

Comparison of Solaris, Linux, and FreeBSD Kernels

Similarities and Differences in some major kernel
subsystems.

Topics Covered

- Scheduling
- Memory Management/Paging
- File Systems
- Observability
- Conclusions

Topics Not Covered

- System Administration
- Performance
- Solaris/Linux/FreeBSD sucks/is the best wars

Some References

- Solaris Internals: Core Kernel Architecture by McDougall and Mauro (visit www.solarisinternals.com)
- The Design and Implementation of the FreeBSD Operating System by McKusick and Neville-Neil (visit www.mckusick.com)
- Linux Kernel Development by Love (see rlove.org/kernel_book)
- Understanding the Linux Kernel by Bovet and Cesati (www.oreilly.com/catalog/linuxkernel2)

More References

- Use the source...
- Use the tools that exist, or write your own
- There's lots of info on the web, but a lot of it is wrong or (very) outdated, or biased
- fxr.watson.org has browsable source for all 3 Oses, plus others

Scheduling and Schedulers

- Scheduling entities
 - Solaris
 - `kthread_t`, `proc_t`, `klwp_t`
 - FreeBSD
 - `Thread`, `proc`, `ksegrp` (kernel scheduling entity group)
 - Linux
 - `task_struct` – used for processes and threads
 - All 3 OSes schedule threads

Scheduling Priorities

- Scheduling decisions are based on priority
 - Solaris – higher priority is better
 - 0-59 – Time Shared, Interactive, Fixed, Fair Share Scheduler
 - 60-99 – System threads
 - 100-159 – Real Time threads
 - 160-169 – Low priority interrupts
 - High priority interrupts run in the interrupted thread

Scheduling Priorities

- Linux – Lower priority is better
 - 0-99 – System threads, Real Time, SCHED_FIFO, SCHED_RR
 - 100-139 – User threads, SCHED_NORMAL

Scheduling Priorities

- FreeBSD – lower priorities are better
 - 0-63 – Interrupts
 - 64-127 – Top half kernel
 - 128-159 – Real Time user threads
 - 160-223 – Time Shared user threads
 - 224-255 – Idle user threads

Scheduling Similarities

- Interactive (I/O bound threads) are favored
- Each CPU has its own “run queue”
- Solaris and Linux use a separate linked list for each priority
- FreeBSD uses a separate list for each group of 4 priority levels
- Warm affinity and load balancing mechanisms

Scheduling Differences

- FreeBSD and Linux use an “active” queue and an “expired” queue
 - Threads are scheduled in priority from the active queue
 - When a thread uses up its time slice, it is moved to the expired queue
 - When the active queue is empty, it is swapped with the expired queue
 - FreeBSD also has an “idle” queue

Scheduling Differences

- Solaris uses a “dispatch” queue for each cpu
 - When a thread uses up its time slice, it is given a new priority and slice, and returned to the dispatch queue
- Linux and FreeBSD do a priority calculation based on CPU usage vs. sleep time
- Solaris does a table lookup for priority (except for the Fair Share Scheduler)
- Solaris and FreeBSD support processor fencing

Scheduling Classes

- Solaris supports multiple scheduling classes running on the system at the same time
 - Time Shared, Interactive, Fixed, Fair Share Scheduler, System, and Real Time
 - New scheduling classes can be added
- All three support POSIX scheduling classes
- Linux and FreeBSD choose scheduler at compile time
- Solaris and Linux support kernel preemption

Memory Management

- Solaris - Address space is in “segments”
 - Visible using `pmap(1)`
 - HAT layer handles hardware
- FreeBSD - Address space in “regions”
 - Pmap handles physical mapping, vmap handles virtual
- Linux – Address space is in “memory areas”
 - Visible with `pmap` and `/proc/.../maps`

Memory Management

- Segments, Regions, and Memory Areas
 - Virtual address of start of area
 - Location in object to which the area is mapped
 - Permissions
 - Size of the mapping

Paging

- All 3 OSes use a variation of a least recently used algorithm for page replacement
- Global working set
- All 3 OSes have a system daemon to do page aging and stealing
 - Solaris
 - Pageout runs periodically and when free memory is low
 - Paging parameters are calibrated at system start based on processor speed, amount of memory, and paging speed

Paging

- FreeBSD
 - vm_pageout daemon runs periodically and calls vm_pageout_scan when memory is low
 - Paging parameters are hard-coded or tunable
- Linux
 - Multiple kswapd daemons
 - Dynamically tuned while system is running

Paging

- Solaris page lists
 - Free list
 - Hashed list
 - Vnode list
 - (Almost) all pages are mapped by an array which is scanned using a “2-handed clock algorithm”

Paging

- FreeBSD page lists
 - Active
 - Inactive
 - Cached
 - Free
 - Locked (“wired”) pages not on any list

Paging

- Linux page lists
 - Multiple sets of 3 “zones”
 - Normal pages
 - DMA pages
 - Dynamically allocated pages
 - Pages move between “hot”, “cold”, and free lists

Paging

- All 3 OSes take NUMA locality into account
- I/O buffer cache is merged into one system page cache
 - I/O pages, memory mapped files, text and data of applications are in the page cache

File Systems

- All 3 operating systems implement a “virtual file system” layer to hide file system dependent operations from the users of a file
- Linux has inode operations and file operations
- Solaris and FreeBSD combine these into vnode operations
- Additional file systems may be added or ported

Observability

- Solaris has the most tools for observing system behaviour (due, in part to lack of source code in the past)
 - mdb/kmdb
 - Dtrace
 - Stat tools
- FreeBSD allows use of gdb on kernel
- Linux has user mode linux, and a kernel debugger (but these must be installed)

Conclusions

- Mechanisms are similar within all 3 OSes for implementing major kernel subsystems
- Linux tends to use data abstraction at a higher level
 - More machine dependent code and data structures
 - May be faster, but more work to support
- FreeBSD may have most documentation available in one place on kernel implementation

Conclusions

- Lots of documentation on Linux, but lots of it is dated as the subsystems change (sometimes re-write) with different releases
- Linux has a large community to answer questions
- Solaris is catching up as kernel developers blog their work, and because of source availability