FreeBSD SMPng: Behind the scenes

Greg Lehey
grog@FreeBSD.org
grog@au1.ibm.com
Sydney, 27 September 2001

Topics

- How we got into this mess.

Topics

- How we got into this mess.
- Why the UNIX kernel is not suited to multiple processors.

Topics

- How we got into this mess.
- Why the UNIX kernel is not suited to multiple processors.
- Solving the problem.

Topics

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

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.

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.
- Looking forward.

The Mindcraft benchmarks

- Common knowledge: UNIX is faster than Microsoft.

The Mindcraft benchmarks

- Common knowledge: UNIX is faster than Microsoft.
- In 1999, Mindcraft published benchmarks showing NT much faster than Linux.

The Mindcraft benchmarks

- Common knowledge: UNIX is faster than Microsoft.
- In 1999, Mindcraft published benchmarks showing NT much faster than Linux.
- Linux people first claimed the results were wrong.

The Mindcraft benchmarks

- Common knowledge: UNIX is faster than Microsoft.
- 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.

The Mindcraft benchmarks

- Common knowledge: UNIX is faster than Microsoft.
- 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.

The Mindcraft benchmarks

- Common knowledge: UNIX is faster than Microsoft.
- 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

The UNIX kernel design

- One CPU
- Processes perform user functions.

The UNIX kernel design

- One CPU
- Processes perform user functions.
- Interrupt handlers handle I/O.

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

- One CPU
- Processes have different priorities.

Processes

- One CPU
- Processes have different priorities.
- The scheduler chooses the highest priority process which is ready to run.

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`).

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

- 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.

Interrupts

- Interrupts cannot be delayed until kernel is inactive.
- Different synchronization: block interrupts in critical 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.

Problems on SMP machines

- UNIX approach doesn’t allow more than one process running in kernel mode.

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.

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.

Getting out of the mess

- Linux people started working very quickly.

Getting out of the mess

- Linux people started working very quickly.
- FreeBSD people stunned, unable to move.

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 understand

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.

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.

The BSD/OS merge

- BSDi (previously BSDI) was the vendor of BSD/OS.
- Initial plans were to merge BSD/OS and FreeBSD.

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.

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’’.

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.

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.

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.

Limiting the delays

- Create ‘‘fine-grained’’ locking: lock only small parts of the kernel.
- If resource is not available, block, don’t spin.

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.

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.

Blocking interrupt handlers

- Interrupt handlers get a process context.
- Short term: normal processes, involve scheduler overhead on every invocation.

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.

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.

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      10 99.0  0.0      0    0  ??  RL   Thu10AM 4332:57.28  (idle: cpu1)
root      11 99.0  0.0      0    0  ??  RL   Thu10AM 4331:26.95  (idle: cpu0)
root      12  0.0  0.0      0    0  ??  WL   Thu10AM   6:02.04  (swi1: net)
root      13  0.0  0.0      0    0  ??  WL   Thu10AM  11:43.06  (swi6: tty:sio clock)
root      14  0.0  0.0      0    0  ??  WL   Thu10AM   0:00.00  (swi4: vm)
root      15  0.0  0.0      0    0  ??  WL   Thu10AM   0:00.00  (swi2: camnet)
root      16  0.0  0.0      0    0  ??  WL   Thu10AM   0:00.00  (swi3: cambio)
root      17  0.0  0.0      0    0  ??  WL   Thu10AM   0:00.00  (swi5: task queue)
root      18  0.0  0.0      0    0  ??  WL   Thu10AM   0:00.16  (irq14: ata0)
root      19  0.0  0.0      0    0  ??  WL   Thu10AM   0:00.00  (irq15: ata1)
root      20  0.0  0.0      0    0  ??  WL   Thu10AM  10:38.31  (irq9: dc0)
root      21  0.0  0.0      0    0  ??  WL   Thu10AM   1:51.00  (irq11: atapci1+)
root      22  0.0  0.0      0    0  ??  WL   Thu10AM   0:00.00  (irq1: atkbd0)
root      23  0.0  0.0      0    0  ??  WL   Thu10AM   0:00.00  (swi0: tty:sio)
root      24  0.0  0.0      0    0  ??  WL   Thu10AM   0:00.00  (irq4: sio0)
root      25  0.0  0.0      0    0  ??  WL   Thu10AM   0:00.00  (irq7: ppc0)
root      26  0.0  0.0      0    0  ??  WL   Thu10AM   0:00.00  (irq0: clk)
root      27  0.0  0.0      0    0  ??  WL   Thu10AM   0:00.00  (irq8: rtc)
root       2  0.0  0.0      0    0  ??  DL   Thu10AM   0:02.33  (pagedaemon)
root       3  0.0  0.0      0    0  ??  DL   Thu10AM   0:00.00  (vmdaemon)
root       4  0.0  0.0      0    0  ??  DL   Thu10AM   0:00.02  (pagezero)
root       5  0.0  0.0      0    0  ??  DL   Thu10AM   0:08.08  (bufdaemon)
root       6  0.0  0.0      0    0  ??  DL   Thu10AM   2:17.24  (syncer)
root       0  0.0  0.0      0    0  ??  DLs  Thu10AM   0:01.66  (swapper)
root       1  0.0  0.1    656   29  ??  ILs  Thu10AM   0:03.93 /sbin/init -d
```

Conclusions at Yahoo!

- Aim for stability, not performance, during the development.

Conclusions at Yahoo!

- Aim for stability, not performance, during the development.
- Port BSDi code, don’t reinvent the wheel.

Conclusions at 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*.

Responsibilities

- Matt Dillon to port locking primitives and *schedlock*.
- Greg Lehey to port interrupt threads.

Responsibilities

- Matt Dillon to port locking primitives and *schedlock*.
- Greg Lehey to port interrupt threads.
- Doug Rabson to handle Alpha issues.

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*.

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.

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.

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.

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.

First steps

- Matt Dillon had a tight deadline.
- Porting the BSD/OS code was more difficult than anticipated.

First steps

- Matt Dillon had a tight deadline.
- Porting the BSD/OS code was more difficult than anticipated.
- Matt rewrote the code.

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

- Greg Lehey started porting interrupt threads.

First steps

- Greg Lehey started porting interrupt threads.
- Greg had two weeks to port the code.

First steps

- Greg Lehey started porting interrupt threads.
- Greg had two weeks to port the code.
- Porting the BSD/OS code was more difficult than anticipated.

First steps

- 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.

First steps

- 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

- Differences between FreeBSD and BSD/OS larger than anticipated.

First steps

- 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.

First steps

- 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
```

Opening the flood gates

- After this point, many people got involved.

Opening the flood gates

- After this point, many people got involved.
- Much cosmetic work.

Opening the flood gates

- After this point, many people got involved.
- Much cosmetic work.
- Terminology problems: what is a ‘‘mutex’’?

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.

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.

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.

The kernel summit

- ‘‘Kernel summit’’ held at Boston USENIX conference on Saturday, 30 June 2001.
- Many interruptions.

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.

Project communications

- Mailing list `smp@FreeBSD.org`, very little traffic.

Project communications

- Mailing list `smp@FreeBSD.org`, very little traffic.
- Much traffic on IRC channel.

Project communications

- Mailing list `smp@FreeBSD.org`, very little traffic.
- Much traffic on IRC channel.
- IRC tended to limit the number of participants.

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.

Debugging tools

- BSD/OS supplied debugging tools.
- Not all BSD/OS tools have been ported.

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.

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.

Documentation

- Change logs very well documented.
- Individual functions well documented.

Documentation

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

Sample man page

```
MUTEX(9)                FreeBSD Kernel Developer's Manual                MUTEX(9)

NAME
     mutex, mtx_init, mtx_lock, mtx_lock_spin, mtx_lock_flags,
     mtx_lock_spin_flags, mtx_trylock, mtx_trylock_flags, mtx_unlock,
     mtx_unlock_spin, mtx_unlock_flags, mtx_unlock_spin_flags, mtx_destroy,
     mtx_initialized, mtx_owned, mtx_recursed, mtx_assert - kernel synchro-
     nization primitives

SYNOPSIS
     #include <sys/param.h>
     #include <sys/lock.h>
     #include <sys/mutex.h>

     void
     mtx_init(struct mtx *mutex, const char *description, int opts);

     void
     mtx_lock(struct mtx *mutex);

     void
     mtx_lock_spin(struct mtx *mutex);

     void
     mtx_lock_flags(struct mtx *mutex, int flags);
```

Sample man page (continued)

```
DESCRIPTION
     Mutexes are the most basic and primary method of process synchronization.
     The major design considerations for mutexes are:

     1.   Acquiring and releasing uncontested mutexes should be as cheap as
          possible.

     2.   They must have the information and storage space to support priority
          propagation.

     3.   A process must be able to recursively acquire a mutex, provided that
          the mutex is initialized to support recursion.

     There are currently two flavors of mutexes, those that context switch
     when they block and those that do not.

     By default, MTX_DEF mutexes will context switch when they are already
     held.  As a machine dependent optimization they may spin for some amount
     of time before context switching.  It is important to remember that since
     a process may be preempted at any time, the possible context switch
     introduced by acquiring a mutex is guaranteed to not break anything that
     isn't already broken.
```

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 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.

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

Performance

- Initial performance drop was expected due to scheduling interrupts.

Performance

- Initial performance drop was expected due to scheduling interrupts.
- Final implementation should be much better (main project aim).

Performance

- Initial performance drop was expected due to scheduling interrupts.
- Final implementation should be much better (main project aim).
- What effect does this have on single processor systems?

Performance

- Initial performance drop was expected due to scheduling interrupts.
- Final implementation should be much better (main project aim).
- What effect does this have on single processor systems?
- Currently we don’t know how good it will be.

Performance

- Initial performance drop was expected due to scheduling interrupts.
- Final implementation should be much better (main project aim).
- 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?

Performance

- Initial performance drop was expected due to scheduling interrupts.
- Final implementation should be much better (main project aim).
- 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?

Performance

- Initial performance drop was expected due to scheduling interrupts.
- Final implementation should be much better (main project aim).
- 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?
- Most of kernel still locked by `Giant`.

Stability

- Prime goal in the short term: sacrifice performance for stability.

Stability

- Prime goal in the short term: sacrifice performance for stability.
- Only partially successful.

Stability

- Prime goal in the short term: sacrifice performance for stability.
- Only partially successful.
- Many developers no longer run `-CURRENT`.

Stability

- Prime goal in the short term: sacrifice performance for stability.
- Only partially successful.
- Many developers no longer run `-CURRENT`.
- Stability strongly dependent on hardware in use.

What a difference a month makes

Paper submitted in August. Since then,

- Matt Dillon ‘‘pushes down’’ system call locking.

What a difference a month makes

Paper submitted in August. Since then,

- Matt Dillon ‘‘pushes down’’ system call locking.
- Julian Elischer introduces kernel threads.

What a difference a month makes

Paper submitted in August. Since then,

- Matt Dillon ‘‘pushes down’’ system call locking.
- Julian Elischer introduces kernel threads.
- The release team decides to postpone the release by one year.

Further information

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

Join in! The FreeBSD project needs more clever hackers.

These slides are available at
`http://www.lemis.com/SMPng/AUUG2001/`