;login:

THE MAGAZINE OF USENIX & SAGE

February 2001 • volume 26 • number 1

inside:

CONFERENCE REPORTS

USENIX & SAGE

The Advanced Computing Systems Association &
The System Administrators Guild

conference reports

This issue's reports are on the 4th Annual Showcase & Conference, (ALS 2000).

OUR THANKS TO THE SUMMARIZERS:
FOR ALS:
Peter Salus
Vikram V. Asrani
Laurel Fan
Thomas Naughton
Zhedong Yu

4th Annual Linux Showcase & Conference, Atlanta

Atlanta, Georgia, USA

October 10–14, 2000

GOODBYE, ATLANTA
by Peter H. Salus
<*peter@matrix.net*>

The Atlanta Linux Showcase was founded in 1996 by the Atlanta Linux Enthusiasts, which had been founded in December 1994. It grew too fast and this past October was run by the USENIX Association, rather than by the amateurs that had made it such a success in 1997, 1998, and 1999.

I was the dinner entertainment in 1998 and in 1999, so I guess I can get away with that rather dour beginning.

I enjoyed myself at ALS 4, but it wasn't the same. And it will be yet further transformed in 2001, when it moves from Atlanta to Oakland. ALS 4, in fact, was the Annual Linux Showcase, no longer Atlanta. . . . *Sic transit gloria mundi.*

The show floor, as expected, was bigger and better. There were plenty of really fine folks to talk to. Several of the invited talks (which I'll get to in a paragraph or so) were very interesting. But the tone was different and it will be more different in Oakland.

USENIX for nearly a decade was an amateur organization. It has changed. But even though I can generate nostalgia, the change and professionalization have been good.

Ken Coar, Director and VP of the Apache Software Foundation, spoke about life, Apache, and Open Source development on Thursday, 12 October. His descriptions of how Apache development works, what's hot right now, and software licensing were interesting, but somehow just didn't fire up the audience (nor me). It may have been the general "preaching to the choir" aspect. He did remark that the Apache license was "BSD-ish."

Coar's description of the HTTP project was more illuminating, involving XML and Djakarta (=Apache-Java).

Ken Coar

Last year there was an interesting ALS session on Beowulf, so, on Friday, I trotted along to hear Jim Reese (of Google) talk about Linux clustering. He referred to his talk as "Scaling the Web: An Overview of Google, A Linux Cluster for Fun and Profit."

Google has waxed tremendously. In June 1999 they had 500 CPU and half a million hits daily. In October 2000 (16 months later) it was 6000 CPU and 50M hits/day.

Jim Reese

Google uses cheap, off-the-shelf, PC hardware in which they replicate everything on there of Exodus' sites: Santa Clara and Sunnyvale (which are connected by OC12 lines) and Herndon, VA (which is connected by 2Gb lines to the West Coast. They run 80 machine clusters with four 1Gb uplinks per cluster. All 100% Linux.

They get 100 queries/sec.; have 500TB of storage; and tens of GB/sec of I/O.

Google runs redundancy like crazy. I wrote down lots more, but it was very impressive.

Hack Linux Track

Refereed Papers

Session: Kernel Performance

Summarized by Laurel Fan

Analyzing the Overload Behavior of a Simple Web Server

Niels Provos, University of Michigan; Chuck Lever, AOL-Netscape; Stephen Tweedie, Red Hat

Niels Provos analyzed phhttpd, a static Web server using a few different signal handling techniques.

Signal-driven I/O is traditionally done with the signal SIGIO, which is raised when I/O events (such as data received, data sent, connection closed) occur. With the siginfo_t struct and sigwaitinfo() syscall, information about what type of event occurred causes the signal. However, this doesn't work well with servers with multiple connections, since there is no information about which socket was involved.

POSIX Real-Time signals (RT signals) are an improvement over SIGIO in many ways. First, they allow a signal to be associated with a file descriptor. Second, signals are queued in the kernel, allowing true event-driven applications. However, the queue is fixed size and can overflow, in which case it falls back to SIGIO until the application clears the queue. The fallback is invoked by a SIGIO signal; the recovery process, however, uses poll() or select().

phhttpd is a static content Web server that uses RT signals. It is multi-threaded, with multiple threads that each use sigwaitinfo to process events one at a time. Load balancing is done by reassigning the listener socket to the next thread every time a connection is accepted. (This is possible because threads in Linux have unique pids.)

The authors implemented a new system call, sigtimedwait4(), which allows more than one RT signal to be sent at a time, similar to poll(). This can increase performance by decreasing the number of system calls and the number of passes through the RT signal queue.

To test the performance of sigtimedwait4, phhttpd was modified to use this and compared to the unmodified phhttpd. The overload behavior, the behavior of the server under the load of a large number of clients, was examined.

After a certain request rate, the performance of phhttpd, as measured by the reply rate, decreases dramatically. They found that merely switching to sigtimedwait4() gave only a small improvement, showing that the system call and signal handling are only a small part of the problem.

Another benefit of sigtimedwait4() is the additional information available. When a server is overloaded, it is too busy accepting new requests to take care of the old ones. With sigtimedwait4(), the server can detect when it is being overloaded and drop new connections in favor of completing old requests.

With this new enhancement, the reply rate no longer showed the steep decline when in overload. Instead, the reply rate leveled off and then decreased slowly. Another interesting result was that dropping connections actually decreased the error rate.

Further information is available at *<http://www.citi.umich.edu/projects/linux-scalability>*.

Linux Kernel Hash Table Behavior: Analysis and Improvements

Chuck Lever, AOL Netscape

Chuck Lever started work on this project while working on the Linux Scalability Project. Hash tables are a commonly used data structure in the Linux kernel because of their fast average insertion and look-up times, compared with lists and trees. Performance of the kernel depends on the performance of these hash tables. Lever wanted to know how the performance of these hash tables scales when moving from smaller memory systems to machines with large physical memory. He cited an example in which a particular hash function unexpectedly broke down (placed all items in only a few buckets) when adding a large number of items to the table.

Can one increase the size of a given hash table and expect the hash function to continue to work as designed? Lever examined the performance of hash tables used by the page cache, buffer cache, directory entry cache, and inode cache in the Linux 2.2.5 kernel. Using kernel instrumentation and the SPEC SDM benchmark, he was able to measure hash table behavior while controlling the offered load on the test system.

Experimental results showed that the hash function works well in the page, inode, and dentry caches, and scales well as hash table sizes increase. The buffer cache hash function was not sufficient to randomize the key, and long hash chains resulted. All hash tables in the 2.2.5 kernel were too small for large memory systems. The inode hash table was so small that the hash chains averaged more than 200 entries each.

Later versions of the Linux kernel implement a dynamic hash table size for these caches, based on the size of a machine's physical memory. Lever believes that in most situations, the table size generated by the Linux kernel is appropriate for good performance.

Lever then spoke about different hash function types. Modulus hash functions are generally expensive because they require a division operation. Table-driven hash functions are not practical because memory operations to read the tables are more expensive than computation on modern CPUs. Shift-add functions suffice in most cases but should be checked with real data prior to use. Multiplicative hash functions are mostly a reasonable

choice because they require only a few instructions, and are generally as good as modulus hash functions at randomizing the input data.

In conclusion, Lever mentioned that dynamic cache sizes provide good scalability. Keeping cache size small and relevant helps. Performance of hash functions depends on the input data set.

Further information is available at *<http://www.citi.umich.edu/projects/linux-scalability>*.

Dynamic Buffer Cache Management Scheme Based On Simple And Aggressive Prefetching

H. Seok Jeon and Sam H. Noh, Hong-Ik University

In this presentation, H. Seok Jeon described his proposed dynamic buffer cache management scheme to reduce I/O latency while incorporating prefetching into the replacement policy. As an overview, Jeon spoke about the various replacement policies described in the literature. He mentioned the three groups of policies which incorporated prefetching: a reference history-based approach, a hint-based approach, and a simple-minded approach using a one block look ahead. The LRU-OBL (one block look ahead) policy is simple and effective and can improve performance by up to 80%. However, its deficiency is that 60% of the blocks prefetched might never be used. Jeon then proposed splitting the cache into two partitions: a weighing room and a waiting room. All referenced blocks are to be placed in the weighing room, while the waiting room is used for the prefetched blocks. Since cache sizes are small, partitioning the cache could possibly result in a deteriorated performance due to the smaller cache size holding referenced blocks. Thus, it is essential to keep the size of the waiting room minimal. Jeon's solution to this was to use a self-adjusted room-size scheme. In the SA-WWR scheme the size of the two partitions is adjusted dynamically, based either on the reference interval for blocks or on the cache miss statistics. Enumerating the cases, Jeon explained how the sizes of the two partitions are modified depending on the positions of blocks i-1,i and i+1.

Jeon then spoke about the implementation in which he modified the bread() function in Linux to use the SA-WWR replacement policy and in which he added a FIFO wait queue. Experimental results by Jeon show that the SA-WWR scheme provided an improved performance as compared to both the Linux replacement policy and the LRU-OBL policy for replacement. He also considered multiple performances with CPU-bound processes before concluding that SA-WWR does provide an improved performance.

At the end, there was a question by Stephen Tweedie, expressing his surprise at the improved performance obtained by the experimental results, explaining that the bread() function was not the function used for sequential file access.

For more information, contact the presenter at *<hsjeon@cs.hongik.ac.kr>*.

Session: XFree86

Summarized by Zhedong Yu

Translucent Windows in X

Keith Packard, Xfree86 Core Team, SuSE Inc.

In X Window System, the core protocol defines which portions of each window are visible and which are not when overlapping happens. But the overlapping windows are always completely opaque. There are many techniques to simulate the non-opaque windows in controlled environments. But they could not be used in a general way to deal with translucency. Keith Packard talked about a general way to solve the problem by assigning alpha values for pixels in occluding windows. Thus the occluded region and the occluding region can be blended. This window-level composting extension will be greatly helpful for application development.

Developing Drivers and Extensions for XFree86-4.x

Dirk Hohndel, SuSE Linus AG; Robin Cutshaw, Intercore

Since XFree86 is the standard implementation of X Window System for PC UNIX systems, it's very important to be familiar with it and know how to develop drivers and extensions for XFree86.

In their paper, Dirk Hohndel and Robin Cutshaw analyzed the problems of previous XFree86 design: lack of a real design document; the device-dependent X (ddx) part was largely untouched and undocumented; the problematic assumption that the video card of PC should be VGA-compatible; and the logistical problem of one driver binary for one OS Device

Then, "Module Loading Architecture" was introduced to make XFree86 more extensible, allowing just the modified/new driver (or extension) module to be provided instead of the full X server. Thus all the OSs on the same hardware architecture can share the same type of modules as well.

Since XFree86-4.x is well documented, it is straightforward to implement a driver.

More information can be obtained from the XFree86 project at *<http://www.XFree86.org>*.

Session: Kernel Ports

Summarized by Laurel Fan

Linux on the System/390

Adam Thornton, Sine Nomine Associates

The System/390 is IBM's largest mainframe. Its strength is in I/O, rather than CPU power, making it suited for tasks such as Web serving. Running Linux on it would allow users with UNIX expertise to use the reliability and I/O power of the

S/390 without dealing with its less desirable characteristics, such as EBCDIC.

One interesting feature is VM, Virtual Machine. This allows a single S/390 to run multiple virtual S/390s, each running any OS, such as Linux. Using VM and Linux S/390, a large virtual server farm can be implemented on a single machine. 41,400 simultaneous copies of Linux have been run in a test environment, and 3,700 copies have been run in production.

This has many practical purposes. Multiple different versions of Linux can be run for testing or academic purposes, without the hassle or expense of multiple machines. ISPs or other service providers can give each of their customers their own virtual Linux server, without worrying about customers affecting each other. A single S/390 can have a lower total cost of ownership than the equivalent number of stand-alone servers.

Porting to the S/390 presented several unique issues, both because of the number of virtual machines and because of the S/390's unique architecture. One issue was the timer interrupt. In Linux, a timer interrupt fires 100 times every second by default. This can decrease performance significantly with many virtual machines. Their current solution is to decrease the frequency of the interrupt, which has the unfortunate effect of decreasing the responsiveness of interactive applications. A good solution to this would be for a virtual kernel to disable timer interrupts when idle, and later restore its time from the host kernels.

For more information, see *<http://www.linux390.com/>*.

A User-Mode Port of the Linux Kernel

Jeff Dike

User-mode Linux is a port of the Linux kernel that itself runs on Linux. It is a full Linux kernel, but instead of running on hardware, it runs on a host kernel. All devices are virtual, and most are implemented in terms of user-level objects. For example, disks are implemented as files, and terminals are implemented as xterms or ptys. The virtual processor is implemented with the kernel's arch interface.

Processes in user-mode Linux run as user-mode processes in the host kernel. Syscalls are implemented using a tracing thread which intercepts system calls and redirects them to the user mode kernel.

A port to user mode presents many challenges and design problems, which Jeff Dike addressed in his talk, such as context switching and virtual memory.

A user-mode kernel has many applications. For example, it can be used as a sandbox for untrusted code, debugging, and as a Linux binary compatibility layer for othere OSes.

While user-mode Linux is quite functional, supporting kernel modules, X clients, and networking, some work still needs to be done, such as SMP support, privileged instruction emulation, and nesting.

For more information, see *<http:/user-mode-linux.sourceforge.net/>*.

Session: PotPourri

Summarized by Laurel Fan

GCC 3.0: The State of the Source

Mark Mitchell and Alexander Samuel, CodeSourcery, LLC

GCC, the GNU Compiler Collection, is the primary compiler for GNU/Linux and an important part of the system. The next major release, GCC 3.0, will include many improvements, and will have a more rigorous quality assurance process.

One major improvement is a standardized C++. ABIGCC 3.0, the next major release of the GNU Compiler Collection, will include a standardized C++ ABI, a major improvement. The ABI, application binary interface, defines how the object code is laid out. Because the C++ ABI has changed between GCC releases in the past, libraries built with different versions of GCC are incompatible. A stable ABI for 3.0 and subsequent releases will make distribution of both free and proprietary C++ libraries easier.

Many other C++ improvements have also been made. Mangled names, especially for complex templates, are much shorter, resulting in smaller object files. Virtual bases are handled more efficiently. A new, more standards-compliant C++ standard library will be included.

The infrastructure of the compiler itself has also been worked on. Better internal memory management allows GCC to use less memory. Many improvements, such as creating a parse tree for a whole function and using flow graphs to allow global optimization, will enable better optimization techniques.

For more information, see *<http://gcc.gnu.org/>*.

SMP Scalability Comparisons of Linux Kernels 2.2.14 and 2.3.99

Ray Bryant, Bill Hartner, Qi He, and Ganesh Venkitachalam, IBM Linux Technology Center

This study compared the SMP scalability (the performance gain from adding more processors) of Linux kernel versions 2.2.14 and 2.3.99. At the time of the study, these were, respectively, the newest stable version and the newest development version, which should have similar performance characteristics to the 2.4 series.

Four benchmarks were used: Volanomark, Netperf, FSCache, and SPECweb99. Volanomark is a chat-room server and client written in Java that makes extensive use of threads and measures scheduler and TCP/IP stack performance. Netperf measures network performance. FSCache measures the

performance of the file system cache. SPECweb99 is a benchmark designed to test a Web server by simulating clients accessing static and dynamic content.

On all of these benchmarks, the 2.3.99 kernels showed a significant increase in SMP scalability. Consequently, the upcoming 2.4 kernels should have better SMP performance than the 2.2 kernels.

Session: Security

Summarized by Laurel Fan

Enhancements to the Linux Kernel for Blocking Buffer-Overflow-Based Attacks

Massimo Bernaschi, Istituto Applicazioni del Calcolo, Italy; Emanuele Gabrielli and Luigi V. Mancini, Universita di Roma "La Sapienza," Italy

Subverting privileged applications using a buffer overflow or similar attack is a major security problem on Linux and other operating systems. Existing solutions, such as adding bounds checking and using a non-executable stack, require modification of application code, break legitimate applications, or can be bypassed.

The objective of this approach was to create a solution that has minimal impact on the kernel – requiring no change or recompilation of applications and minimal performance penalty – and is easy to set up.

The technique described here involves making a check on an Access Control Database (ACD) when a controlled system call is invoked by a privileged process. Controlled system calls, such as open, execve, and chmod, are those that an attacker could use to gain control of the system.

When a system call is invoked, the ACD entry for that particular system call is examined. The information contained in the ACD varies with each controlled system call. For example, the ACD entry for execve contains information about which executable files each privileged process is permitted to execve, and stored information (such as last modified date and size) about the executable files. A call to execve fails if either the process does not have permission to execute the file, or the target file has been modified since the ACD entry was created.

This feature is implemented with a small kernel patch, a new command to manage the ACD, and a change to chmod. The performance impact is limited because the new functionality is not accessed often: only for the controlled system calls and never in user mode. This approach has been shown to protect against several buffer-overflow-based attacks.

For more information, see *<http://www.iac.rm.cnr.it/newweb/tecno/indexsecurity.htm>*.

Domain and Type Enforcement for Linux

Serge E. Hallyn and Phil Kearns, College of William and Mary

Domain and Type Enforcement is a method of access control for protecting the system from a trusted user. Processes belong to "domains," and files belong to "types."

Access is controlled from domains to types (processes of read/write/execute /etc. files) and between domains (sending signals and changing domains). A process can change domains explicitly or automatically by executing a file defined as an entry point.

For example, an ftp daemon can be prevented from giving up a root shell by making the ftpd binary an entry point into a domain that does not have permission to execute system binaries or change domains.

Hallyn talked about his implementation of DTE for Linux kernel 2.3.38. The DTE policy is contained in a file which is read read on bootup, and which contains information about domains, types, and permissions. This information is stored in memory, and when a process attempts to access a type (by calling open), access a domain (by calling signal), or change domains (with execve), a DTE check is done.

There is a slight performance impact with adding DTE to the kernel. Adding DTE to the kernel slows performance slightly, but for normal workloads, in which executing files is rare, this should be relatively insignificant.

For more information, see *<http://www.cs.wm.edu/~hallyn/dte>*.

Piranha Audit: Kernel Enhancements and Utilities to Improve Audit/Logging

Vincenzo Cutello, Emilio Mastriani, and Francesco Pappalardo, University of Catania, Italy

Auditing and logging is an important part of system security. If you can detect an attack when it's happening, you might be able to stop it. Even if the attack succeeds, audit data can help you decide what to do to prevent it from happening in the future. Auditing is described in TCSEC, a set of criteria for secure systems. Piranha Audit is an attempt to meet those requirements.

One problem with existing logging systems is that in a root compromise situation, the logs can be altered to hide the intrusion. Piranha Audit's solution to this is to take steps to protect the logging system in the kernel. With Piranha, some tasks, such as signaling the monitoring task or editing such important files, such as the audit log and the Piranha binaries, would require both root access and an additional password.

Another problem is that the audit log can become too long and contain too much unimportant information for a human to read through. Piranha's solution to this is to provide intrusion detection tools to analyze the logs and find attacks, either to alert an administrator or to take action itself.

Testing has shown that Piranha Audit can detect and prevent attacks, and does not cause excessive performance degradation.

Session: Kernel Performance II

Lockmeter: Highly Informative Instrumentation for Spin Locks in the Linux Kernel

Ray Bryant, IBM Linux Technology Center; John Hawkes, SGI

Summarized by Vikram V. Asrani

Ray Bryant introduced spin locks as the low-level synchronization primitives in the SMP (symmetric multiprocessing) system. The two types of spin locks are spinlock_t and rwlock_t, the latter supporting multiple read/write access. These locks are operated upon using macros. Bryant believes that the primary reason for the use of Linux in the market is so that it can be used as a server operating system. However, vendor systems in the variants of UNIX provide a better performance for SMP. Thus there is a need to improve Linux SMP performance.

Bryant described path length and lock contention as the two main issues determining SMP performance. Path length can be examined using profiling tools. However, measuring lock contention was a harder task.

The reasons are as follows: Linux has a fast implementation for locks. Gathering statistical information can potentially increase the overheads, and one wants to keep this overhead minimal. Since the lock structures have been specifically designed to optimize their performance in the presence of a cache, one cannot increase the size of the lock structure.

One way to reduce the overhead of lock instrumentation is to store all lock statistics in per-CPU data structures. This has the advantage of not introducing additional cache traffic between processors that would occur if there were a single lock statistics structure shared among CPUs. Additionally, there is no need to lock the statistics structure, since it is only updated by one CPU.

In short, Bryant believes that the instrumented code should study the original problem and not deviate to examining the instrumented problem.

Bryant then described their solution: the Lockmeter, which is a set of instrumented spin-lock routines providing certain lock usage statistics on a per call basis. He described the implementation of both spin locks as well as the rwlocks using the idea of saving a hash index in some field in the lock structure. Bryant showed a large set of useful statistics provided by the Lockmeter. In addition, the authors had also examined the overheads introduced by this instrumentation. This instrumentation increased system time up to 20% and system throughput up to 14% (because of the larger set of instructions to be executed). Bryant mentioned that they have examined the problem and have gotten some results. They now need to use the results in order to examine why the SMP performance does not scale; they will be continuing work on this. The current version of Lockmeter is available from <*http://oss.sgi.com/lockmeter*>. An updated version of the Lockmeter paper can be found at <*http://oss.sgi.com/projects/lockmeter*> or <*http://oss.software.ibm.com/developerworks/opensource/linux*>.

EXTREME LINUX TRACK

Session: Potpourri

Summarized by Thomas Naughton

The Linux BIOS

Ronald G. Minnich, James Hendricks, and Dale Webster, Los Alamos National Laboratories

The session chair, Donald Becker, introduced Ron Minnich and mentioned that he was working with clusters when they (Becker, et al.) began the Beowulf project at CESDIS in the early 1990s. Minnich briefly introduced several of the clusters they currently are working with, making note of the various manufacturers as well as multiple BIOSes. These included Pancake: 36 Compaq Photon nodes, Rockhopper: 128 Intel L440 GX+ SMP, and Sarnoff: 161 various types. He explained the current quandary regarding BIOS and the lack of a sufficient standard. The major vendors like Intel, DEC, Compaq, Dell, all have different BIOSes and the availability of specifications also varies.

Since no standard is present he discussed a few options, one being to use a free BIOS, but these currently lack the necessary maturity to make this a realistic option. Minnich also noted that the proposed Intel standard PXE leaves much to be desired (or reduced, given the oversized technical docs). In light of these issues he asked the question, "Can we get out of the BIOS mess?" Can Linux cold boot Linux? As it turns out, the answer is yes. They can build a 32K hardware startup program and then unzip the kernel. They can thus gain control of the machine from power on instead of having to deal with intermediate software.

The key question for LinuxBIOS was – Why? The ability to gain control over previously BIOS-managed matters offers several attractive options, such as allowing log buffers to survive for diagnostic information where they usually get zeroed out by default. Possibly the most stunning point was his demonstration of booting to a single user in 3 to 5 seconds and approximately 10 seconds to reboot SMP. Also impressive was the fact that there is no proprietary license and no more hangs for hit <F1>, etc. And on a purely geeky note, they are able to one-up the DEC BIOS's "tinky Yellow Rose of Texas" by being able to play an MP3 – from the BIOS!

The LinuxBIOS is working on select Intel, SiS, and VIA boards. It is not currently working on Acer and RCC (Dell/Compaq use this), mainly due to a lack of available details from RCC. But

Dell and Compaq are trying to help with information.

The closing summary mentioned the following: own node from startup; no Band-Aids for BIOS defects; node behavior like you want; don't need working floppy, CD-ROM, or disk; and every node, regardless of vendor, will boot the same way.

The first question from the audience was aptly, "How many boards have you toasted?" Minnich's response, "Five . . . all Intels." He also confirmed that Linux BIOS could be used for embedded devices. Currently there is no support for Power Management. He noted that they might possibly go the route used by a FreeBSD venture to move this to the kernel. The issue of security between reboots for multi-user environments was briefly mentioned and noted as something that could easily be handled by possibly zeroing memory upon reboot when desirable. Another point that was mentioned was the difficulty in maintaining support for the ever-growing number of mainboards from vendors. Minnich explained that they are targeting clusters, and hence support of a subset of boards should be reasonable. Also, their end goal is to have manufacturers participate in support as has been the case with the code contributions for the SiS port.

Further information about LinuxBIOS can be obtained from <*http://www.acl.lanl.gov/linuxbios/*>.

KLAT2's Flat Neighborhood Network

Hank Dietz and Tim Mattox, University of Kentucky

The new cluster at the University of Kentucky, KLAT2 (Kentucky Linux Athlon Testbed 2), offers some very interesting results. The presentation discussed the new network architecture they have developed for use with this cluster. The costs for connecting the 64 nodes of the cluster using other popular topologies caused them to investigate this new architecture. The infeasibility of connecting all the nodes on a single inexpensive switch prompted the development of the Flat Neighborhood Network (FNN) topology. This allows multiple NICs per node to be used to attach the nodes to several switches for full connectivity while still maintaining low cost and high performance.

The difficulty in configuring many nodes with multiple NICs and switches prompted them to make use of a Genetic Algorithm (GA) to assist with the configuration and construction of the interconnection network. The GA is used to optimize the network so that the routing and wiring can be produced automatically. The output from the GA is a color-coded wiring diagram as well as the routing tables that are used for each node. The GA is also used to optimize the networks so that a minimum number

Keynote Speaker Larry Wall and Theodore Ts'o

of NICs are used (not all nodes need the same number of NICs). The GA can optimize for a specific program's constraints, however the default properties of FNNs appear to be sufficient for most cases.

The total cost for the 64-node KLAT2 network was ~$8,100. Dietz and Mattox have seen significant price-to-performance results from KLAT2 with the FNN. They are currently a finalist for a Gordon Bell Price/Performance award for their results on a full CFD (Computational Fluid Dynamics) code. (They obtained $2.75/MFLOPS and $1.86/MFLOPS price/performance for double and single precision, respectively.)

Mattox's concluding remarks pointed out that FNNs offer a substantial performance increase as well as a significant price reduction for a sound interconnection network. They offer several tools at their Web site for working with FNNs, including a CGI that can be used to demonstrate the GA that is used for configuration/design.

A member of the audience raised the issue of increasing the number of NICs per PC; the response was that there is not much payoff other than connectivity, which they already manage. Also, using more than four or five NICs at once would exceed the current PCI bandwidth of most commodity PCs. Another audience member commented that some of the switch has been moved to the node, and this appears to be a cost tradeoff. The response was that the routing and NICs are already there; why not make use of it? A question regarding IP addresses/NICs was asked, and Mattox explained that currently each NIC has a different IP and the switch is acting as a "subnet." He also mentioned that there are issues with exceeding arp cache if they try to get to all nodes. A question about cabling elicited the interesting point that often they do not have to recable but rather do the rerouting through software (from the GA). They can recable everything if needed in a reasonably small amount of time, but it's not something you want to do every week. A final question about locating faulty network cables was asked, and Mattox said they generally use ping and ifconfig to locate faulty network hardware.

Further information about FNN can be obtained at <*http://aggregate.org/FNN/*> with other related information at the root of the site.

The Portable Batch Scheduler and the Maui Scheduler on Linux Clusters

Brett Bode, David M. Halstead, Ricky Kendall, and Zhou Lei, Ames Laboratory; David Jackson, Maui High Performance Computing Center

Summarized by Vikram V. Asrani

At the start of this talk, Brett Bode introduced batch systems. He spoke about the two classes of parallel aware schedulers: namely, cycle stealers and dedicated system schedulers. The Portable Batch Scheduler (PBS) is one example of a dedicated system scheduler. Schedulers must be stable, portable, and should provide efficient resource management. In his opinion, PBS is probably the most commonly used and probably the best solution available. The Maui scheduler was originally used on HP systems and performed well. The PBS scheduler is a FIFO-like scheduler, scheduling jobs in a FIFO order, except when the first task in the FIFO queue is blocked by another task. The PBS system prevents starvation using a starving job mechanism.

Bode then provided an overview of the Maui scheduler on Linux clusters. It is fully parallel aware, as it knows about the attributes, memory, and utilization of each node. It is a time-based reservation system, and idle nodes are back-filled with small jobs. Bode then described the scheduler test for the PBS and Maui scheduler on a 64-node cluster of Pentium Pros. The simulation profile consisted of large, medium, and small debug and failed tasks. The results with backfill turned off showed that the Maui scheduler provides a better processor usage. The Maui scheduler required five hours less to complete the tasks for which a sequential execution processor took between 90 to 100 hours.

In response to a question on what happens when a node fails, Bode informed us that PBS does not restart the node and that this was indeed a problem. The server daemon simply hangs and other mechanisms need to be used to restart. Another person from the audience asked about the ability to perform progress migration on clusters. Bode responded that no such mechanism existed on PBS. To a question on what happens when the user's processor utilization time has reached the allotted amount, he answered jobs are killed. In addition, PBS generates a signal five minutes before the deadline. On PBS, users can also alter job request times.

Panel: Has Cluster Administration Been Solved?

Moderator: Rémy Evard

Participants: Susan Coghlan, Turbo Linux; Richard Ferri, IBM; Brian Finley, VA Linux; Greg Lindahl, HPTi; John-Paul Navarro, Argonne National Laboratory; Lee Ward, Sandia National Laboratory; and Stephen Scor, Oak Ridge National Laboratory

Summarized by Vikram V. Asrani

The organizations represented on this panel are working on clusters of various sizes ranging up to 1,500 node clusters.

The first question posed by moderator Evard to the panelists was, "Why does everyone have their own cluster solutions? Will we ever reach a state when a single common solution will exist?" Greg Lindahl responded that since people have different specific requirements, they develop specific solutions. Another panelist concurred, adding that clusters with more than 64 nodes had specific requirements and, hence, vendors developed their own solutions. One of the panelists thought that it was essential to come up with a common solution. Susan Coghlan provided an analogy for this problem with enterprise management. She said that one required flexible tools to meet everybody's needs. However, the present-day tools did not even do all that they were supposed to.

Evard then asked the panelists, "Is the cluster architecture dependent on the computing model in the system administration solution? If yes, how should it be changed?" One of the panelists answered that it was a matter of getting the tools to work with the clusters, and the tools (rather than the clusters) needed to be tweaked. Another panelist asked whether the same set of tools could be used for clusters with 32 nodes and clusters with more than 32 nodes. The same set of APIs should exist, was the opinion of one panel member. Coghlan mentioned that the tools required to manage small and large clusters are bound to be different since the complexity lies essentially in the tools. Lindahl found out from the audience that only ~20% of the audience ran clusters with more than 64 nodes.

Evard posed the next set of questions: "What are the biggest scaling issues in system administration? What scaling problems have the panelists run into? Why do the panelists consider clusters with more than 64 nodes large? Where do large clusters stress the existing tools?" Answering the question on scalability, Lindahl clarified that the use of a database for cluster administration was a gross mistake. Brian Finley pointed out that tools to automate tasks was one of the main scaling issues.

The next question to the panelists was, "What is the right community approach for cluster administration? Should the plan be to (a) depend on vendors to provide solutions (which may be proprietary or otherwise)? (b) converge on a set of tools that everyone else uses (presumably open source)? (c) try to maintain a good mix of solutions, keeping them environment-rich in competitive variability? or (d) continue to build their own solutions?" As before, one panel member suggested the development of a standard API set. However, some of the panel members were in favor of the open source set of tools licensed under GPL. Finley suggested that if a tool breaks in a particular

users environment, then, with the open source, one can fix it and everybody benefits. Lindahl suggested that currently there is no market for cluster vendors to provide specific tools. Finley suggested that this market will soon exist. His opinion was supported by the audience, who cited this as something to work toward in the future.

The final question put to the panel members was, "What do you wish you could do with the existing sysadmin tools that you cannot do?" One panel member suggested active diagnostic management. However, Finley said that it all depends on how much money you are willing to spend, how much your customer wants, and what your customer says. Tool construction is heavily customer dependent.

Valerie Cox and Ve Martin of ALS

Illiad signing his book for his fans

Video game room in action

Ted T'so giving Best Paper Award to Robert Ross

There are Old Farts even at ALS . . .