The FreeBSD SMP implementation

Greg ‘‘groggy’’ Lehey
The FreeBSD Project
grog@lemis.com
Singapore, 30 October 2003

Topics

- How we got into this mess.
- Why the UNIX kernel is not suited to multiple processors.
- Solving the problem.
- Team dynamics.
- Current state of play.

Free OS performance

- Common knowledge: UNIX is faster than Microsoft.
- Example: ftp server recommendations, 1999.
- Microsoft: Three systems, each four processors, 512 MB memory.
- Only one system ran, the other were backups.
- Throughput: 6 GB/day.
- FreeBSD: *wcarchive.cdrom.com* had only one CPU.
- No failover needed.
- Throughput: 700 GB/day.

The Mindcraft benchmarks

- In 1999, Mindcraft published benchmarks showing NT much faster than Linux.
- Linux people first claimed the results were wrong.
- Linux people later realised the results were correct, but the benchmark was contrived.
- FreeBSD people kept very quiet.
- One of the problems was the ‘‘big kernel lock’’ SMP implementation.

The UNIX kernel design

- One CPU
- Processes perform user functions.
- Interrupt handlers handle I/O.
- Interrupt handlers have priority over processes.

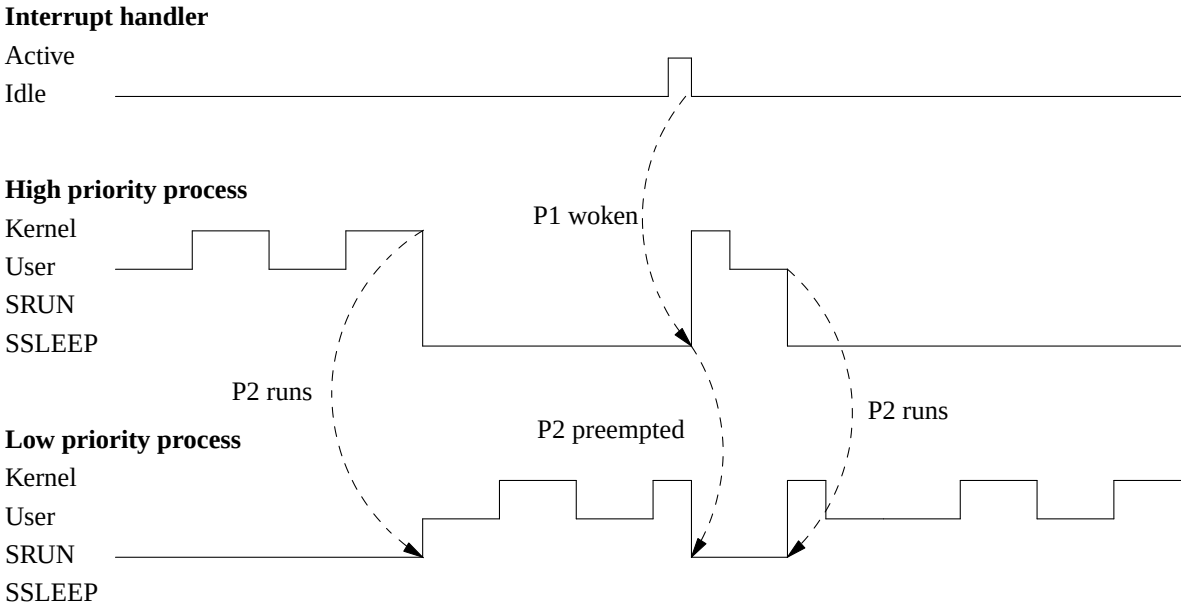
Processes

- One CPU
- Processes have different priorities.
- The scheduler chooses the highest priority process which is ready to run.
- The process can relinquish the CPU voluntarily (`tsleep`).
- The scheduler runs when the process finishes its time slice.
- Processes are not scheduled while running kernel code.

Interrupts

- Interrupts cannot be delayed until kernel is inactive.
- Different synchronization: block interrupts in critical kernel code.
- Finer grained locking: `splbio` for block I/O, `spltty` for serial I/O, `splnet` for network devices, etc.

Ideal single processor scheduling



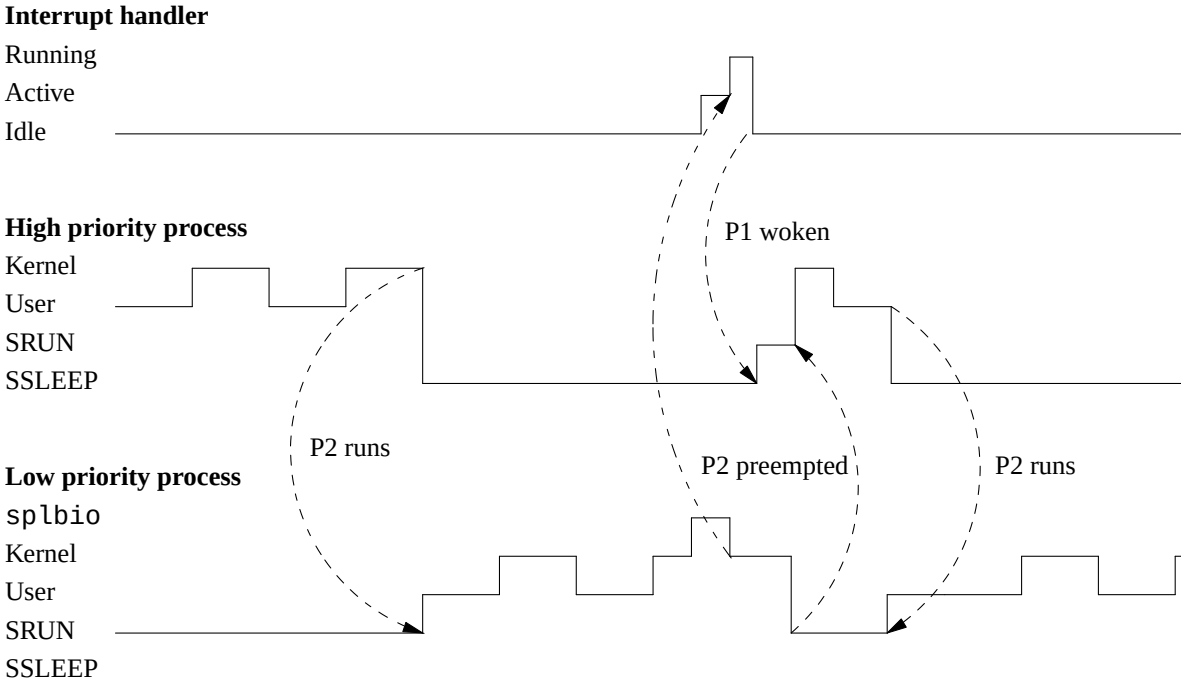
Problems with this approach

Kernel synchronization is inadequate. UNIX can't guarantee consistency if multiple processes can run in kernel mode at the same time.

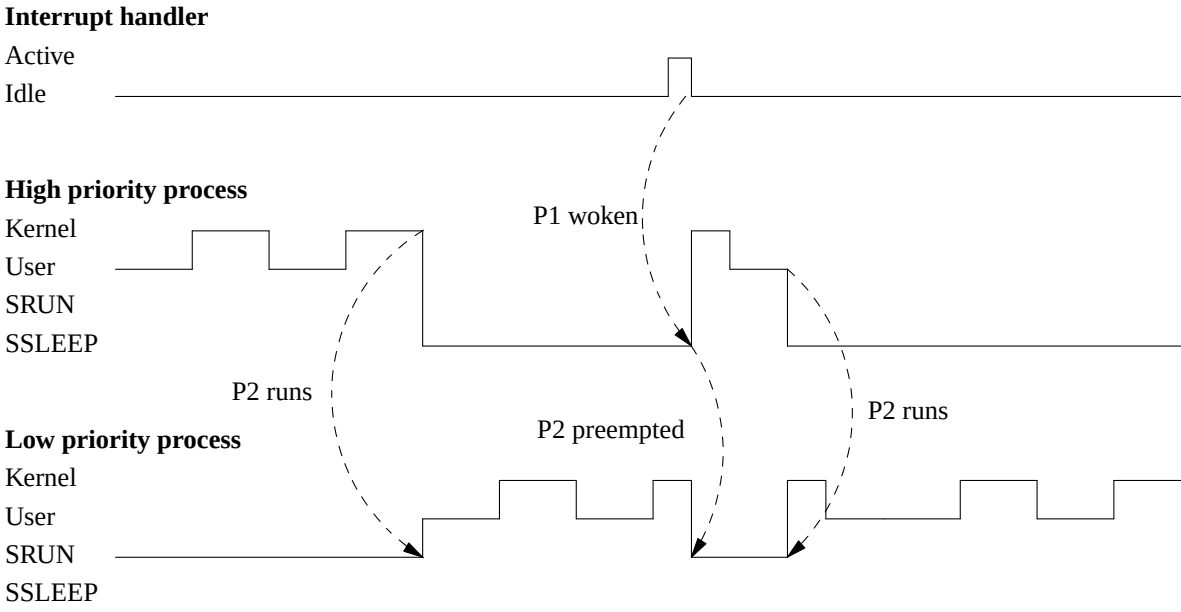
Solution: Ensure that a process leaves kernel mode before preempting it. Since processes do not execute kernel code for very long, this causes only minimal problems.

Danger: If a process does stay in the kernel for an extended period of time, it can cause significant performance degradation or even hangs.

Real single processor scheduling



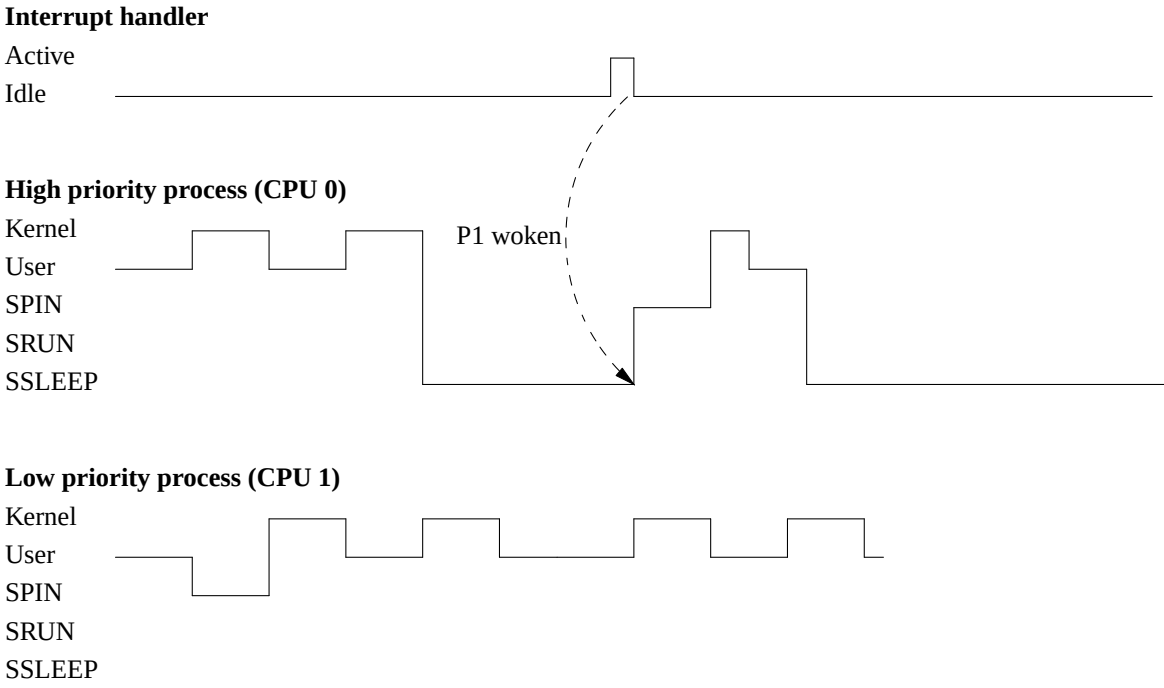
Ideal single processor scheduling



Problems on SMP machines

- UNIX approach doesn't allow more than one process running in kernel mode.
- ''Solution'': introduce Big Kernel Lock. Spin (loop) waiting for this lock if it's taken.
- Disadvantage: much CPU time may be lost.

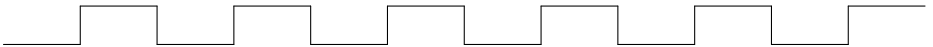
Real dual processor scheduling



Quad processor scheduling: ideal

Process in CPU 0

Kernel
User
SPIN



Process in CPU 1

Kernel
User
SPIN

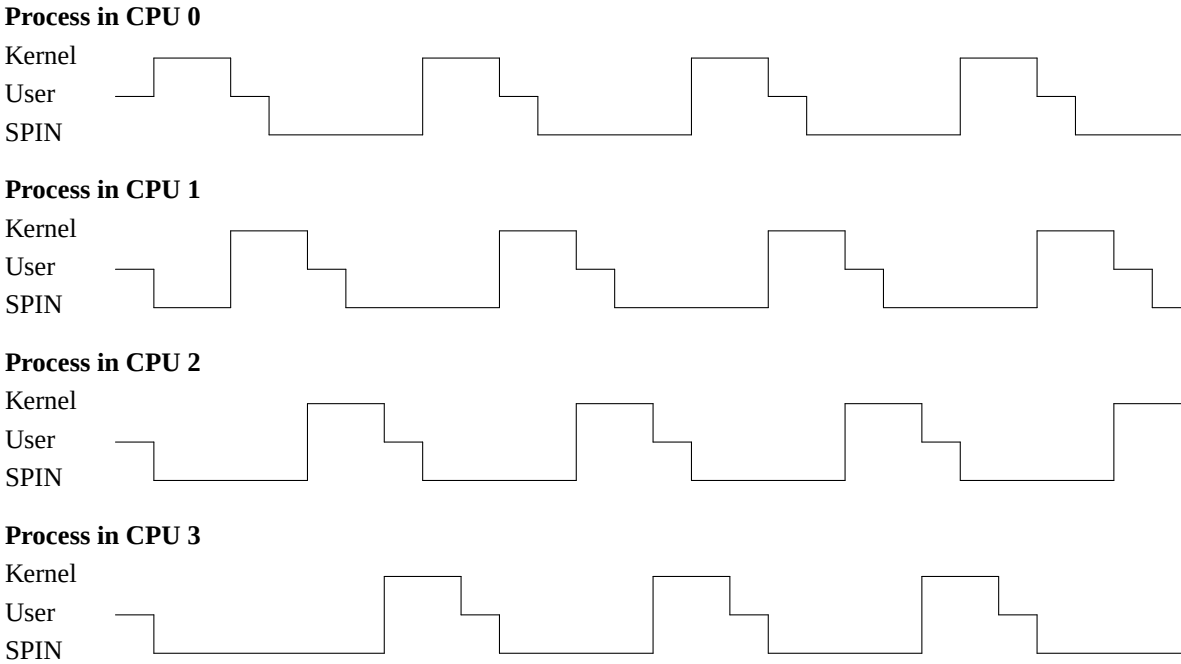
Process in CPU 2

Kernel
User
SPIN

Process in CPU 3

Kernel
User
SPIN

Quad processor scheduling: real



Getting out of the mess

- Linux people started working very quickly.
- FreeBSD people stunned, unable to move.
- FreeBSD people looked at the Linux solution and didn’t under^H^H^H^H^Hlike it.
- BSDi bought out Walnut Creek CDROM.

The BSD/OS merge

- BSDi (previously BSDI) was the vendor of BSD/OS.
- Initial plans were to merge BSD/OS and FreeBSD.
- Licensing and technical issues stopped this merge.
- BSD/OS already had better SMP support under development, code name ‘‘SMPng’’.
- FreeBSD developers were given access to BSD/OS source code.
- FreeBSD developers allowed to merge ‘‘significant parts’’ of BSD/OS code.
- Decision made to merge SMPng.

The meeting at Yahoo!

- Interested developers met at Yahoo! in June 2000.
- Total of 20 participants.
- 11 FreeBSD developers.
- 3 Apple developers.
- 3 Yahoo! staff.
- 2 BSDi developers.

Limiting the delays

- Create ‘‘fine-grained’’ locking: lock only small parts of the kernel.
- If resource is not available, block, don’t spin.
- Problem: interrupt handlers can’t block.
- Solution: let them block, then.

Blocking interrupt handlers

- Interrupt handlers get a process context.
- Short term: normal processes, involve scheduler overhead on every invocation.
- Longer term: ‘‘light weight interrupt threads’’, scheduled only when conflicts occur.
- Choice dictated by stability requirements during changeover.
- Resurrect the idle process, which gives a process context to each interrupt process.

Blocking interrupt handlers

```
USER     PID %CPU %MEM    VSZ  RSS  TT  STAT STARTED      TIME COMMAND
root      11 99.0  0.0      0   12  ??  RL    2:11PM 150:09.29  (idle: cpu1)
root      12 99.0  0.0      0   12  ??  RL    2:11PM 150:08.71  (idle: cpu0)
root       1  0.0  0.3    756  384  ??  ILs   2:11PM   0:00.11 /sbin/init --
root      13  0.0  0.0      0   12  ??  WL    2:11PM   0:30.61  (swi8: tty:sio clock)
root      15  0.0  0.0      0   12  ??  WL    2:11PM   0:01.32  (swi1: net)
root       2  0.0  0.0      0   12  ??  DL    2:11PM   0:02.35  (g_event)
root       3  0.0  0.0      0   12  ??  DL    2:11PM   0:01.53  (g_up)
root       4  0.0  0.0      0   12  ??  DL    2:11PM   0:01.63  (g_down)
root      16  0.0  0.0      0   12  ??  DL    2:11PM   0:01.05  (random)
root      17  0.0  0.0      0   12  ??  WL    2:11PM   0:00.16  (swi7: task queue)
root      18  0.0  0.0      0   12  ??  WL    2:11PM   0:00.00  (swi6:+)
root       5  0.0  0.0      0   12  ??  DL    2:11PM   0:00.00  (taskqueue)
root      21  0.0  0.0      0   12  ??  WL    2:11PM   0:00.01  (swi3: cambio)
root      22  0.0  0.0      0   12  ??  WL    2:11PM   0:00.08  (irq14: ata0)
root      24  0.0  0.0      0   12  ??  WL    2:11PM   0:02.38  (irq2: rl0 uhci0)
root      25  0.0  0.0      0   12  ??  DL    2:11PM   0:00.00  (usb0)
root      26  0.0  0.0      0   12  ??  DL    2:11PM   0:00.00  (usbtask)
root      27  0.0  0.0      0   12  ??  WL    2:11PM   0:00.00  (irq5: fwohci0++)
root      28  0.0  0.0      0   12  ??  WL    2:11PM   0:00.00  (irq10: sym0)
root      29  0.0  0.0      0   12  ??  WL    2:11PM   0:00.01  (irq11: sym1)
root      30  0.0  0.0      0   12  ??  WL    2:11PM   0:00.00  (irq1: atkbd0)
root      31  0.0  0.0      0   12  ??  WL    2:11PM   0:00.00  (irq6: fdc0)
root       6  0.0  0.0      0   12  ??  DL    2:11PM   0:00.04  (pagedaemon)
root       7  0.0  0.0      0   12  ??  DL    2:11PM   0:00.00  (vmdaemon)
root      39  0.0  0.0      0   12  ??  DL    2:11PM   0:01.00  (syncer)
```

Types of locking constructs

- Semaphores.
- Spin locks.
- Adaptive locks.
- Blocking locks.
- Condition variables.
- Read-write locks.
- Locking constructs are also called *mutexes*.

Semaphores

- Oldest synchronization primitive.
- Include a *count* variable which defines how many processes may access the resource in parallel.
- No concept of ownership.
- The process that releases a semaphore may not be the process which last acquired it.
- Waiting is done by blocking (scheduling).
- Traditionally used for synchronization between processes.

Spin locks

- Controls a single resource: only one process may own it.
- ‘‘Busy wait’’ when lock is not available.
- May be of use where the delay is short (less than the overhead to run the scheduler).
- Can be very wasteful for longer delays.
- The only primitive that can be used if there is no process context (traditional interrupt handlers).
- May have an *owner*, which is useful for consistency checking and debugging.

Blocking lock

- Controls a single resource: only one process may own it.
- Runs the scheduler when lock is not available.
- Generally usable where process context is available.
- May be less efficient than spin locks where the delay is short (less than the overhead to run the scheduler).
- Can only be used if there is a process context.
- May have an *owner*, which is useful for consistency checking and debugging.

Adaptive lock

- Combination of spin lock and blocking lock.
- When lock is not available, spin for a period of time, then block if still not available.
- Can only be used if there is a process context.
- May have an *owner*, which is useful for consistency checking and debugging.

Condition variable

- Tests an external condition, blocks if it is not met.
- When the condition is met, all processes sleeping on the wait queue are woken.
- Similar to *tsleep*/*wakeup* synchronization.

Read-write lock

- Special case of a blocking lock.
- Allows multiple readers or alternatively one writer.

Comparing locks

Lock type	Multiple resources	owner	requires context
Semaphore	yes	no	yes
Spin lock	no	yes	no
Blocking lock	no	yes	yes
Adaptive lock	no	yes	yes
Condition variable	yes	no	yes
Read-write lock	yes	no	yes

Recursion

- What do we do if a process tries to take a mutex it already has?
- Could be indicative of poor code structure.
- In the short term, it's very likely.
- Solaris does not allow recursion, and this has caused many problems.
- Currently FreeBSD allows recursion.
- Discussion continues.

Condition variables

- Acquire a condition variable with `cv_wait()`, `cv_wait_sig()`, `cv_timedwait()` or `cv_timed-wait_sig()`.
- Before acquiring the condition variable, the associated mutex must be held. The mutex will be released before sleeping and reacquired on wakeup.
- Unblock one waiter with `cv_signal()`.
- Unblock all waiters with `cv_broadcast()`.
- Wait for queue empty with `cv_waitq_empty`.
- Same functionality available from the `msleep` function.

msleep

- A version of `tsleep` which takes a mutex parameter.
- The mutex will be released before sleeping and reacquired on wakeup.
- Similar to the behaviour of `tsleep` with `splx` functions in traditional UNIX.
- `tsleep` reimplemented as a macro calling `msleep` with null mutex.
- Functionality equivalent to condition variables, which should be used for new code.

Shared/exclusive locks

- Really read/write locks.
- More expensive than mutexes, should only be used where very few write (exclusive) accesses occur.
- Create an sx lock with `sx_init()`.
- Attain a read (shared) lock with `sx_slock()` and release it with `sx_sunlock()`.
- Attain a write (exclusive) lock with `sx_xlock()` and release it with `sx_xunlock()`.
- Destroy an sx lock with `sx_destroy`.

Project planning: after Yahoo!

- Aim for stability, not performance, during the development.
- Port BSDi code, don’t reinvent the wheel.
- Use heavy-weight threads at first to enable better debugging.

Responsibilities

- Matt Dillon to port locking primitives and *schedlock*.
- Greg Lehey to port interrupt threads.
- Doug Rabson to handle Alpha issues.
- Paul Saab to enhance *ddb*.
- Jonathan Lemon to convert network drivers.
- Chuck Paterson of BSDi to be project liaison.
- Jason Evans to be project manager.
- (But we never had a project manager before).

First steps

- Matt Dillon had a tight deadline.
- Porting the BSD/OS code was more difficult than anticipated.
- Matt rewrote the code.
- Code later ported by Jake Burkholder.

First steps (2)

- Greg Lehey started porting interrupt threads.
- Greg had two weeks to port the code.
- Porting the BSD/OS code was more difficult than anticipated.
- Heavyweight threads only in SPARC code.
- SPARC and Intel code very different.

First steps (3)

- Differences between FreeBSD and BSD/OS larger than anticipated.
- Four-way comparison: FreeBSD 4.0, BSD/OS 4.0, BSD/OS 5 Intel, BSD/OS 5 SPARC.
- Took two months.

Initial commit

```
From: Jason Evans <jasone@FreeBSD.org>
To: cvs-committers@FreeBSD.org, cvs-all@FreeBSD.org
Subject: cvs commit: src/bin/ps print.c src/share/man/man9 mutex.9 Makefile
         src/usr.bin/top machine.c src/sys/alpha/alpha mp_machdep.c
         synch_machdep.c clock.c genassym.c interrupt.c ipl_funcs.c
         locore.s machdep.c mem.c pmap.c prom.c support.s swtch.s trap.c ...

jasone      2000/09/06 18:33:03 PDT

  (file names omitted)
  Log:
  Major update to the way synchronization is done in the kernel.  Highlights
  include:

  * Mutual exclusion is used instead of spl*().  See mutex(9).  (Note: The
    alpha port is still in transition and currently uses both.)

  * Per-CPU idle processes.

  * Interrupts are run in their own separate kernel threads and can be
    preempted (i386 only).

  Partially contributed by:     BSDi (BSD/OS)
  Submissions by (at least):    cp, dfr, dillon, grog, jake, jhb, sheldonh
```

Original locks

- `Giant`: protects the kernel.
- `sched_lock`: protects the scheduler.

Current situation

- `Giant` still protects most of the kernel, but is being weakened.
- `softclock` and signal handling are now MP-safe and do not require `Giant`.
- Individual components protected by leaf node mutexes.
- Many device drivers now converted.
- Choice of construct often left to individual developer.
- Few mid-range locking constructs.

Opening the flood gates

- After this point, many people got involved.
- Much cosmetic work.
- Terminology problems: what is a ‘‘mutex’’?
- No clear direction: Many contributors decided on their own locking constructs.
- Loss of Jason Evans as project leader in March 2001.
- No replacement project leader.

The kernel summit

- ‘‘Kernel summit’’ held at Boston USENIX conference on Saturday, 30 June 2001.
- Many interruptions.
- Conclusion: release FreeBSD 5.0 in November 2001, ready or not.
- FreeBSD 5.0 was finally released in December 2002.

Project communications

- Mailing list `smp@FreeBSD.org`, very little traffic.
- Much traffic on IRC channel.
- IRC tended to limit the number of participants.
- No record of discussions.

Debugging tools

- BSD/OS supplied debugging tools.
- Not all BSD/OS tools have been ported.
- Few new tools have been written: debugging is not interesting enough.
- Good task for ‘‘Junior Kernel Hacker’’.

Documentation

- Change logs very well documented.
- Individual functions well documented.
- Overall project plan less clear.

Sample commit log

```
jhb         2001/09/10 14:04:49 PDT

  Modified files:
    sys/kern             kern_shutdown.c
  Log:
  - Axe holding_giant as it is not used now anyways and was ok'd by
    dillon in an earlier e-mail.
  - We don't need to test the console right before we vfprintf() the
    panicstr message.  The printing of the panic message is a fine
    console test by itself and doesn't make useful messages scroll
    off the screen or tick developers off in quite the same.

  Requested by:      jlemon, imp, bmilekic, chris, gsutter, jake (2)

  Revision  Changes    Path
  1.110     +5 -36     src/sys/kern/kern_shutdown.c
```

Other commit logs

```
From: John Baldwin <jhb@FreeBSD.org>
Date: Tue, 21 Nov 2000 13:10:15 -0800 (PST)

jhb         2000/11/21 13:10:15 PST

  Modified files:
    sys/kern             kern_ktr.c
  Log:
  Ahem, fix the disclaimer portion of the copyright so it disclaim's the
  voices in my head.  You can sue the voices in Bill Paul's head all you
  want.

  Noticed by:   jhb

  Revision  Changes    Path
  1.6       +3 -3      src/sys/kern/kern_ktr.c
```

Other commit logs

```
--- kern_ktr.c  2000/11/15 21:51:53     1.5
+++ kern_ktr.c  2000/11/21 21:10:15     1.6
@@ -14,10 +14,10 @@
   *    may be used to endorse or promote products derived from this software
   *    without specific prior written permission.
   *
- * THIS SOFTWARE IS PROVIDED BY Bill Paul AND CONTRIBUTORS ''AS IS'' AND
+ * THIS SOFTWARE IS PROVIDED BY JOHN BALDWIN AND CONTRIBUTORS ''AS IS'' AND
   * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
   * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
- * ARE DISCLAIMED.  IN NO EVENT SHALL Bill Paul OR THE VOICES IN HIS HEAD
+ * ARE DISCLAIMED.  IN NO EVENT SHALL JOHN BALDWIN OR THE VOICES IN HIS HEAD
   * BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
   * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
   * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
@@ -26,7 +26,7 @@
   * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF
   * THE POSSIBILITY OF SUCH DAMAGE.
   *
- * $FreeBSD: src/sys/kern/kern_ktr.c,v 1.5 2000/11/15 21:51:53 jhb Exp $
+ * $FreeBSD: src/sys/kern/kern_ktr.c,v 1.6 2000/11/21 21:10:15 jhb Exp $
   */

  /*
```

Other commit logs

```
From: John Baldwin <jhb@FreeBSD.org>

On 21-Nov-00 John Baldwin wrote:
> jhb         2000/11/21 13:10:15 PST
>
>   Modified files:
>     sys/kern             kern_ktr.c
>   Log:
>   Ahem, fix the disclaimer portion of the copyright so it disclaim's the
>   voices in my head.  You can sue the voices in Bill Paul's head all you
>   want.
>
>   Noticed by: jhb

Oh geez.  That should be 'Noticed by: jlemon'.  I guess the voices
are getting a bit too rambunctious.
```

Other commit logs

```
From: Warner Losh <imp@village.org>

In message <XFMail.001121131818.jhb@FreeBSD.org> John Baldwin writes:
: Oh geez.  That should be 'Noticed by: jlemon'.  I guess the voices
: are getting a bit too rambunctious.

It could be worse.  You could be talking about yourself in the third
person.  Warner hates it when he does that.
```

Other commit logs

```
From: John Baldwin <jhb@FreeBSD.ORG>

On 21-Nov-00 Warner Losh wrote:
> In message <XFMail.001121131818.jhb@FreeBSD.org> John Baldwin writes:
>: Oh geez.  That should be 'Noticed by: jlemon'.  I guess the voices are
>: getting
>: a bit too rambunctious.
>
> It could be worse.  You could be talking about yourself in the third
> person.  Warner hates it when he does that.

Well, I'm sure Warner will have a private discussion with Warner about
doing that in public.

I wonder how the voices do their locking...
```

Other commit logs

```
From: Warner Losh <imp@village.org>

In message <XFMail.001121133952.jhb@FreeBSD.org> John Baldwin writes:
: On 21-Nov-00 Warner Losh wrote:
: > In message <XFMail.001121131818.jhb@FreeBSD.org> John Baldwin writes:
: >: Oh geez.  That should be 'Noticed by: jlemon'.  I guess the voices are
: >: getting
: >: a bit too rambunctious.
: >
: > It could be worse.  You could be talking about yourself in the third
: > person.  Warner hates it when he does that.
:
: Well, I'm sure Warner will have a private discussion with Warner about
: doing that in public.

Warner will do that only if Warner notices.

: I wonder how the voices do their locking...

Warner Speculates that Warner's voices don't do
locking.
```

Other commit logs

```
From: John Baldwin <jhb@FreeBSD.ORG>

On 21-Nov-00 Warner Losh wrote:
> In message <XFMail.001121133952.jhb@FreeBSD.org> John Baldwin writes:
>: Well, I'm sure Warner will have a private discussion with Warner about
>: doing that in public.
>
> Warner will do that only if Warner notices.
>
>: I wonder how the voices do their locking...
>
> Warner Speculates that Warner's voices don't do locking.
John thinJohn's voices are too inks that the consefficient to useole driver
doesn't ha sleep locks and endve any lock up sping yeinning a lott.
fatal double fault
eip = 0x000000
ebp = %62F k epomn e
```

(untangled)

John thin*John's voices are too in*ks that the cons*efficient to use*ole driver doesn't ha *sleep locks and end*ve any lock *up sp*ing ye*inning a lot*t.

1. John thinks that the console driver doesn't have any locking yet.
2. John's voices are too inefficient to use sleep locks and end up spinning a lot.

Performance

- Initial performance drop was expected due to scheduling interrupts.
- Performance was better than initially expected.
- Final implementation should be much better (main project aim).
- Most of kernel still locked by `Giant`.
- Current status: comparable to release 4, but some areas need work.
- Light-weight threads still not implemented.

Performance (continued)

- What effect does this have on single processor systems?
- Currently we don't know how good it will be.
- How do we address the ‘‘how many processors?’’ question?
- How much good will light-weight interrupt threads do?

Current status

- SMP conversion effectively complete.
- BSDi code was never released.
- Numerous subsystems not converted.
- Conversion may happen during reimplementation (e.g. GEOM, KSE).
- Light-weight threads to be implemented as part of KSE project.
- KSE project not yet complete.
- Many developers no longer run `-CURRENT`.

Further information

`http://www.FreeBSD.org/`

Join in! The FreeBSD project needs more clever hackers.

These slides are available at
`http://www.lemis.com/grog/SMPng/Singapore/`