve this anonymity, mobile nodes indirectly communicate with location services via a trusted anonymity proxy. To provide k-anonymity, this proxy alters the position data in a location service request before forwarding it to the actual service. Communication between a node and its proxy is authenticated and encrypted to prevent eavesdropping. There are two primary areas of future work. Even though the proxy is trusted, it introduces another principal that can be subverted; a better system would eliminate the need for a proxy that is separate from the node itself. More research is also needed to discover appropriate values for k in different application environments.

SUPPORTING APPLICATIONS OVER MOBILE NETWORKS

Reservations for Conflict Avoidance in a Mobile Database System

Nuno Preguiça, J. Legatheaux Martins, Miguel Cunha, Henrique Domingos, Universidade Nova de Lisboa

Mobile devices often have intermittent network connectivity. If multiple clients can autonomously manipulate a shared central database, there must be a method for reconciling updates that occur when the devices are disconnected. To guarantee that client updates can always be successfully reconciled, the Mobisnap database gives clients reservations before they disconnect. For example, a reservation might allow a client to use a record value for a given amount of time, even though that value may be outdated when the client reconnects. A client can determine whether its transactions will commit on the central server by examining its personal reservation set. This introspection only requires local state. Thus, even disconnected clients can be confident that their transactions will commit if they have the appropriate reservations.

Protecting Applications with Transient Authentication

Mark D. Corner and Brian D. Noble, University of Michigan

Current authentication systems typically retain long-term authority to act on their users' behalf after login. Unfortunately, if your laptop is stolen after you login, the authentication system will not prevent a thief from rummaging through your private information. There is a fundamental tension between requiring frequent authentication, which is secure but irritating, and permitting infrequent authentication, which is more usable but less safe. In the transient authentication system, a laptop's hard disk data is always encrypted. The user wears a token that has wireless networking capabilities. The token automatically and securely releases the keys that enable the laptop to decrypt its disk data. When the user leaves (as indicated by the token moving out of communication range), the laptop encrypts its memory. When the user returns, the token transparently authenticates the user and provides the necessary keys for the laptop to decrypt its memory and resume execution. Corner and Noble also provide an API that allows applications to selectively protect sensitive in-memory information.

iFlow: Middleware-assisted Rendezvous-based Information Access for Mobile Ad Hoc Applications

Zongpeng Li, Baochun Li, and Xin Zhou, University of Toronto; Dongyan Xu, Purdue University

iFlow is a middleware framework for disseminating information in mobile ad hoc applications. iFlow leverages node mobility to support "information rendezvous": data suppliers spread popular content on third-party nodes as they travel, and data consumers collect these information deposits as they move around the network. Suppliers use Tor-

nado codes and network coding to efficiently break content into smaller units that are easily distributed and reconstructed. Using these techniques, iFlow uses less communication bandwidth than systems that deliver data directly from supplier to consumer.

SYSTEMS SUPPORT FOR MOBILITY

Full TCP/IP for 8-bit Architectures

Adam Dunkels, Swedish Institute of Computer Science

Conventional wisdom states that the TCP/IP protocol suite is too complex to fully implement in a constrained resource environment. Dunkels presented compact implementations of the TCP/IP stack for 8-bit architectures. These stacks satisfy the necessary properties from RFC 1122 that enable a host to act as an endpoint for generic TCP traffic. 8-bit applications interact with the stacks through an event-driven API.

Several audience members challenged the need for an 8-bit TCP/IP stack. One person suggested that there are no 8-bit applications that need complete TCP/IP support and that more complex chips have enough resources to support the unmodified stack. Another person proposed that the full protocol stack should be run on a proxy machine. This solution would provide embedded devices with the reliability of TCP while keeping embedded code size small.

System Services for Ad Hoc Routing: Architecture, Implementation, and Experiences

Vikas Kawadia, University of Illinois, Urbana-Champaign; Yongguang Zhang, HRL Laboratories, LLC; Binita Gupta, Qualcomm Inc.

The authors argued that current operating system architectures are inappropriate for supporting ad hoc routing. Most OSes separate the notions of packet forwarding and packet routing. However, in ad hoc protocols, a data packet can also have a routing function (e.g., route discovery). The authors defined a generic API to add ad hoc routing support to an OS's existing networking framework. They implemented this API for the Linux 2.4 kernel using a user-level library and a small loadable kernel module. The authors used their new API to implement AODV and part of DSR. Referring to the difficulties they encountered in correctly implementing these protocols, the authors argued that separating routing and forwarding mechanisms in ad hoc protocols is "profoundly important" for ensuring protocol efficiency, extensibility, and ease of implementation. This proposition was vigorously criticized by Dave Johnson, who said that ad hoc protocols combine forwarding and routing for valid reasons. Johnson dismissed the notion that implementation difficulty always justifies changes to valid design decisions; he cited the example of TCP, which is complex but effective.

Predictive Resource Management for Wearable Computing

Dushyanth Narayanan, Carnegie Mellon University; Mahadev Satyanarayanan, Carnegie Mellon University and Intel Research Pittsburgh

Applications for wearable computers (e.g., speech recognition or translation software) always desire more resources than are available in such a constrained environment. To ensure low response times for these resource-intensive applications, a system can support multi-fidelity computations. A multi-fidelity computation is one which accepts a computation request and a description of available resources, and generates the highest fidelity result that it can achieve with those resources. The authors describe a concrete system which, given the current resource supplies, can automatically predict the latencies associated with generating outputs of varying fidelity. The system executes the highest quality operations that still have tolerable latencies; it observes the actual correlations between output fidelity and resource usage to dynamically calibrate its predictions. Experiments show that this approach can reduce mean latency by 60% and latency variability by 30%.

SENSOR NETWORKS

Design and Implementation of a Framework for Efficient and Programmable Sensor Networks

Athanassios Boulis, Chih-Chieh Han, and Mani B. Srivastava, University of California, Los Angeles

SensorWare is a framework for creating distributed applications in wireless ad hoc sensor networks. All programs are represented as event-driven state machines. After a user injects a program into the network, the program autonomously migrates and/or produces multiple copies of itself in response to changing environmental conditions. Users are not burdened with the chore of assigning tasks to nodes. The SensorWare runtime environment provides abstractions for radios, sensors, batteries, etc., so writing portable code is easy. SensorWare also provides support for threading and message queues.

An Entity Maintenance and Connection Service for Sensor Networks

Brian Blum, Prashant Nagaraddi, Anthony Wood, Tarek Abdelzaher, Sang Son, and Jack Stankovic, University of Virginia

The primary goal of sensor networks is to monitor environmental events. Blum et al. described an API for associating addresses with these events, making it easy for applications to communicate with nodes in the vicinity of the event. The API also allows the network to associate state with each event; this state migrates with the event as it moves through the network. One audience member questioned whether this model pushed too much work onto the lightweight sensor nodes. Blum noted that the amount of data sent to the base station is reduced, since information about each event is only conveyed to the base station by a single "leader node." However, he admitted that the complexity of

pattern recognition for different event types is not explicitly addressed by his system.

ENERGY MANAGEMENT

Operating System Modifications for Task-Based Speed and Voltage Scheduling

Jacob R. Lorch, Microsoft Research; Alan Jay Smith, University of California, Berkeley

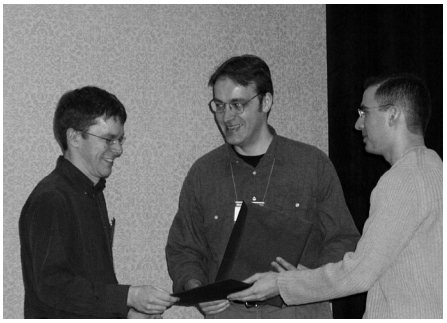
RightSpeed is a dynamic voltage scheduler for Windows 2000. Given a set of task deadlines, RightSpeed minimizes the voltage (and thus the processor speed) needed to meet these deadlines. Applications can explicitly provide deadlines, or RightSpeed can infer them by observing user-interface events and thread activity. A voltage/speed setting is "worthwhile" if using it saves more power than an emulated version that uses a combination of faster and slower settings. RightSpeed needs at least three worthwhile settings to improve performance. Interestingly, the authors discovered that several popular voltage-scaling processors do not have enough worthwhile settings for RightSpeed to save energy! However, simulations show that future processors with more worthwhile settings will reap large energy savings.

Energy-Aware Lossless Data Compression

Awarded Best Paper

Kenneth Barr and Krste Asanović, MIT

An add instruction consumes less than a nanojoule of energy, but sending a single bit over a wireless network can consume 1000 nanojoules. Barr and Asanović explored the energy savings that can be achieved via data compression and decompression over wireless links. They found that receiving and decoding compressed data is usually more efficient than receiving uncompressed data. The authors also observed that compression programs have poor cache behavior. Handling a cache miss is very expensive, so compressing data before transmission can lead to more energy consumption than regular uncompressed transmission! The authors demonstrated how careful selection of data structures can improve cache behavior and provide the desired energy savings. They described



Program Chair Robert T. Morris with Award winners Barr and Asanović

how idle power consumption affects the choice of compression algorithms. Finally, they showed how to optimize the energy savings for communication involving devices with different hardware profiles.

Energy-Adaptive Display System Designs for Future Mobile Environments

Subu Iyer, Robert Mayo, and Parthasarathy Ranganathan, Hewlett Packard Labs; Lu Luo, Carnegie Mellon University

Iyer et al. noted that displays consume over half of the power in mobile devices. They also presented a new user study showing that people using desktop and laptop computers typically focus on only 60% of the total display area. In their new "dark windows" system, the user's focused window retains its normal brightness and color, but the remaining screen areas are dimmed or displayed in a different energy-saving hue. By combining dark windows with new OLED screen technology, displays can consume 30% less power. An audience member questioned the authors' experimental methodology. He noted that he used 100% of his small laptop screen but a much lower percentage of his big desktop display. Therefore, it may be inappropriate to derive behavioral generalizations from a user set having heterogeneous display sizes, and thus potentially divergent usage patterns.

MOVING PARTS OF APPLICATIONS

Tactics-Based Remote Execution for Mobile Computing

Rajesh Krishna Balan, SoYoung Park, and Tadashi Okoshi, Carnegie Mellon University; Mahadev Satyanarayanan, Carnegie Mellon University and Intel Research Pittsburgh

How can users run resource-intensive applications such as speech recognition software on a resource-constrained mobile platform? In the basic remote execution approach, mobile clients offload work to nearby servers. These servers use their more powerful computational resources to calculate the required results, which are then shipped back to the mobile client. Balan et al. introduce the new idea of tactics. In their Chroma system, developers split programs into functional units called modules. Each application also exports a tactics list which enumerates the useful module-level partitions. These tactics constrain the search space for module distribution, reducing the time needed to determine the best allocation. Given current resource availability and expected resource demands, the best tactic is the one that provides the smallest latency and the best fidelity.

An audience member raised the insightful question of whether Chroma is applicable to applications people currently use. Chroma may work well for speech recognition and language translation, but would it improve the performance of popular email clients, Web browsers, or text editors?

Collaboration and Multimedia Authoring in Mobile Devices

Eyal De Lara, University of Toronto; Rajnish Kumar and Dan S. Wallach, Rice University; Willy Zwaenepoel, École Polytechnique Fédérale de Lausanne

To support multimedia collaboration between weakly connected mobile devices, the authors introduce two new concepts. Adaptation-aware editing distinguishes between user updates and fidelity modifications introduced by the adaptive system; users can edit low-fidelity data and later merge their changes with the shared high-fidelity version. Progressive update propagation reduces the upload time of updates by shipping partial or reduced-fidelity versions of these updates. The authors also decompose top-level documents into multiple component documents, e.g., sound, video, text. By reducing communication overhead and sharing granularity, mobile users can issue more frequent updates with fewer conflicts.

UNDERSTANDING AND BUILDING BETTER MOBILE NETWORKS

Characterizing Mobility and Network Usage in a Corporate Wireless Local-Area Network

Magdalena Balazinska, MIT; Paul Castro, IBM T.J. Watson Research Center

The authors provided detailed traces of a corporate WLAN environment spread over three buildings. They characterized user behavior along two primary dimensions: persistence (session duration) and prevalence (patterns of access point usage). The period of mobility for most users was typically more than one day, and 50%–80% of all users were occasionally or somewhat mobile. Most people spent a majority of their time at a single "home" access point. However, a user's average bandwidth usage was the same at arbitrary access points, and users had more short sessions than long sessions across all access points. The authors also observed that the aggregate bandwidth usage of an access point is somewhat correlated with its number of users, but it is strongly correlated with the identity of these users. An audience member commented that corporate infrastructures usually have fast-wired networks that people use for the majority of their work. He argued that the authors' traces do not truly represent "mobile" users. The session chair suggested that the study examine "portable" users as opposed to "mobile" users.

HotOS-IX

May 18–21, 2003
Lihue, Hawaii

[*This is a somewhat abbreviated set of summaries of the events at this conference. A complete set of summaries is available at* http://www.usenix.org/events/hotos03/. *Ed.*]

INVITED TALK

Summarized by David Oppenheimer

Operating Systems: Shouldn't They Be Better?

Andrew Hume, AT&T Labs–Research

Andrew Hume gave the HotOS keynote talk, explaining that his perspective comes from having designed, implemented, and delivered large-data applications for more than 10 years. The problems he discussed in the talk were that operating systems have gone "from a help to a hindrance," that even users' lowered expectations for operating systems have not been met, and that as a result, applications have to be designed around OS quirks. Hume pointed out that this situation hasn't always been the case, citing WORM-based backup systems in research versions of UNIX and a cluster-based billing system that AT&T built using Plan 9 as examples of systems that were highly reliable, even under load.

The first problematic system Hume described was Gecko, a large-scale (250GB/day) billing system implemented in 1996 on Solaris 2.6. AT&T required 1GB/sec. of file-system throughput and predictable use of memory. Among the problems encountered were: Solaris crashed every few days for the first six months that the system was in production; Solaris limited file throughput to about 600MB/sec.; reading large files sequentially crashed the VM; and a "VM roller coaster" developed when a large chunk of memory was allocated (causing a repetitive page-out, page-in cycle of all the system's physical memory, rather than just paging out the amount of new memory needed).

The second problematic system Hume described was a replacement for Gecko that required six times the capacity of the original Gecko. This system was implemented on a cluster running Linux. The architecture was a "Swiss canton" model of loosely affiliated independent nodes with a single locus of control, data replication among nodes, and a single error path so that software could only halt by crashing (there was no explicit shutdown operation). Hume described eight problems the Gecko implementers experienced with Linux (versions 4.18 through 4.20), including Linux's forcing all I/O through a file-system buffer cache with highly unpredictable performance scaling (30MB/sec. to write to one file system at a time, 2MB/sec. to write to two at a time), general I/O flakiness (1–5% of the time corrupting data read into gzip), TCP/IP networking that was slow and that behaved poorly under overload, lack of a good file system, nodes that didn't survive two reboots, and slow operation of some I/O utilities such as df. In general, Hume said he has concluded that "Linux is good if you want to run Apache or compile the kernel. Every other application is suspect."

Hume proposed the following definition of OS reliability: "[The OS] does what you ask, or it fails within a modest bounded time." He noted that FreeBSD has comparable functionality to Linux, better performance, and higher reliability, and he speculated that this might stem from BSD's (and other "clean, lean, effective systems") having been built using "a small set of principles extensively used, and a sense of taste of what is good practice, clearly articulated by a small team of mature, experienced people." Hume took Linux to task for not demonstrating these characteristics, in particular for being too bloated in terms of features, and for having been developed by too large a team. Further, he singled out the Carrier Grade Linux effort for special condemnation for "addressing zero of the [types of] problems" he has had.

SESSION: THE EMPEROR'S CLOTHES

Summarized by Matt Welsh

High Availability, Scalable Storage, Dynamic Peer Networks: Pick Two

Charles Blake and Rodrigo Rodrigues, MIT Laboratory for Computer Science

Charles Blake spoke on the overheads of "maintenance bandwidth" – network bandwidth consumed to maintain a given level of replication or redundancy – in a peer-to-peer storage system. The basic argument is that maintenance bandwidth across the WAN, not the aggregate local disk space, is the fundamental limit to scalability in these systems. Given the dynamics of nodes joining and leaving the system, Charles presented a conservative estimate of the maintenance bandwidth that scales with the WAN bandwidth and average lifetime of nodes in the system. Under a typical scenario (100 million cable modems with a certain bandwidth available for replication, one week average lifetime, and 100GB storage per node), only 500MB of space per node is usable, only 0.5% of the total.

To try to address these problems, Charles looked at alternatives such as admission control (only admitting "reliable" nodes) or incentivizing nodes to have long lifetimes. It turns out that a small core of reliable nodes (such as a few hundred universities with a single reliable machine dedicated to hosting data) yields as much maintenance bandwidth reduction as millions of home users with flaky connections. The talk concluded with a number of open issues in organizing WAN-based storage systems, such as whether it is appropriate to assume millions of flaky users and whether the requirement of aggregate data availability should be reconsidered.

ONE HOP LOOKUPS FOR PEER-TO-PEER OVERLAYS

Anjali Gupta, Barbara Liskov, Rodrigo Rodrigues, MIT Laboratory for Computer Science

Anjali Gupta presented a talk on the use of one-hop lookups in peer-to-peer systems, avoiding the high latency associated with the typical log (N) lookup paths required by most systems. The challenge is keeping up with membership change information on all hosts. For example, the UW Gnutella study in 2002 showed an average node session time of 2.9 hours, implying 20 membership changes per second in a system with 100,000 hosts. Anjali presented a hierarchical scheme, in which the address space (forming a ring) is subdivided into slices, each with a slice leader that is the successor to the midpoint in the slice. Slices are further subdivided into units.

The basic approach is for nodes to exchange frequent keep-alive messages with their predecessor and successor nodes. A change to a node's successor is an event that is propagated by piggybacking a recent event log onto keep-alive messages. A node change event is relayed to the slice leader, which periodically (every 30 seconds) notifies other slices of the updates. Internally to a slice, slice leaders periodically (every 5 seconds) notify unit leaders of node change information. Given some reasonable assumptions on the size of the system, all nodes can be updated within 45 seconds of a node leaving or joining the system, which permits a 99% "hit rate" for an address lookup. In this scheme, it is important to choose good slice leaders that are well-provisioned. Anjali concluded with a summary of ongoing implementation and experimentation work, noting that systems larger than a million nodes will require two-hop lookups.

AN ANALYSIS OF COMPARE-BY-HASH

Val Henson, Sun Microsystems

Val Henson presented one of the most controversial papers of the conference, admonishing those systems that rely upon comparison of data by comparing cryptographic hashes of the data. Many systems (such as rsync, Venti, Pastiche, LBFS, and OpenCM) use this technique, but it is not yet widely accepted by OS researchers, due to little characterization of the technique and many unanswered questions. The risk of collision using (say) a 160-bit SHA-1 hash is just 2^{-160}, which is far lower than a hardware failure or probability of an undetected TCP error. So why the controversy?

First, these techniques assume that data is random, but real data is not random and has a fair amount of commonalities (think about ELF headers and English text). Second, cryptographic hashes were designed for authentication and care about "meaningful" collisions, such as two contracts with the same text but different dollar amounts that happen to collide in the hash space. Third, hash algorithms are short-lived, and obsolescence is inevitable – systems need an upgrade strategy. Finally, collisions are deterministic – two blocks that collide always collide – rather than a transient error such as a hardware fault. Hash collision is therefore a silent error in those systems that rely on compare-by-hash techniques. Val claims that we should be striving for correctness in systems software, not introducing "known bugs." It is OK to rely on compare-by-hash when the address space is not shared by untrusted parties, and when the user knows and expects the possibility of incorrect behavior — citing rsync as an example. Note that "double hashing" is not an acceptable solution, as this results in just another hash function, albeit one with a lower collision probability.

Some alternatives to compare-by-hash were discussed, such as content-based addressing that checks for collisions, using compression, maintaining state to only send or store identical blocks once (as in LBFS), sending diffs instead of an entire block, or using universal IDs for common blocks.

WHY EVENTS ARE A BAD IDEA (FOR HIGH-CONCURRENCY SERVERS)

Rob von Behren, Jeremy Condit, Eric Brewer, University of California, Berkeley

Rob von Behren raised the argument of thread-based versus event-driven concurrency in high-concurrency servers, claiming that thread-based approaches are far better, due to their ease of programming. To counter the arguments that threaded systems have inherently higher overhead than events, Rob presented early results from a lightweight user-level thread system that performed as well as an event-driven system on a Web server benchmark. Furthermore, threads have better programming and debugging tools, leading to increased productivity. To address the problem of high overhead for per-thread stacks, Rob proposed the use of compiler support to automatically compress stacks, for example, by moving "dead" elements off the stack across a blocking operation. Using cooperative scheduling avoids the overhead of generic thread synchronization, but there are some issues to address here such as fairness, the use of multiprocessors, and how to handle spontaneous blocking events such as page faults.

Rob pointed out that events have the advantage of permitting very complex control flow structures, but very few programmers use these structures and threads can capture the vast majority of scenarios. Another problem with thread schedulers is that they are "generic" and have little knowledge of application structure. To permit greater cache locality, Rob proposed "2D" batch scheduling, in which the compiler annotates the application code to indicate to the scheduler system where the various stages of the thread's execution are located.

Rob presented some measurements of a simple Web server benchmark based on his user-level threads package, capable of supporting over 100,000 threads, implemented in about 5000 lines of C code. The server outperforms a Java-based event-driven Web server, probably due to the large number of context switches in the event-driven system. Rob concluded that it may be possible to achieve higher performance using threads than events, in part because events require dynamic dispatch through function pointers that makes it difficult to perform inlining and branch prediction.

Panel Discussion

Charles Blake kicked it off by asking why Val Henson's birthday paradox probability was so hard to compute. She responded that essentially it comes down to the infinitesimal numbers involved.

Eric Brewer pointed out that systems should use CRC, not MD5; since CRC is no good for preventing malicious collisions, there is no illusion that it is. One should also use a random salt with the checksum, which should help with the non-randomness of real data. Val responded that if you have to recompute the checksum across the actual data, then you are losing the benefits of this technique.

George Candea raised the point that although a P2P client that prevents a user from disconnecting appears less desirable at first, it would lead to higher availability for the service as a whole. This makes the service more valuable, and hence provides greater incentive to use it (i.e., download the client).

Ethan Miller asked whether people are really comfortable with the concept of probabilistic storage. Val agreed that the notion of dynamic, unreliable storage systems makes her uncomfortable.

Ranjita Bhagwan pointed out that Charles's calculations don't push P2P out of the picture, asking whether there may be a cost benefit to a peer-to-peer approach versus a centralized approach. Charles said that fundamentally his argument was economic, concerning the bandwidth versus storage requirements for these systems. Andrew Hume said that the best nodes are professionally managed and that high-bandwidth connections and support are expensive, so the economics of the two approaches are more similar than they are different.

Mohan Rajagopalan said that compiler optimizations actually perform very well for event-based systems, and that implementation is really what matters. Event-based systems permit a decoupling between caller and callee, so it is easier to write an event-based "adaptive" program than a threadbased one. Isn't this a fundamental benefit? Rob responded that events do make it easier to perform composition and interpositioning, but that this can also be done in the thread model. Eric mentioned that Click is very configurable and runs as a single large thread.

Peter Druschel was skeptical that we can do P2P storage based on home-connected desktops, but that the alternative is not centralized systems. For example, one can reap the benefits of unused desktop systems within a large organization. Charles did not disagree with that.

This was followed by an exchange between Peter and Rodrigo Rodrigues about using so-called "scalable lookup" vs. some other organization for P2P file storage. Basically, Rodrigo pointed out that in a scenario where the individual nodes are very available/reliable and the network isn't giant, there is no need for scalable lookup and other considerations should take priority. Peter responded that having a large number of nodes and security implied the need for small lookup-state optimizations.

SESSION: POPPING & PUSHING THE STACK

Summarized by Ranjita Bhagwan

TCP Offload Is a Dumb Idea Whose Time Has Come

Jeffrey C. Mogul, Hewlett Packard Laboratories

TCP offload in the traditional sense violates performance requirements, has practical deployment issues, and targets the wrong applications. TCP Offload Engines (TOEs) impose complex interfaces and cause suboptimal buffer management. Moreover, lots of small connections overwhelm savings because of connection management. Event management is a problem. Lots of virtual resources need to be managed. Also, one of the main motivations for TOE has been that TCP implementation in the OS is bad.

However, it is no longer a dumb idea, because now we are offloading higher-level protocols onto hardware. The justification for offloading TCP is simply that you can't offload the higher-level protocols without also offloading TCP. The sweet spot for TCP offload is when the application uses very high bandwidth and has relatively low end-to-end latency, long connection durations, and relatively few connections (e.g., storage server access and graphics). Also, several economic trends favor TCP offload. One would like to replace special-purpose hardware with cheap commodity parts, such as 1- or 10-gig Ethernet. This helps because with these in place, operators have only one kind of fabric to provision, connect, and manage. Still, many challenges remain. Data copy costs still dominate, and busses are too slow. Zero copy and single copy seem too hard to adopt in commercial OSes. However, with the advent of RDMA, vendors want to ship RNICs in volume, allowing one kind of chip for all applications. It would mean cheaper hardware. There are also several upper-level protocols available, such as NFSv4 and DAFS. Still, many problems of TCP offload remain:

There are security concerns, and so far the benefits have been elusive. The new networking model may require changes to traditional OS APIs. Systems people need to give this due consideration.

TCP Meets Mobile Code

Parveen Patel, Jay Lepreau, University of Utah; David Wetherall, Andrew Whitaker, University of Washington

The authors address the problem of deployment of transport protocols by proposing an extensible transport layer, called XTCP. The main argument is that transport protocols, such as TCP, need a self-upgrade mechanism, and untrusted mobile code can help build such a mechanism. Several modifications to TCP, as well as alternative transport protocols, have been proposed. However, as with any new protocol, deployment is an issue. Currently, it takes many years before a new protocol or an extension can be used by applications: A new protocol or extension has to be approved by standards committees, implemented by OS vendors, and finally enabled-by-default at both ends of communication.

In the proposed solution, untrusted peers can upgrade each other with new transport protocols using mobile code. A typical usage scenario is that of a Web server. A Web server can download a high-performance version of TCP, after which it tells every client to download the same version from it. Then the client and the server can speak the upgraded version of TCP. This solution avoids all the steps of the deployment process that need approval and support from third parties, such as standards committees and OS vendors.

There are several challenges to building such an extensible layer, notably host and network safety. The presenter contrasted XTCP with "active networking" and argued that the domain of transport protocols is restricted enough that host and network safety challenges can be met without degrading performance.

Host safety is assured by providing memory protection and resource control. Memory protection is achieved by using Cyclone, a typesafe C-like language. The stylized memory-usage pattern of TCP extensions – no shared state between extensions and predictable ownership of data buffers – makes resource control possible using traditional runtime methods. XTCP uses the well-understood notion of TCP-friendliness as a measure of network safety. All extensions written using the XTCP framework are forced to conform to TCP-friendliness using the ECN nonce mechanism. In contrast, active networking had no such well-defined notion of network safety, and host safety in the face of arbitrary code was costly.

XTCP has been implemented in FreeBSD 4.7. Support for user-level transports is being developed currently.

Exploiting the Synergy Between Peer-to-Peer and Mobile Ad Hoc Networks

Y. Charlie Hu, Saumitra M. Das, Himabindu Pucha, Purdue University

There appear to be a number of similarities in the problems addressed by research in peer-to-peer and mobile ad hoc networking. One such area is that of routing. The speaker showed the similarity between the problems solved by Pastry and how it can be used in ad hoc networking, too. He described a new protocol, DPSR, which stores routing state in a manner similar to Pastry. This reduces routing state per node from O(N) to O(log N). DPSR uses node ID assignment, node state, routing, node join procedures, and node failure or out of reach in much the same manner as Pastry; inherits all DSR optimizations on source routes; and contains a number of additional optimizations related to Pastry's routing structures and operations.

Simulations of DPSR for a 50-node system show that the routing overhead of DPSR scales better than that of DSR. In short, DPSR outperforms DSR when the number of connections per source is greater than 1; performance is otherwise equivalent.

Panel Discussion

Bogdan Popescu asked Parveen Patel if you could use a signing mechanism to detect unresponsive connections. Parveen said that the nice thing about XTCP is that it works well without it. Bogdan said that then you could have DoS attacks.

Rob von Behren said that it would be very easy to do DoS on XTCP, such as mallocing large amounts of memory, using a lot of CPU time, etc. Parveen said that each malloc call is accounted for. Rob responded that there is the problem of DDoS. With a considerable number of nodes using a little too much memory, one could perform a DDoS attack. Parveen said that this is possible and the only way to avoid it is strict admission control.

Jeff Mogul said that you have to make sure that XTCP itself is not subvertible. Because if it is, then it is a very rich environment for spreading worms.

Peter Steenkiste said that in the early '90s, after six months of effort, he had decided that TCP offloading is no good. In general, enthusiasm for TCP offload seems lukewarm. He asked Jeff if it would take off. Jeff responded that he does believe that it will take off, mainly for commercial reasons. Having only one fabric to manage for data centers seems good. Peter said that there appears to be a contradiction: Earlier on, we wanted to move things to the software level, and now attempts are being made to move them to the hardware. Jeff said that switches are clearly a larger investment than NICs. So commoditizing the NICs would be good.

Geoff Voelker asked Charlie Hu about how much the benefits of his approach depended on the amount of shared source routes. Did he have a sense of the minimum degree of shared source

routes needed for DPSR to work? Charlie answered that so far, the sharing was small, but even in this scenario, DPSR does no worse than DSR. So it seems like a total gain over DSR.

SESSION: DISTRIBUTED SYSTEMS

Summarized by Amit Purohit

Scheduling and Simulation: How to Upgrade Distributed Systems

Sameer Ajmani, Barbara Liskov, MIT Laboratory for Computer Science; Liuba Shrira, Brandeis University

Sameer Ajmani presented a solution to upgrade distributed software automatically with minimal service disruption. He described a technique that uses a combination of centralized and distributed components. The infrastructure consists of three main components: scheduling functions tell the node when to upgrade; simulation objects enable communication among nodes running different versions; and transform functions change a node's persistent state from one version to a higher one.

Development Tools for Distributed Applications

Mukesh Agrawal, Srinivasan Seshan, Carnegie Mellon University

Mukesh Agrawal explained the motivation for his current research. He claimed that the lack of distributed applications is because of implementation difficulties. He identified routing table upgrades for distributed applications as one of the harder problems. He mentioned the ns-2 simulator as a tool that helps to compare design choices. And DHT is developing building blocks to help implement distributed systems. Then he pointed out some inherent flaws in the current approaches. Research mainly concentrates on the initial stages of the life-cycle of the applications, while his work mainly addresses the issues with later life-cycle stages.

Virtual Appliances in the Collective: A Road to Hassle-Free Computing

Constantine Sapuntzakis and Monica S. Lam, Stanford University

Constantine Sapuntzakis envisioned a computing utility that runs not only Internet services but highly interactive applications commonly run on desktop computers. On desktops, patches arrive frequently and there is much multiple-application sharing of such things as OSes and libraries; hence, application upgrades can disrupt other applications. Constantine argued that it is possible to borrow an idea from "network connected computer appliances" to improve the manageability and usability of computers. In the architecture of their framework, groups of virtual appliances are maintained by makers without user involvement. Cheaper hardware made virtualization an attractive option.

POST: A Secure, Resilient, Cooperative Messaging System

Alan Mislove, Ansley Post, Charles Reis, Paul Willmann, Peter Druschel, and Dan S. Wallach, Rice University; Xavier Bonnaire, Pierre Sens, Jean-Michel Busca, and Luciana Arantes-Bezerra, Université Paris VI

A P2P solution was presented that interoperates seamlessly with a wide range of collaborative services by providing one serverless platform. It provides three basic services to applications: secure single-copy message storage; event notification; and single-writer logs that allow applications to maintain metadata. The claim was made that these features are sufficient to support a variety of collaborative applications.

Panel Discussion

Mike Swift asked Constantine Sapuntzakis about the cost of complex virtual appliances. He also noted that device drivers talk to hardware, hence couldn't be virtualized, and can crash if they are buggy. Constantine said future device drivers could be written in user-land and the problem could be solved. But if the application crashes, not much can be done.

Eric Brewer stated the view that the hard part is sharing information: Having separate virtual appliances for everything only works if they don't share any information, which means that the user must replicate all "shared" information by hand (as we do now with real appliances, e.g., setting the clock). The path of safe sharing leads you to shared segments as in Multics, including layers (rings) and call gates for protected calls into shared resources. The author replied that Multics has some problems and that they are planning to address them as well.

OUTRAGEOUS OPINIONS SESSION

Summarized by David Oppenheimer and Matt Welsh

In classic HotOS tradition, the Outrageous Opinions session consisted of a stream of short presentations, some serious, some mundane, some hilarious.

Val Henson argued against the use of checksums at all levels in a storage system versus end-to-end checksums at the application. Andrew Hume countered that it's good to have accountability at each level when something goes wrong in the system.

Matt Welsh presented "a brief history of computing systems research," in which he urged computer scientists to think about how their research can help to address social problems. He pointed out that computer scientists have always worked on improving life for computer scientists, focusing on improving their own day-to-day tasks. He suggested that computer scientists should think more about social problems rather than "How can I download porn and pirate music more efficiently?" In particular, he cited education and health care, particularly outside of the United States and Europe, as social problems that computer scientists could help tackle – for example, by empowering local government and

remote communities. Specific technologies he cited were peer-to-peer wireless networks for emergency and disaster response systems; censorship-proof electronic publishing of dissenting political opinions; sensor networks for environmental monitoring, precision agriculture, and inexpensive medical diagnostics; highly reliable power environments; and maintenance-free PCs.

Mendel Rosenblum talked about the similarities between micro-kernels and virtual machine monitors (VMMs) for running multiple untrusted environments on top of an operating system. Both provide isolation, resource management, and a small operating environment with simple interfaces, leading to high assurance. The key difference is what runs on top of each – for a VMM you don't need to port applications to run on it, whereas for a micro-kernel you do. He pointed out that despite this advantage for VMMs, academics and industry researchers are often interested in micro-kernels because, by not leveraging existing software, micro-kernels provide academics an opportunity to write lots of papers and industry an opportunity to define APIs, leading to an industry control point.

Mike Chen presented "two easy techniques to improve MTTR": redirect users' attention to what's still available when something becomes unavailable (e.g., "Please read this new privacy policy"), and blame it on the user (e.g., "Did you forget your password? Please try again."). He pointed out that by tricking the user, perceived availability can easily be increased.

Timothy Roscoe railed against the construction of scalable systems, claiming that systems should only scale as far as needed and no further. For example, email does not need to scale globally; who needs to send an email to everybody? After all, whitelisting your email to reduce spam doesn't scale, but it works. Timothy used Google as an example of a system where extra scalability only concentrates power. He cited libraries as a non-scalable alternative to Google.

Dan Wallach talked about all of the recent work on virtualizing resources and getting multiple virtual machines to share resources, such as shared libraries and the OS kernel. He proposed an alternative to these approaches, a radical concept called a "process."

Eric Brewer issued a call for IT research for developing regions. He made five claims: (1) there is a need for a direct attack by developing new devices rather than giving developing regions hand-me-down machines, which are a bad fit on cost, power, user knowledge, administration, and user literacy; (2) there is a need for infrastructure to support thousands of projects which are currently not sharing any infrastructure; (3) building IT for developing regions is economically viable, in that there is a market of 4 billion people, but IT must create income for its users (e.g., by offering a franchise model akin to that used to provide cell phone service to rural Bangladesh) because users do not have disposable income; (4) the time is right with the availability of five-dollar 802.11 chipsets, systems on a chip, low-power designs, and solar panels; and (5) this work can have a big impact by reducing poverty and disease and improving the environment, by providing developing regions with a source of income that they can in turn use to improve their standard of living, stability, and security. Lots of research directions here, including very low-power and low-cost wireless communications, new speech-based user interfaces, network proxies, and sensors.

George Candea discussed why he believes that wide-area decentralized systems such as peer-to-peer networks are "a good idea whose time has passed." He argued that such systems are hard to build, test, debug, deploy, and manage, and that they have little economic incentive beyond "lack-of-accountability" applications. He suggested that the principles learned from building wide-area distributed systems – strong compo nentization, using open protocols, loose coupling, reducing correlated faults through diversity, and writing components while keeping emergent behaviors in mind – should be used to build highly dependable "distributed" systems within the data center. He summarized by saying, "Don't distribute centralizable apps into the wide-area network, take the good ideas from distributed systems and apply them in the system-area network."

Geoff Voelker presented a novel idea based on the notion of value prediction from hardware architecture: "result prediction." Rather than running the program, we can simply guess the results, leading to excellent speedup potential!

Emmett Witchel asked whether there is a use for anti-optimization. One use he suggested was to allow users to specify in advance the amount of resources a computation takes, possibly eliminating covert channels. This would be accomplished by intentionally adding delay loops to code to use all available CPU and by spreading allocated memory all over the address space.

Ethan Miller proposed SCUBA: Scalable Computing for Underwater Biota Assessment, in which 802.11 networks would be deployed on coral reefs.

Sameer Ajmani suggested that systems researchers consider work in computational biology: "We help biologists, then they help thousands of people through pharmaceuticals, genetics, etc. – it's easier than sending computers to Africa." As specific examples of computational biology problems that can directly apply well-known computer science algorithms, he cited string alignment (dynamic programming) and database searches for genes (hashtable with 2-tuples and 3-tuples). Andrew Hume added that the National Institutes of

Health also has more money than the National Science Foundation.

Armando Fox called for a bet whether a peer-to-peer application would exist before the next HotOS, that would make more sense (economically and technically) to deploy as a peer-to-peer system than as a centralized service. Seven people in the audience said yes, 17 said no. Armando offered to bet someone (Armando taking the "no" side) for a case of alcohol valued less than the conference registration fee in 2005, but there were no takers.

SESSION: WHEN THINGS GO WRONG

Summarized by Amit Purohit

Crash-Only Software

George Candea and Armando Fox, Stanford University

George Candea explained how to build Internet services that can be safely and cheaply recovered by crash-rebooting minimal subsets of components. He stated that most downtime-causing bugs are transient and intermittent and that it is not feasible to guarantee that an application can never crash. For recovery-safe applications, recovery can be too long. Crash-only software achieves crash-safety and fast-recovery by putting all the important non-volatile state outside the application components into crash-only state stores. For systems of crash-only components to be crash-only, the components must be decoupled from each other, from the resources they use, and from the requests they process. He conceded that steady-state performance of crash-only systems may suffer, but argued that (1) the overall goal is to maximize the number of requests successfully served, not to serve them fast and then be unavailable for a long time, and (2) that techniques will evolve that will improve performance of crash-only systems, the way compilers improved the performance of programs written in high-level languages.

The Phoenix Recovery System: Rebuilding from the Ashes of an Internet Catastrophe

Flavio Junqueira, Ranjita Bhagwan, Keith Marzullo, Stefan Savage, and Geoffrey M. Voelker, University of California, San Diego

This presentation explained the design of an operative, distributed remote backup system called the Phoenix. Operating systems and user applications have vulnerabilities. A large number of hosts may share vulnerabilities and this can result in major outbreaks. Phoenix uses a strategy with attributes and cores. By replicating data on a set of hosts with different values for each attribute, it is possible to reduce the probability of error to near zero. In the Phoenix system there is no single point of failure; copying with a large number of requests is achieved by exponential backoff.

Using Runtime Paths for Macroanalysis

Mike Chen, Eric Brewer, University of California, Berkeley; Emre Kiciman, Armando Fox, Stanford University; Anthony Accardi, Tellme Networks

Mike Chen emphasized the benefits of microanalysis, namely latency profiling, failure handling, and detection diagnosis. He introduced the concept of "runtime path analysis, where paths are traced through software components and then aggregated to understand global system behavior via statistical inference." Runtime paths are also used for failure handling and to "diagnose problems all in an application-generic fashion." The group explained that their work could be extended to P2P message paths, event-driven systems, forks, and joins.

Magpie: Online Modeling and Performance-Aware Systems

Paul Barham, Rebecca Isaacs, Richard Mortier, and Dushyanth Narayanan, Microsoft Research Ltd, Cambridge, UK

Magpie is "a modelling service that collates traces from multiple machines . . ., extracts request-specific audit trails, and constructs probabilistic models of request behaviour." The presenter mentioned that workload description and hardware modeling could be used to predict performance. It is possible to augment the system by getting feedback from past models.

Using Computers to Diagnose Computer Problems

Joshua A. Redstone, Michael M. Swift, Brian N. Bershad, University of Washington

Redstone described "building a global scale automated problem diagnosis system that captures the . . . workflow of system diagnosis and repair." A computer generates search terms and locates problem reports. It stores problem reports in a canonical global database. When a problem occurs the computer detects symptoms and searches the problem database. Joshua argued that expending more effort in building the database could be a key for cheaper diagnosis. He mentioned that the main challenge lies in creating a database structure in which it is possible to meet user expectations.

Panel Discussion

Jeff Mogul was concerned about whether the system effected time to recovery. The author said there aren't critical time recovery requirements, as it is more important to get the data back. He pointed out that it is also possible to make it faster by adding redundancy. But it is a trade-off between storage and time. Somebody from MIT also had a question regarding the deployment of the Phoenix system. The author said it seems reasonable if there are enough hosts running diversified OSes. Ranjitha from UCSD asked Joshua Redstone what would be the motivation for people to fill the database. Joshua said an organization uses external support and, hence, people would find it easier to resolve problems at the cost of the effort. Mike Jones proposed an alternative scheme that asks users for requests and then

posts any solutions. Joshua was not sure how to maintain the database. Mike said, you can save the entire request and response and then infer offline. Jay Lepreau asked George Candea about the impact of crash-only software on throughput. George said it's likely to be lower, because of the inherently distributed approach (loose coupling, explicit communication, etc.) to building the system, but with time performance will improve, as in the transition from assembly languages to high-level languages. High-level languages enabled a qualitative jump in the types of software able to be written, even if the process was slower than writing in assembly. He also argued that "goodput" (throughput of successfully handled requests) is more important than absolute throughput.

SESSION: PERFORMANCE OPTIMIZATION

Summarized by Ranjita Bhagwan

Using Performance Reflection in Systems Software

Robert Fowler and Alan Cox, Rice University; Sameh Elnikety and Willy Zwaenepoel, EPFL

The main idea of this work is to use application-independent measures such as hardware instrumentation mechanisms and general system statistics to adapt system behavior. Performance indicators such as TLB misses and cache misses can be used to measure overhead, while bytes sent to a network card and flop rate can be used to measure productivity. Productivity and overhead are used to determine if the system needs to be tuned. Sameh Elnikety showed results on how server throttling of mySQL using the TPC-W workload succeeded in keeping the throughput at the maximum level while load increased, whereas, without using reflection, the throughput dropped at higher loads.

Cassyopia: Compiler Assisted System Optimization

Mohan Rajagopalan and Saumya K. Debray, University of Arizona; Matti A. Hiltunen and Richard D. Schlichting, AT&T Labs–Research

The main idea of Cassyopia is to combine program analysis and OS design to do performance optimizations. While the compiler has a local perspective of optimizations, the OS has a more general view. Cassyopia merges the two and tries to bring about a symbiosis between the OS and the compiler. An example of this is system call optimization, which profiles system call sequences and clusters them together using compiler techniques. The clustered system calls are called multi-calls. With OS support for multi-calls, performance can be improved and, according to preliminary results, significant savings obtained.

Cosy: Develop in User-Land, Run in Kernel-Mode

Amit Purohit, Charles P. Wright, Joseph Spadavecchia, and Erez Zadok, Stony Brook University

User applications are only allowed restricted access, which causes a lot of crossings of the user-kernel boundary. To prevent this, Amit Purohit proposed a compound system call (Cosy) in which one can execute a user-level code segment in the kernel. The authors have a modified version of gcc that uses Cosy. Kernel safety is ensured by limiting kernel execution time; x86 segmentation and sandboxing techniques can also be used. The performance benefits of Cosy have been evaluated, and 20–80% performance improvements are reported.

Panel Discussion

Mike Swift said that using all the compiler techniques from Cassyopia for kernel execution might be one way to proceed. Mohan Rajagopalan clarified that they do not plan to move any user-level code into the kernel, since he believed that interpretation in the kernel had high overhead. They primarily wanted to reduce the boundary-crossing cost, and, hence, the ideologies are not the same. Amit Purohit said that interpretation in the kernel is a bottleneck, and so they use a small interpreter. Checking pointers would cost a lot, and so they are using TLBs. This has some overhead, but it's a one-time check. Mohan brought up the point that the aim of Cassyopia is to apply optimizations that are quite obvious but have not yet been done.

Ethan Miller said that the TLB overhead in Cosy could be unavoidably high. Amit said that is true, but the savings they are getting are a lot more than the TLB overhead.

Margo Seltzer hit the nail on the head, by saying that what matters finally is the kernel-user API. All this work on extensible OSes and moving code across the boundary is probably done because the kernel-user API needs to be revisited. So let's fix the API. Applause.

Andrew Hume said that for the applications he has looked at, apart from zero-copy and I/O, there is not much to be gained by putting code into the kernel. Apart from the stated scenarios, the chances of using a multi-call are small. Sometimes, reassurance outweighs performance benefits. Mohan said that they are also looking at smaller devices, such as cell phones. All devices are resource constrained. In these cases, there is also the issue of energy savings apart from performance. Eric Brewer asked why, for small devices, such as in sensor networks, you would even want a kernel boundary. Mohan answered that cell phones and iPAQs can now have JVMs running on them. They are not targeting reprogrammable devices.

Matt Welsh asked why one would care so much about performance on an iPAQ.

Timothy Roscoe said that since there are different kinds of devices, there should be different OSes for them, and then the question would be where to put the ker-

nel boundary, if any, on them. Whether there is a generic answer to that question is still unclear.

SESSION: STORAGE 1

Summarized by David Oppenheimer

Why Can't I Find My Files? New Methods for Automating Attribute Assignment

Craig A.N. Soules and Gregory R. Ganger, Carnegie Mellon University

Craig Soules described new approaches to automating attribute assignment to files, thereby enabling search and organization tools that leverage attribute-based names. He advocates context analysis to augment existing schemes based on user input or content analysis. Context analysis uses information about system state when the user creates and accesses files, using that state to assign attributes to the files. This is useful because context may be related to the content of the file and may be what a user remembers when searching for a file. Google has proven the usefulness of context analysis; it chooses attributes for a linked site by using the text associated with the link, and it analyzes user actions after a search to determine the user's original intent in the search. However, the kind of information Google's context analysis relies on cannot be applied directly to file systems. In particular, information such as links between pages does not exist in traditional file systems, and individual file systems do not have enough users or enough "hot documents" to make Google-like context statistics useful.

Soules described access-based context analysis, which exploits information about system state when a user accesses a file, and interfile context analysis, which propagates attributes among related files. The former relies on application assistance or existing user input (e.g., file names), while the latter relies on observing temporal user access patterns and content similarities and differences between potentially related files and versions of the same file. Based on a trace analysis of usage of a single graduate student's home directory tree over a one-month period, Soules concluded that a combination of the techniques he proposes could be useful for automatically assigning attributes. For example, a Web browser can relate search terms to the document the user ultimately downloads as a result of the search; files created and accessed in a single text editor session can be considered related; and attributes about documents used as input to a distiller such as LaTeX or an image manipulator program can be distilled for attachment as attributes of the output file. Soules also found that examining temporal relationships between file accesses in the trace successfully grouped many related files.

Soules stated that as future work he is investigating larger user studies, mechanisms for storing attribute mappings, appropriate user interfaces, and how to identify and take advantage of user context switches, e.g., users moving from one program to another.

Secure Data Replication over Untrusted Hosts

B.C. Popescu, B. Crispo, and A.S. Tanenbaum, Vrije Universiteit, Amsterdam, The Netherlands

B.C. Popescu described a system architecture that allows arbitrary queries on data content that is securely replicated on untrusted hosts. This system represents an improvement over systems based on state signing, which can support only semi-static data and predefined queries, and systems based on state machine replication, which require operations to be replicated across multiple machines. In the authors' system, every data item is associated with a public-private key pair; the private key is known only to the content owner, and the public key is known by every client that uses the data. There are four types of servers: master servers that hold copies of content and are run by the content owner; slave servers that hold copies of data content but are not controlled by a content owner and thus are not completely trusted; clients, which perform read/write operations on content; and an auditor server, described later. The master servers handle client write requests and lazily propagate updates to slave servers. Master servers also elect one of themselves to serve as an auditor, which performs background checking of computations performed by slaves, taking corrective action when a slave is found to be acting maliciously. Slave servers handle client read requests; they may use stale data to handle requests, but clients are guaranteed that once a time parameter maxLatency has passed since a write was committed at a master, no other client will accept a read that is not dependent on the write. All content in the system is versioned; the content version of a piece of data is initialized to zero when it is created and is incremented each time the data item is updated.

The key challenge in building this system is to enable clients to feel safe having their queries handled by untrusted slave hosts. This is accomplished probabilistically, by allowing clients to send the same request to a (trusted) master and (untrusted) slave when they wish, and to compare the results. When a slave returns the result of a read, it attaches a signed "pledge" packet containing a copy of the request, the content version timestamped by the master, and the secure hash of the result computed by the slave. If the slave returns an incorrect answer, the "pledge" packet can be used as proof of the slave's malfeasance. This probabilistic checking mechanism is augmented by an auditing mechanism in which after a client accepts a result from a slave, it forwards the slave's "pledge" packet to a special auditor server. The auditor server is a trusted server that does not have a slave set and serves just to check the validity of "pledge" packets by re-executing the requests and verifying that the secure hash of the result

matches the secure hash in the packet. The auditor is expected to lag behind when executing write requests, executing writes only after having audited all the read requests for the content version preceding the write.

Palimpsest: Soft-Capacity Storage for Planetary-Scale Services

Timothy Roscoe, Intel Research at Berkeley; Steven Hand, University of Cambridge Computer Laboratory

Timothy Roscoe described Palimpsest, a "soft-capacity storage service for planetary-scale applications." Palimpsest is designed to serve as a storage service for ephemeral data from planetary-scale applications running on a shared hosting platform like PlanetLab, XenoServers, or Denali. Examples of the type of data to be stored are static code and data for services, application logs of various sorts, and ephemeral system state such as checkpoints. Despite the temporary nature of the data, it must be highly available during its desired lifetime, thus making single-node local disk storage unsuitable. Traditional file systems such as NFS and CIFS provide facilities unnecessary for planetary-scale applications, and don't meet service provider requirements such as space management, billing, and security mechanisms that allow users to store their data on a shared infrastructure without having to trust the infrastructure provider. Palimpsest aims to provide high data availability for limited periods of time, data protection and resistance to denial-of-service attacks, flexible cost/reliability/performance trade-offs, charging mechanisms that make sense for service providers, capacity planning, and simplicity of operation and billing. To achieve these goals it uses soft capacity, congestion-based pricing, and automatic space reclamation.

To write a file, a Palimpsest client erasure codes the file, encrypts each resulting fragment, and sends each encrypted fragment to a fixed-length FIFO queue at the distributed hashtable node corresponding to the hash of the concatenation of the file name and the fragment identifier. To retrieve a file, a client generates a sufficient number of fragment IDs, requests them from the block stores, waits until a sufficient number of the requested fragments are returned, decrypts and verifies them, and recreates the original file. Because the queues are fixed-length, all files stored in Palimpsest are guaranteed to eventually disappear. To keep a file alive, the client periodically refreshes it, and to "delete" the file, the client simply abandons it. The key to predictable file lifetimes is the "time constant" (T) associated with each block store; T measures how long it takes a fragment to reach the end of the queue and disappear. Clients piggyback requests for information about T on read and write requests, and block stores provide a recent estimate of T by piggybacking on responses. Palimpsest providers charge per (fixed-length) write transaction. Clients can use information about each block store's T value and fee per write transaction to flexibly trade off data longevity, availability, retrieval latency, and robustness. Clients pay providers using anonymous cash transactions. Denial-of-service attacks are discouraged by charging for writes. Providers can perform traffic engineering by advertising values of T that deviate from the true value. Congestion pricing is used to encourage users to attain an efficient write rate.

Panel Discussion

Andrew Hume asked Timothy Roscoe whether a simpler scheme than Palimpsest could be used to store ephemeral files just like regular files and cycle them using a generational scheme, eventually deleting the files that graduate from the oldest generation. Roscoe responded that Palimpsest provides an easy charging and pricing mechanism, while standard network file systems like NFS do not. Val Henson asked Popescu what he thought of the "High Availability, Scalable Storage, Dynamic Peer Networks: Pick Two" talk, in light of the fact that his system is targeted toward storing data on untrusted hosts. Popescu responded that they're more interested in environments in which hosts are less transient than in standard peer-to-peer networks. Jeff Chase observed that Palimpsest clients have no control over the placement of their data, and he asked Roscoe whether he thought that was significant. Roscoe responded that selecting specific nodes on which to store data could be provided by an orthogonal mechanism. Benjamin Ling asked whether widespread adoption of Palimpsest would be hindered by the lack of hard guarantees about data longevity. Roscoe responded that legal contracts akin to service level agreements could be layered on top of Palimpsest to ease users' concerns about the inherent risk of data loss. Furthermore, this is really a futures market: Third parties can charge premiums for providing guarantees and taking on the risk of data loss themselves.

SESSION: TRUSTING HARDWARE

Summarized by Matt Welsh

This session turned out to be the most controversial of the conference, as two of the three talks discussed the use of secure hardware and systems such as Microsoft's Palladium architecture.

Certifying Program Execution with Secure Processors

Benjie Chen and Robert Morris, MIT Laboratory for Computer Science

Benjie Chen is interested in the potential uses for trusted computing hardware other than digital rights management (DRM). Since all PCs may include this hardware in the future, he is interested in exploring the hardware and software design for such systems. His running example was secure remote login, such as from a public terminal at an Internet cafe, where the client machine can attest to the server that it is running unadulterated software (OS, SSH client, etc.). Of course, this does not preclude low-

tech attacks such as a keyboard dongle that captures keystrokes, but that is outside of the immediate problem domain.

Benjie presented an overview of the Microsoft Palladium (or Next Generation Secure Computing Base) architecture, which uses a secure "Nexus" kernel and a secure chip that maintains the fingerprints of the BIOS, bootloader, and Nexus kernel. A remote login application would send an attestation certificate, generated by Nexus and the secure chip, to the server. The issues here are how to keep the Nexus kernel small and how to verify the OS services (such as memory paging or the network stack). Some ways to improve Palladium's security and verifiability were discussed, such as using a small micro-kernel that allows attestation of all OS modules above it, as well as a flexible security boundary (where some, not all, of the OS modules are verified). There is a connection with the XOM secure processor work, which prevents physical attacks on DRAM by storing data in encrypted form in memory and only decrypting it into a physically secure cache. Borrowing some of these ideas, one could run the micro-kernel within the secure processor that authenticates all data transfers to DRAM; the application, hardware drivers, network stack, etc., could all be encrypted in DRAM.

Hardware Works, Software Doesn't: Enforcing Modularity with Mondriaan Memory Protection

Emmett Witchel and Krste Asanović, MIT Laboratory for Computer Science

Emmett Witchel proposed the use of efficient, word-level memory protection to replace the use of page- or segment-based protection mechanisms. This is motivated by the use of fine-grained modules in software, with narrow interfaces in terms both of APIs and of memory sharing. He argued that safe languages are not the answer, in part because it is difficult to verify the compiler and runtime system. Rather, allowing hardware protection to operate at the word level is much simpler and permits a wide range of sharing scenarios. For example, when loading a new device driver into the kernel, the MMP hardware would be configured to permit the module to access its own code and data, as well as to make calls to other modules and share very fine-grained memory (e.g., part of a larger data structure in the kernel). Some of the challenges involve cross-domain calls through call gates; dealing with the stack (as no single protection domain "owns" the stack); and automatically determining module boundaries through information already present in code, such as symbol import/export information in kernel modules. Other potential uses include elimination of memory copies on system calls, specialized kernel entry points, and optimistic compiler optimizations (e.g., write-protect an object and run cleanup code if a write fault occurs).

Flexible OS Support and Applications for Trusted Computing

Tal Garfinkel, Mendel Rosenblum, Dan Boneh, Stanford University

Tal Garfinkel's talk returned to the question of using secure hardware for applications other than DRM, and shared much of the motivation and background of Benjie's talk. The core problem with open platforms (as opposed to closed platforms such as ATMs and cell phones) is that applications can be deployed across a wide range of existing hardware but it is difficult to manage trust. Tal proposed the use of virtual machine monitors as a potential solution to providing a trusted OS environment. For example, the VMM can run either an "open box" VM (such as a standard OS) or a "closed box" VM (a trusted system).

One closed box VM might be a virtual Playstation game console, which prevents cheating in a multiplayer game through attestation. Another potential application could be a distributed firewall, where one could push a customized firewall into the VMM to protect the network from the host, by preventing port scanning or IP spoofing, for example, or enforcing connection rate limits. Tal also discussed applications to reputation systems and third-party computing (à la SETI@Home). He concluded the talk with a review of current efforts in this area, including TCPA, Palladium, and LaGrande.

Panel Discussion

The political issues surrounding trusted computing platforms raised a number of interesting – and heated – questions from the audience. Dan Wallach started off by describing the recent XBOX hack, where that system was supposedly trusted hardware. Tal and Andrew Hume countered that there are no guarantees of correctness, just trade-offs in terms of risk assessment. Mendel suggested that cheating at Quake was not a big concern for industry, so this was not of paramount concern in the XBOX hack.

Timothy Roscoe was disturbed that the three speakers seemed to be too much in agreement, and raised the question of big protection domains (i.e., VMs) versus tiny protection domains (i.e., Mondrian memory protection). They didn't take the bait, though, and Tal said that these approaches were not mutually exclusive.

Jay Lepreau raised the concern that nobody has yet demonstrated an entire (real) operating system based on the micro-kernel model. He again argued that MPP does not solve the whole domain protection issue, since control flow protection is just as important as memory protection. Emmett admitted that there is some complexity involved, but by making memory protection domains both finer grained and more explicit, programmers have to think about them more carefully and will document them in the code. Jay argued that chopping up a monolithic system in a "half-assed way" makes it more complex, but Emmett argued that most sys-

tems software is already making use of well-defined modules, simply without strong protection between them.

Val Henson returned to the trusted computing discussion and argued that these platforms were more useful for restricting rights (as with DRM) than for giving us more rights. Benjie argued that the bleak view is that this hardware is going to get pushed out regardless, so we should be considering how to use it for something other than DRM. Tal concurred and said that this work was also about intellectual honesty.

Jay argued that Tal and Benjie were really just giving cover to the real commercial driver for this technology, which is DRM. Mike Swift wanted to know where the line between research and the corporate world should be drawn, and whether our community should support this work at all. Tal mentioned again that their work is just bringing more information to the table, but Mike drew an analogy with nuclear weapons research, claiming that expanding knowledge is not the only side effect of this work.

Dirk Grunwald wondered whether calling this "trusted computing" was like the rebranding of "out-of-order execution" to "dynamic execution" – trusted computing cannot deal with hardware attacks, so consumers may be misled into believing it's more safe. Tal pointed out that this is no different from calling a protocol "secure."

Jeff Mogul made the points that technology tends to reinforce existing power structures, and that the issue with trusted computing is not about security but rather, whether the technology reinforces existing power relationships or diffuses them. He summarized, "Are you advocating a system that helps the powerful or that helps the weak?" Eric Brewer claimed that he didn't want "trusted" software on his PC, since that only enforces a particular kind of trust relationship between two parties. He would rather have control over his trust relationships, but trusted computing platforms remove that control. Tal admitted that there is a real concern about using Palladium as an industry control point. Timothy wrapped up the session by pointing out that this discussion had been rather "sterile" and had not touched on any of the issues of politics, lawmaking, or market forces surrounding the DRM and trusted-technology debate.

SESSION: PERVASIVE COMPUTING

Summarized by Ranjita Bhagwan

Sensing User Intention and Context for Energy Management

Angela B. Dalton and Carla S. Ellis, Duke University

Angela Dalton spoke about the use of low-power sensors to monitor user behavior and to reduce system energy consumption. She described a case study called FaceOff, which uses a camera to perform image capture. Following face detection, the information is fed into a control loop that does the system-level energy management, by turning off the display. The authors have built a prototype of this, which uses image capture and skin detection. The authors have determined that FaceOff can provide significant energy savings. The authors also describe various ways of increasing the system's responsiveness and describe several optimizations with that objective.

Access Control to Information in Pervasive Computing Environments

Urs Hengartner and Peter Steenkiste, Carnegie Mellon University

Locating people in a pervasive computing environment can be done at many levels and granularities. But there should be an access control mechanism for such information. Access control mechanisms for conventional information, however, cannot be used as is for such environments. Urs Hengartner described a people locator service that uses digital certificates for defined location policies. The authors use three design policies: identify information early, design policies at the information level, and exploit information relationships. Their approach is to use a step-by-step model, where validation of a client node is done at every server node. They also use a policy description language and an information description language in their people locator service.

Privacy-Aware Location Sensor Networks

Marco Gruteser, Graham Schelle, Ashish Jain, Rick Han, and Dirk Grunwald, University of Colorado at Boulder

Marco Gruteser gave a brief description of a system that uses sensor networks to gather information while maintaining some degree of anonymity. Describing the problem, Marco said that sensor networks could be used to identify precise location of certain entities, and this could be undesirable. With an anonymous data collection scheme, you do not need to negotiate a privacy policy, and the risk of accidental data disclosures is reduced, since databases do not store this information anymore. The author described the notion of k-anonymity in database systems, and said that it could be applied to location information.

Panel Discussion

Urs Hengartner asked Angela Dalton whether she had thought about turning off only parts of the screen if there were multiple windows open but only one active? Angela said that there is related work that deals with energy adaptive displays. Presently this cannot be done. But you could use some form of gaze tracking to do this. Andrew Hume asked whether the partial screen turn-off works at the hardware level. Angela responded that currently there is no hardware that does that. New kinds of displays are assumed. But you would still need to control it through the system.

Andrew Hume commented that this could be used for covert operations: for example, a laptop screen shutting down would indicate that there is no one close to it. Wouldn't the security aspect be that you could detect location to some extent? Angela said that larger networks could do this, and yes, there are lots of security implications.

Val Henson asked what the trend of built-in sensors is. Angela replied that most devices can now have built-in cameras. In general, these sensors are low-power, cheap and increasingly pervasive, especially the cameras.

Andrew Hume asked what kind of camera resolution is needed to make this work well. Angela said that the detection method currently is skin color based, and you could do this even with a low-resolution black-and-white camera.

Geoff Voelker asked Urs if he had thought of foreign users coming into an administrative domain. Urs responded that since they use SPKI/SDSI digital certificates, they do not require a PKI and have the benefit that they could give access to anyone.

Val Henson asked Marco how you decide what the shape of the location is. Marco said that the current assumption is that the sensors are pre-configured in a room which has a certain room ID, the same applying to a building, and so on.

Jay Lepreau asked Angela whether she had data on how people use laptops so she could evaluate how well her scheme would work with these user behavior patterns. Angela said that more and more people are using laptops as their primary system. Moreover, energy awareness is important generically. However, this question is valid for cell phones, PDAs, etc., which could have widely varying usage patterns. Jay said that the power management on his laptop is frustrating, because it is stupid. It would be good to have a diagnostic tool, with the user being able to guide the system to some extent. Has Angela considered providing the user with a diagnostic tool? Angela said that you can imagine a user interface, that could be used to measure the annoyance factor of the power management. Yes, there can be problems when things kick in at the wrong time.

SESSION: STORAGE 2

Summarized by David Oppenheimer

FAB: Enterprise Storage Systems on a Shoestring

Svend Frølund, Arif Merchant, Yasushi Saito, Susan Spence, and Alistair Veitch, Hewlett Packard Laboratories

Alistair Veitch of HP Labs presented "FAB: Federated Array of Bricks." He described a project that is aimed at making a set of low-cost storage bricks behave, in terms of reliability and performance, like an enterprise disk array, but at lower cost and with greater scalability. The FAB array is built from bricks, each of which consists of a CPU, memory, NVRAM, a RAID-5 array of 12 disks, and network cards. Clients connect to the array using a standard protocol such as iSCSI, Fibre Channel, or SCSI, and the storage bricks communicate amongst themselves using a special FAB protocol running on Gigabit Ethernet. The goals of the array are zero data loss, continuous availability, competitive performance, scalable performance and capacity, management as a single entity, online upgrade and replacement of all components, and higher-level features such as efficient snapshots and cloning. The principal research challenges are failure tolerance without losing data or delaying clients, asynchronous coordination that does not rely on timely responses from disks or the operating system, and the ability to maximize performance and availability in the face of heterogeneous hardware. The techniques FAB incorporates to achieve these goals are a quorum-based replication scheme, dynamic load balancing, and online reconfiguration.

After outlining FAB's goals and the high-level techniques used to achieve those goals, Veitch described the quorum-based replication scheme used to achieve reliability. It uses at least three replicas for each piece of data, and reads and writes a majority of replicas. It survives arbitrary sequences of failures, achieves fast recovery, and can be lazy about failure detection and recovery as opposed to needing to do explicit fail-over. The array configuration that the designers envision achieves a mean time to data loss of about a million years, which Veitch described as "at the bottom end of what's acceptable."

The Case for a Session State Storage Layer

Benjamin C. Ling and Armando Fox, Stanford University

Benjamin Ling presented "SSM, a recovery-friendly, self-managing session state store." SSM is a storage system specialized for storing session state typically associated with user interactions with e-commerce systems. The system assumes a single user making serial access to semi-persistent data. Ling explained that existing solutions such as file systems, databases, and in-memory replication exhibit poor failure behavior, recovery performance, or both, and that they are difficult to administer and tune. SSM is designed to be recovery friendly, meaning it can recover instantly without data loss, and self-managing in terms of handling overload and performance heterogeneity of components.

SSM is based on a redundant, in-memory hashtable distributed across nodes called bricks and on stateless client stubs, linked in with application servers, that communicate with the bricks. Bricks consist of RAM, CPU, and a network interface (no disk). SSM uses a redundancy scheme similar to quorums. A stub writes to some N random nodes (four in Ling's example) and waits for the first M of the writes to complete (two in Ling's example); this scheme is

used to avoid performance coupling. The remaining writes are ignored. Reads are issued to a single brick. SSM is "recovery friendly" in that no data is lost as long as no more than M-1 disks in a write quorum fail. Moreover, state is available for reading and writing during a brick failure. SSM is "crash-only," using no special-case recovery code; when a brick is added to the system or returns to service after a failure, it is simply started up without any need to run a recovery procedure. SSM is "self-managing" in that it dynamically discovers the performance capabilities of each brick, as follows. Each stub maintains a count of the maximum allowable inflight requests to each brick, using additive increase to grow this window upon each successful request and multiplicative decrease to shrink the window when a brick operation times out. If an insufficient number of bricks are available for writing, either due to failures or due to a full window, the stub refuses the request issued by the client of the stub, thereby propagating back-pressure from bricks to clients.

Towards a Semantic-Aware File Store

Zhichen Xu, Magnus Karlsson, and Christos Karamanolis, Hewlett Packard Laboratories; Chunqiang Tang, University of Rochester

Zhichen Xu explained that storage systems are in some ways an extension of human memory, but that computer-based storage systems have been "dumb" because, unlike humans, they do not associate meanings (semantics) with the data that is stored. For example, when humans search, they consider abstract properties and relationships among objects, and when they "store" data, they may group objects into categories or record only the differences between objects. Moreover, humans discover the meanings of objects incrementally. These observations motivate the incorporation of versions, deltas, and dependencies among objects stored in the semantic-aware file store.

The semantic-aware file store seeks to create a framework that captures and uses semantic information for fast searching and retrieval of data; stores data efficiently by using data compression based on semantic relationships of data; provides high performance through semantic data hoarding, data placement, replication, and caching; and enables highly available data sharing by balancing consistency and availability according to the data semantics. The semantic-aware file store uses a generic data model based on RDF to capture semistructured semantic metadata. The challenges Xu described are identifying the common semantic relations of interest, finding ways to capture semantic information, finding ways to handle dynamic evolution of semantics, and defining the tools and APIs that users and applications require.

Panel Discussion

Margo Seltzer asked Alistair Veitch whether in 10 years HotOS will have a paper explaining why FABs are just as expensive as today's storage arrays. Veitch answered, "I hope not, but who knows."

Andrew Hume asked Veitch whether the FAB design makes it difficult to predict performance as compared to a disk connected to a single machine, especially in the face of failures, due to the large number of potential interactions among bricks in the FAB. Veitch responded that the performance can in fact be modeled as a function of factors such as the overhead of doing a disk I/O, the number of nodes, the desired mean time to data loss, and so on. He added that even today no storage system gives you hard-and-fast performance guarantees, and that the more difficult question is how to model overall system reliability and the optimal trade-offs among design parameters.

Rob von Behren asked Veitch whether he had considered using non-RAID disks instead of RAID arrays inside each brick. Veitch responded that by using RAID, the mean time to data loss is controlled by the failure rate of each brick rather than the failure rate of individual disks, thereby increasing reliability. He added that it's very difficult to come by good numbers on actual failure rates of system components, so estimating overall reliability is difficult.

2003 European Tcl/Tk User Meeting

Nürnberg, Germany May 30 and 31, 2003

Summarized by Clif Flynt

The State of Tcl

Andreas Kupries, ActiveState

Kupries described the status of the current Tcl release as well as ActiveState's current offerings. He reported that the use of Tcl (based on downloads) is still growing, with several major manufacturers joining the Tcl ranks.

The 8.4 series is now officially closed to new features, though still open for maintenance. (New features are being added to the 8.5 release.) The 8.4 release included support for the Virtual File System, which allows a zip or tar file to be used as a file system, and supports mounting an HTTP or FTP site, several new high-level widgets (including paned windows, labeled frames, and a spinbox), support for 64-bit integers and file systems (even on 32-bit systems), and many performance improvements. The Tcl core has been Unicode compliant and thread safe since the 8.1 release. Performance was improved with the 8.4 release.

The Tcl maintainers are continuing to improve the Tcl core, and several TIPS are scheduled for inclusion in 8.5. The new dict data structure has been added already, and more options for the expr command and more high-level widgets are planned.

ActiveState has continued to support Tcl with the introduction of a Tcl Cookbook series of code snippets, with the Tcl Dev Kit software development tools, and by continuing to extend the Tcl support in the Komodo IDE, including incorporating a new GUI builder.

NPS Portal Manager and NPS Content Manager

Thomas Witt, Infopark

Witt described Infopark's content management system, NPS, and why and how they decided to use Tcl as the application scripting language.

The NPS package is written in Objective C. They extend the application's functionality by embedding the Tcl interpreter because:

- Tcl is open source, using the BSD license, and so is free for commercial use.
- The same Tcl code runs on all platforms.
- Tcl is a normal glue language, with no surprises for the casual programmer.
- It's easy to learn. This is particularly important when teaching Tcl to nonprogrammers.

Witt described the basics of integrating new commands into the Tcl interpreter and demonstrated how the new commands are accessed from a Tcl script.

One extremely useful feature of Tcl is the trace command, which allows NPS to support dynamic callbacks based on changes to a document. This is equivalent to using a database trigger.

Creative Use of the Text Widget

Arjen Markus

Arjen Markus described techniques for using the text widget for more than displaying text, highlighting the versatility of the text widget by using it to generate the PowerPoint-like slides for his presentation.

The text widget is one of the more sophisticated and powerful widgets in Tk. Not only can a script insert text into the widget, using a variety of fonts, foreground and background colors, and event bindings connected to the text, a script can also insert images and other Tk widgets.

Markus pointed out that the ability to put other widgets onto the text widget provides another technique for geometry management which is particularly well adapted to fill-out-a-form style applications. This makes it easy to automatically create a "Poor Man's GUI" from a description of the data in a form.

He also demonstrated an application of Knuth's "Literate Programming" paradigm by mixing text and code in the text widget. His example uses the text widget's bind command to trigger actions based on the text being inserted into the widget.

ORASTATE (Oracle Structure Tool and Table Editor)

Martin Fickert

ORASTATE is an application that generates a graphical representation of Oracle tables from a database or user interaction. Martin Fickert described how this package allows the user to perform database queries or edit tables, fields, and relations via a simple GUI.

This application is designed to allow users to interact with an Oracle database without needing to learn SQL and database commands.

This application represents a table as a drop-down list in which the top element is the name of the table and lower elements are the names of fields. During default display, the list is rolled up, and only the table name is displayed. Upon receiving an <Enter> (cursor over) event, the label "unrolls," displaying the table fields, which can be queried and modified via pop-up dialog windows. Relations between fields are displayed with lines, and can be added by the user drawing lines in the display.

Database queries can be created and evaluated via a set of pop-up windows that allow the user to select fields, operations between fields, and values for the operations.

UIE: Creating a Generic Interface Between Tcl/Tk and Other Languages

Veronica Loell

Veronica Loell described the User Interface Engine (UIE), which uses Tcl as an

engine rather than as glue. This framework supports multiple software agents, cooperating to provide a natural language interface with other applications.

The UIE framework has several cooperating processes, including a data manager, translators, and applications which operate by passing messages to each other via the translators.

The translation engines allow different applications to communicate cleanly by converting application-specific commands into a generic intermediate language and back.

Tcl Used for Testing in the Mobile World

Hans Banken

SIGOS GmbH uses a Tcl application to test mobile networks. Hans Banken described how the application is constructed and how it works.

The requirement was to simulate user behavior using real mobile equipment. The application needed to be able to exercise all the functions a user could request, utilize all available hardware, and be easily configurable.

Banken described the SITE system, a database-backed test engine capable of running 180 distributed test systems. This system uses Tcl for the GUI, glue, and database interactions. The tests are described in a Tcl-style configuration file that allows the Test Definition Language to be easily converted to Tcl scripts for execution.

Banken noted that with no previous Tcl experience the team was able to construct a large, robust, and successful commercial application in only two weeks.

dbWidget: A Rapid Application Development System for the Tcl Community

Franco Violi

Franco Violi described the dbWidget package developed to easily port legacy COBOL applications to modern environments. The name dbWidget is a misnomer, given that the package is more than an interface to databases, and more than a widget. This package allows a developer to define an application with a template file that is used to generate the GUI and database interactions. The applications inherit base functionality from the dbWidget package, including audio cues, online help, and support for printing.

The dbWidget package interfaces with Open Office to provide platform-independent printing, and uses a C database extension to interface to the Berkeley db, Faircom's Ctree, or other database engines.

This package was developed to upgrade their clients' applications. It's been in use for several years and is robust and effective.

XOTclIDE

Artur Trzewik

Artur Trzewik described XOTclIDE, an integrated development environment for XOTcl (Extended Objective Tcl). This IDE has a strong small-talk orientation and includes support for building components and version control.

Trzewik introduced XOTcl to the audience and noted that XOTcl uses a method-chaining style of inheritance and provides more support for introspection than [incr Tcl]. XOTcl also supports assertions, Mixins/Filters, and classes that contain metadata.

The XOTclIDE application allows the user to build an application in an interactive manner, constructing classes and methods bit by bit and testing them in a bottom up manner until the final top-level class (the application) is completed.

The XOTclIDE debugger is based on the atkdebugger, which allows the user to set breakpoints and step into/over procedures and modify the original source, making the edit/test/debug sequence more integrated.

The Version Control is based on a relational database (MySQL, Postgres, SQLite, ODBC) that works in the background, rather than requiring explicit check-in/check-out sequences. This paradigm supports a team-programming style, allowing a user to check at any time, via a code browser, to find out what modules are in sync with the repository.

I18n: Tcl for the World

Richard Suchenwirth

Richard Suchenwirth described Tcl's support for internationalization and localization of applications. He explained the evolution of Unicode, from the early Baudot code, through ASCII, EBCDIC, JIS, and UTF-8, and, finally, to the modern four-byte Unicode that can represent all known glyphs.

Suchenwirth demonstrated Tcl/Tk's facility with non-European fonts with a display from his iPAQ that showed similar phrases in several languages, including Chinese, Japanese, Korean, Greek, and Russian.

Since version 8.1, Tcl/Tk has used UTF-8, with support for up to three bytes per character, for internal character representation. Thus, a character may be a single or multiple bytes in the internal representation, but to a Tcl script each character is a single entity. He described using the msgcat package to support code localization and also how the text widget will automatically load fonts as required to display various glyphs.

He finished by discussing techniques for entering non-ASCII data from a standard keyboard. Combinations of shift and a letter can be mapped to other glyphs, or one could type the name of the character until the application recognizes the pattern and provides a menu of possible glyphs. Another option is to use a variant on greeklish, a style in which two character combinations are

used to represent non-ASCII characters. For example, *a*' would be used to represent the accented a character.

Generation R

Pascal Scheffers

Pascal Scheffers reported on the uses of Tcl by the Generation R project at the Erasmus University Medical Center in Rotterdam. Five university departments and over 20 external organizations are working together to track the lives of 10,000 children from pre-birth to young adulthood. This task involves many questionnaires and a great deal of information in many different databases maintained by the different groups working on the project.

The project is using AOLServer and the OpenACS package for much of the logistics and data management services. AOLServer is an HTTP server that uses Tcl as the scripting language for server-side processing. OpenACS is a package of tools for creating database-backed Web sites using AOLServer.

AOLServer provides a platform for building easy-to-use interfaces between the non-computer scientists using the data and the different database engines and data formats.

One continuing problem that the group faces is the need to generate customized form letters. After trying LaTex, PageMaker, and MS-Word, they decided the optimal solution was to save the basic form letter as a PostScript file and then massage the text with a Tcl script. This solution provides the fastest mail-merge and print options.

Tcl's string and list manipulation features are also used to parse a set of Health Level 7 data. The ISO Level 7 data format is a sequence of character-separated, nested lists with different separation characters for each level of nesting. The Tcl split command makes parsing this data easy. The commercial parsing packages cost about $6500. Developing a Tcl solution from scratch took two days.

Scheffers finished his talk by noting that while Tcl has easy support for many different database engines, the binary library support is not always easy to acquire. The assumption is that anyone who needs to use a database engine has the required libraries and C compiler for the platform they are using. In a multi-disciplinary project like this, that's often not the case.

Embedding the Tcl Interpreter for Portable Applications

Clif Flynt

The ease with which Tcl can be extended using other libraries and C code is well known and documented. Clif Flynt presented a tutorial talk describing how to embed the Tcl interpreter within a C application.

He described the Tcl library API that supports this, and demonstrated a short C/Tcl application that uses a C language main() function for the primary entry point and calculation function, and Tcl scripts for the GUI and data representation.

Flynt noted that this provides a "best of all worlds" development environment. The mission-critical, stable, platform-independent code is maintained in C, while the customer-experience, most-likely-to-be-changed, platform-dependent code is written in Tcl, where it can easily be modified to fit customer requirements. Because the user interface code is written in Tcl, it takes advantage of Tcl's multi-platform support, and what would otherwise be platform dependent becomes platform indepen dent as well.

This ability to embed the interpreter makes the Tcl/Tk libraries a good alternative to using the Xt, Athena, or MS libraries for project development.

;login:

USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710

POSTMASTER
Send Address Changes to *;login:*
2560 Ninth Street, Suite 215
Berkeley, CA 94710

PERIODICALS POSTAGE
PAID
AT BERKELEY, CALIFORNIA
AND ADDITIONAL OFFICES
RIDE ALONG ENCLOSED