;login:

THE MAGAZINE OF USENIX & SAGE

August 2002 volume 27 • number 5

inside:

CONFERENCE REPORTS

2002 Linux Kernel Summit

USENIX & SAGE

The Advanced Computing Systems Association &
The System Administrators Guild

2002 Linux Kernel Developers Summit

June 24–25, Ottawa, Canada

Summarized by David B. Sharp

This year's two-day summit featured participants armed with laptops of every size, feature, and brand. The summit was followed by the four-day Ottawa Linux Symposium, with many of the talks and presenters following up on topics discussed during the summit.

Requirements for Carrier Grade Linux

Timothy D. Witham, OSDL

Timothy presented the requirements for carrier grade Linux. Carrier grade service, as required by the telecom sector, differs from enterprise grade service in that enterprise networks typically service networks with nodes that number in the thousands. Carrier grade networks, however, service nodes that may number in the millions. Thus, an interruption of a carrier grade network can be very costly, especially when service agreements with clients include penalties. Service levels usually demand 5 9s of reliability.

OSDL outlined a list of priorities they felt were required to achieve carrier grade performance from a Linux-based network. Some of these include:

- Compliance with LSB, POSIX, Ipv6, IPSec MIPv6, and all three SNMP versions.
- SCTP: this is a new streaming protocol built on top of IP and sister to UDP and TCP.
- Boot-cycle detection to sense frequent reboots.
- Hot-swapability.
- Alternate boot selection; if not A, then B.
- Six 9s availability.
- A hardened driver specification.
- Application heartbeat monitors.
- A watchdog timer pre-interrupt. This would allow systems and applications to be notified prior to a timeout by a watchdog timer.
- Live upgrades and rollbacks. This would allow the state of a system to be captured and, in the event of failure of an upgrade, to be rolled back to its former stable version.
- Better serviceability. This would allow a system in the field to be serviced remotely with tools such as dynamic debug and probe insertion.
- Device enumeration of hot-plugged devices.
- The ability to service a component without having to take the whole system offline.
- Forced umount.
- Better user-level tools. This would allow users without significant kernel knowledge and experience to perform triage.
- Kernel profiling and system tools.
- Kernel preemption.
- A selectable scheduling policy framework and O(1) performance to better service soft realtime applications.
- Support for a high number of timers, threads, and processes. This item was an acknowledged problem, but such support would allow for easier porting of existing applications and models.

Many of the items listed above are currently in progress.

AMD Hammer Port (AMD)

Wayne Meretsky and Vojtech Pavlik, AMD

Wayne and Vojtech presented AMD's new Hammer series of 64-bit processors and outlined what would be required to implement Linux. Unlike Intel's implementation of the Itanium, with its entirely new architecture, the Hammer chip extends the IA-32 and adds two new instructions. The features for the Hammer include:

- Larger L1 and L2 caches and better branch prediction characteristics.
- Double the number of general purpose registers.
- A greater number of integer and floating point functional units. This and the previous two features allow, according to AMD's preliminary tests, 32-bit applications to run up to 20% faster.
- A "Hyper-Transport" (see Figure 1) high-bandwidth point-to-point interconnect built into the chip. This allows the Hammer chip to connect to other chips and devices. AMD stressed that this was not a replacement for PCI.
- Four page levels for memory access and a larger Translation Look-Aside Buffer (TLB). This differs from the current architecture, which uses three page levels. The first implementation will allow 512GB of 48-bit addressable memory.
- Memory controller support for up to 333MHz DDR memory and future support for DDR2 memory.

Kernel Parameters

Rusty Russell and Patrick Mochel, OSDL

This discussion looked at some of the issues with boot parameters; these are used at boot time or during module loading to specify various kernel behaviors.

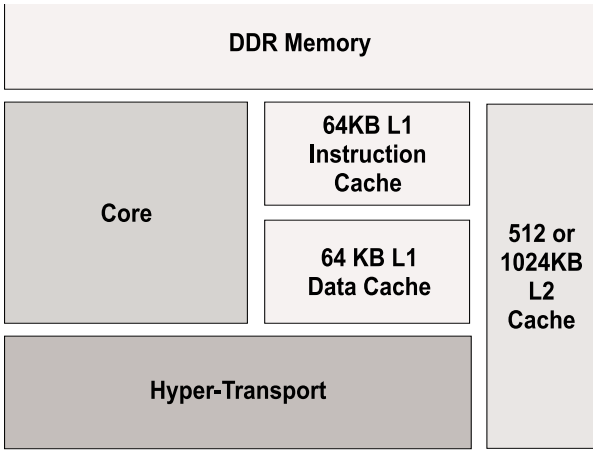

Figure 1: AMD Hammer Architecture

Rusty presented the problems with the MODULE_PARAM, one of which is there is no type checking. This would be replaced with a macro, PARAM(), which defines a callback for new variable types.

Patrick discussed the driverfs/. This is a virtual file system, similar to the /devices file system, used for debugging purposes, and allows the export of kernel device attributes in user space. The driverfs/ differs from the /proc tree in that there is one value per entry. One problem noted with the /proc tree is that it is not centrally maintained. The driverfs/ would enforce a strict interface, which should eliminate some of the anarchy in the 9 directory.

It was suggested that the driverfs/ could be used for event notification and that select polling would be easy to build in. Linus countered that this does not map to a lot of events you would want to have and that it would not be a convenient interface.

The driverfs/ prompted a lot of discussion, in particular regarding what belongs in the driverfs/, what doesn't, and where the latter should be exported.

Living with Modules

Rusty Russell

Rusty presented a candid and animated discussion on the problems with loadable modules. He started by stating that a module's only purpose is to add hardware.

The first problem he noted is with module implementation. A poorly written interface is almost impossible to use; the intended function needs to be clear. As an example of what constitutes a good interface, Rusty posed the problem of a module failing during init. He suggested that a kernel module linker be used instead of the insmod user space routine.

Initialization problems were then discussed. Most code assumes it is run at boot time; if there is a problem at this point, error code is often nonexistent. Someone asked about dropping module versioning for 2.5, but it was pointed out that this would be a nightmare for the vendor. Module versioning also deals with issues such as SMP and non-SMP modules.

One proposed initialization solution was to break the kernel registration into two parts. The first part would involve *reserving* the resources required by the module. If this stage were successful, the second stage would involve committing these resources with a use function call. This would solve the problem of bad entries in the proc/ directory.

Another problem is that there is no way to force the removal of a module; for example, the module may always be in use. A solution for this was another two-stage clear-and-destroy approach. The first stage would involve a reference count for each module. When we want to remove a module, no new users will be allowed to reference the device. When the reference count reaches zero, then the module can be destroyed.

Virtual Memory

Andrea Archangeli, Daniel Phillips, and Rik van Riel

The first topic discussed in this presentation was reverse mapping of memory with a new function call to rmap(). This function would allow mapping of physical pages to page tables. There was some discussion about whether the overhead of rmap() would be worth it. With terabytes of memory, VM management is too complex and takes too much CPU time to manage page tables. By including reverse mapping, it was pointed out, VM balancing would be possible.

There was some discussion about how to benchmark systems with such large memory and whether dbench was a fair measurement tool. We need a tool to quantify measurements and evaluate changes against different workloads. Further, we need to agree on the different benchmarks and tests. If the "right" set of tests existed, then the process could be automated.

Another issue presented was accounting and out-of-memory conditions. We still cannot accurately measure an out-of-memory condition, i.e., how many pages are free. There is a need for a heuristic algorithm to provide proper memory balancing and an intelligent way to kill a task to free up memory. This algorithm must efficiently keep track of how much memory is currently pinned. The problem now is that failing to do so properly can lead to system deadlock.

Miscellaneous VM issues included:

- NUMA scalability. A suggestion to reduce the overhead was to put pages into groups of four, producing one-fourth of the page structs.
- Dealing with very large page tables. By using page coloring, an increase in performance of 300%–400% in page allocation and a complexity of O(1) can be achieved.

Block I/O

Jens Axboe

Block I/O is a big topic in 2.5. The first topic discussed dealt with ordered writes and barriers. Ordered writes using barriers are required to maintain the integrity of a journaling file system such as ext3. Write barriers can be achieved with IDE by setting a barrier bit to force cache flushing. SCSI can achieve the same thing using ordered tags. These need to be implemented at the device-driver level.

I/O scheduling was another topic discussed. The current method is a single-direction elevator algorithm. A new elevator algorithm with a bounded delay using a 1-second deadline resulted in a 3% decrease in throughput. With a 100ms deadline there is an 8% decrease in throughput. The current I/O schedule does not implement any priority to increase deterministic behavior. This needs to be looked at in the future.

BoF sessions

The Birds of a Feather Sessions ran in three rooms and included VM, Crypto, ext2 Htree, memory management, swapped-to-RAM, and swapped-to-disk.

Scalability for Database Performance (IBM)

Ken Rozenthal

Ken gave a presentation on IBM's database requirements on Linux. The focus was on performance, scalability, and throughput; some of the requirements discussed included:

- Reduced capacity effects. The database is cached in user address space, and the goal is to have no page faults.
- Larger page size. One of the benefits of a larger page size is reduced TLB misses.
- Parallelizing I/O across many spindles of many devices.
- Large sequential-block I/O to avoid breaking up I/O requests.
- The ability to do raw I/O and avoid data buffering. The databases understand I/O and do not want to do it over a file system.
- Asynchronous I/O. There is a need for non–blocking completion detection.
- DMA support to high physical memory addresses to avoid copying.
- Reduced lock contention, with finer granularity device locks at the driver level.
- Numa effects. This can be accomplished with memory affinity, i.e., intelligent page allocation to the closest processor.

Again, it should be noted that many of the items are currently under construction (e.g., asynchronous I/O, by Ben LaHaise).

HP Kernel Wish List

Bdale Garbee

Bdale's presentation of HP's wish-list requirements for Linux focused on:

- SCSI subsystem support for a LUN count of >1024, better error handling, and support for failover and load balancing.
- Large system support, with >256 PCI busses.
- Hot-plug support. Make PCMCIA work like other devices.
- Device naming. For example, would like to have a interface such as `eth0` not move when a new device is plugged in.
- `Unionfs` support to transparently add other memory devices, such as docking stations to an Ipac.

During the discussion there was concern raised about multiple PCI bus handling. Bdale, on behalf of his Debian alter ego, discussed the problem of proprietary firmware being distributed with device drivers in the kernel distribution. The concern that this might violate the GPL prompted some discussion.

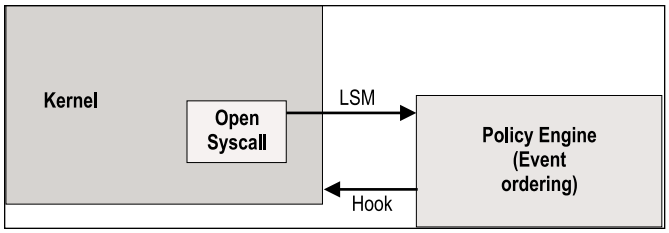

Figure 2

Loadable Security Module

Chris Wright

In the classic UNIX security model, the root user has access to everything. Other users are assigned privileges based on individual or group access. There are many custom security patches available. The Loadable Security Model (LSM) solution looks at a standardized security model for Linux. The LSM does not, however, support auditing for Common Criteria.

The LSM, shown in Figure 2, is an intrusive patch that consists of approximately 150 hooks distributed in the kernel and mediates 15 core kernel objects; each hook controls the function of an individual process. The total size of LSM is 3000. Policies are registered one at a time, but it is possible to register a composition policy. The reported overhead for the LSM security, using lmbench tests, was 0%–2%; the exception noted was GB Ethernet at 20%. LSM is available for both the 2.4 and 2.5 kernels.

Asynchronous I/O

Ben LaHaise

Asynchronous I/O potentially has some far-reaching effects in the kernel. Most of the new syscalls in the fs/aio.c have not yet been implemented, but those that have been tested have worked well.

One of the issues raised was support for two streams of I/O, one for synchronous and one for asynchronous. This would result in a lot of duplicate code. The suggestion was made to build the synchronous I/O on top of a complete implementation of asynchronous I/O. This would result in a lot of things breaking in the kernel. Linus was supportive and his philosophy in this area was to use the KISS Principle, doing only a minimal implementation, adding features as required, and fixing the problems as they come up.

SCSI

James Bottomley

This session provided an overview of the SCSI stack subsystem and some issues outstanding. Three of the areas that James focused on were the error-handling component, "cruft," and reducing or eliminating the SCSI mid-layer, moving most of it to the upper layer.

The discussion started with a backgrounder on the SCSI stack. He noted that the error-handling component of the SCSI stack is inadequate and in need of a major rework. James discussed some of details regarding this compo-

nent. He would like to replace the error-handler API with a more generic message-passing API.

The next area he focused on was "cruft." This is the result of a great deal of obsolete legacy code that needs to be reworked or eliminated. The SCSI subsystem is an important component, and emphasis was placed on a careful but deliberate reworking.

Some of the miscellaneous items noted were that the SCSI needs to be able to deal with hot-pluggable devices; that does not exist at this point. The subject of SAN devices was brought up; with the current system, the entire network can be probed for devices. It was recommended that a "lazy," or as-needed, allocation method be used. Overall this session was quite topical and provoked a good deal of constructive discussion.

Goals for 2.6 – Improving the Release-Engineering Process

Theodore Ts'o

Ted started out the closing session by asking for any suggestions on how to improve the next summit. These comments can be emailed. He noted that the size of the group was, at this point, quite manageable and productive and about as big as they would want to go.

The next discussion was on how to get 2.6 out. Currently, when a freeze is announced, there is a rush to get in all the last-minute patches. The result can often be that patches are not tested with other patches and things break. When things break, delays occur. There was some discussion about when the feature-freeze for the 2.6 release should occur, and Halloween (of this year!) was decided on.

There was some discussion of offloading from Linus the responsibility of maintaining stable releases, but no capable and willing volunteers found. (The paradox cited was that the person needed to be smart enough to do the job and stupid enough to take on the job.)

The session closed with a warm thanks to Ted for his work in organizing the summit.

BoF Sessions

The Birds of a Feather sessions discussed ACPI, SAN attachments, fbdev, and boot code issues and changes for faster boots.