Bootchart 2

what you see under the hood ...
Feb 2010

Michael Meeks

michael.meeks@novell.com

"Stand at the crossroads and look; ask for the ancient paths, ask where the good way is, and walk in it, and you will find rest for your souls..." - Jeremiah 6:16

Novell

Overview

- Bootchart 1
  - a breakthrough for it's time
- Bootchart 2
  - slightly less lame
- Other useful & developing tools
- Why boot time is slow
  - and whether it should be ...
- How you can get involved helping out
- Conclusions

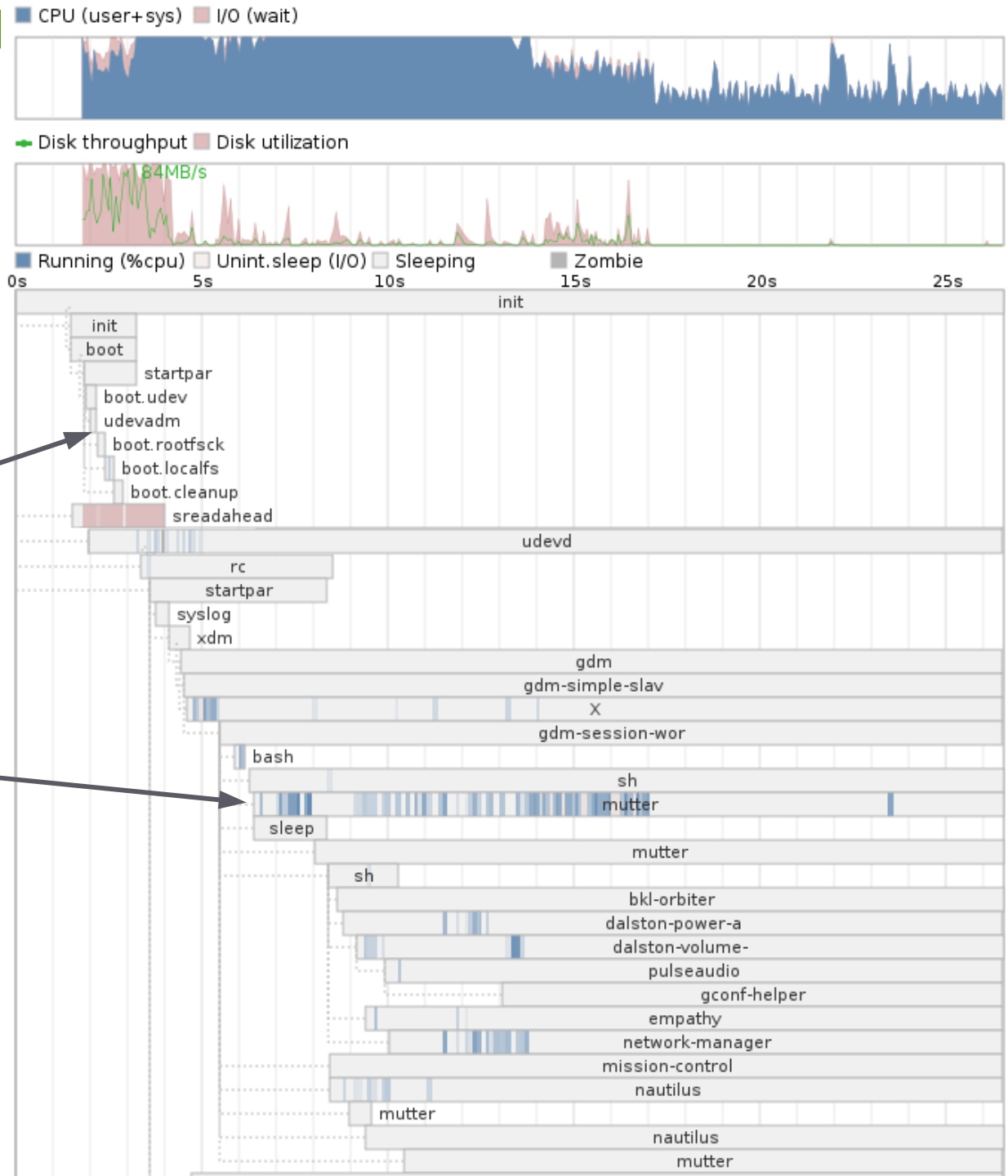
Bootchart 1

Bootchart 1 – it rocked

- http://www.bootchart.org
  - Ziga Mahkovec <ziga.mahkovec@klika.si>
  - Met Owen Taylor's challenge ...
  - showed us for the first time what was going on
- We noticed a lot was wrong in booting
  - and started to fix it.
  - boot times of 1 minute+ - common
  - years of accumulated bug fixes in booting:
    - *Yeah, that would be faster, but what about NFS root, with an NTP server, and remote syslogging !? - I need to fix that fast !*
  - the problems were **so** prevalent, this was no issue

Bootchart 1

- Poor resolution – 25 pixels per second, and lower res data collection.
- Processes appear to take no time, when we know they are busy: eg. boot.udev, udevadm
- Many processes appear to take no CPU time, even at startup when they are linking.
- Bootchart 1 - less truthful even than gdb !



Bootchart 1 – other issues ...

- Initial version rendered using **Java**
  - Not ubiquitous on Linux, requires compilation
- Enter: pybootchartgui
  - http://code.google.com/p/pybootchartgui
  - Anders Norgaard & Henning Niss' blow for freedom
  - Hackable: python / cairo rendering – to SVG, PNG
- Initial version data collection written in **shell**
  - `while true; do cat /proc/*/stat > log-file; done`
- Bootchart-collector:
  - Scott James Remnant 's contribution ...
  - re-write in C for faster collection
  - still using low-res `/proc/<task>/stat` data

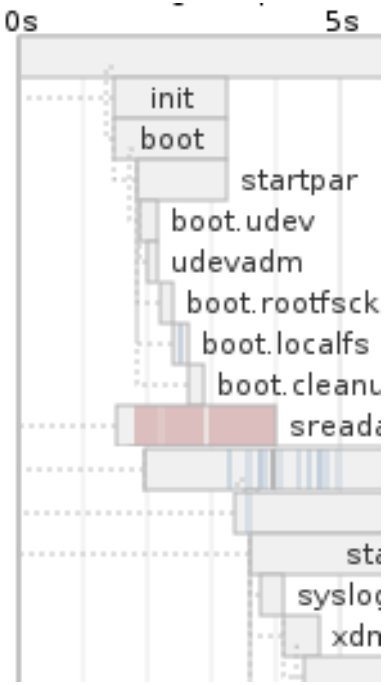
Bootchart 2

for the world of the
ten second boot

Bootchart 2 – a new approach

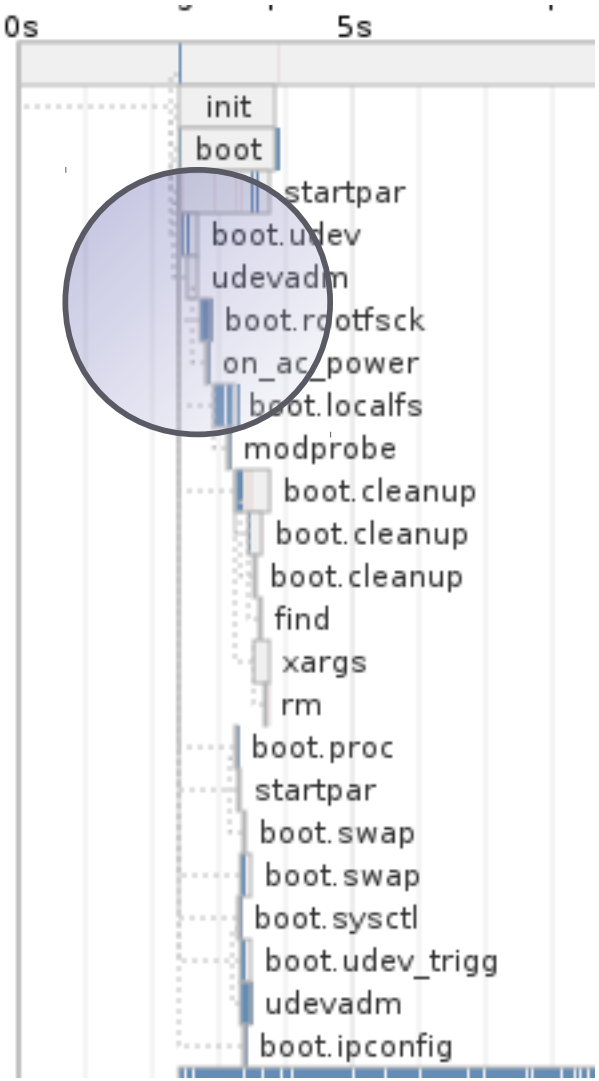
- I re-wrote bootchart-collector:
  - Use kernel's taskstats interface: **ns** accurate time accounting for processes.
  - Interface baroque, unpleasant, and inefficient
    - **But** – available in all shipping kernels (unless you turn it off -go Moblin!)
  - allows us to say: “which process used how much CPU”
- Integrates & improves pybootchartgui
  - better coupling with the collector - key.

Old



New, and differently broken bootchart !

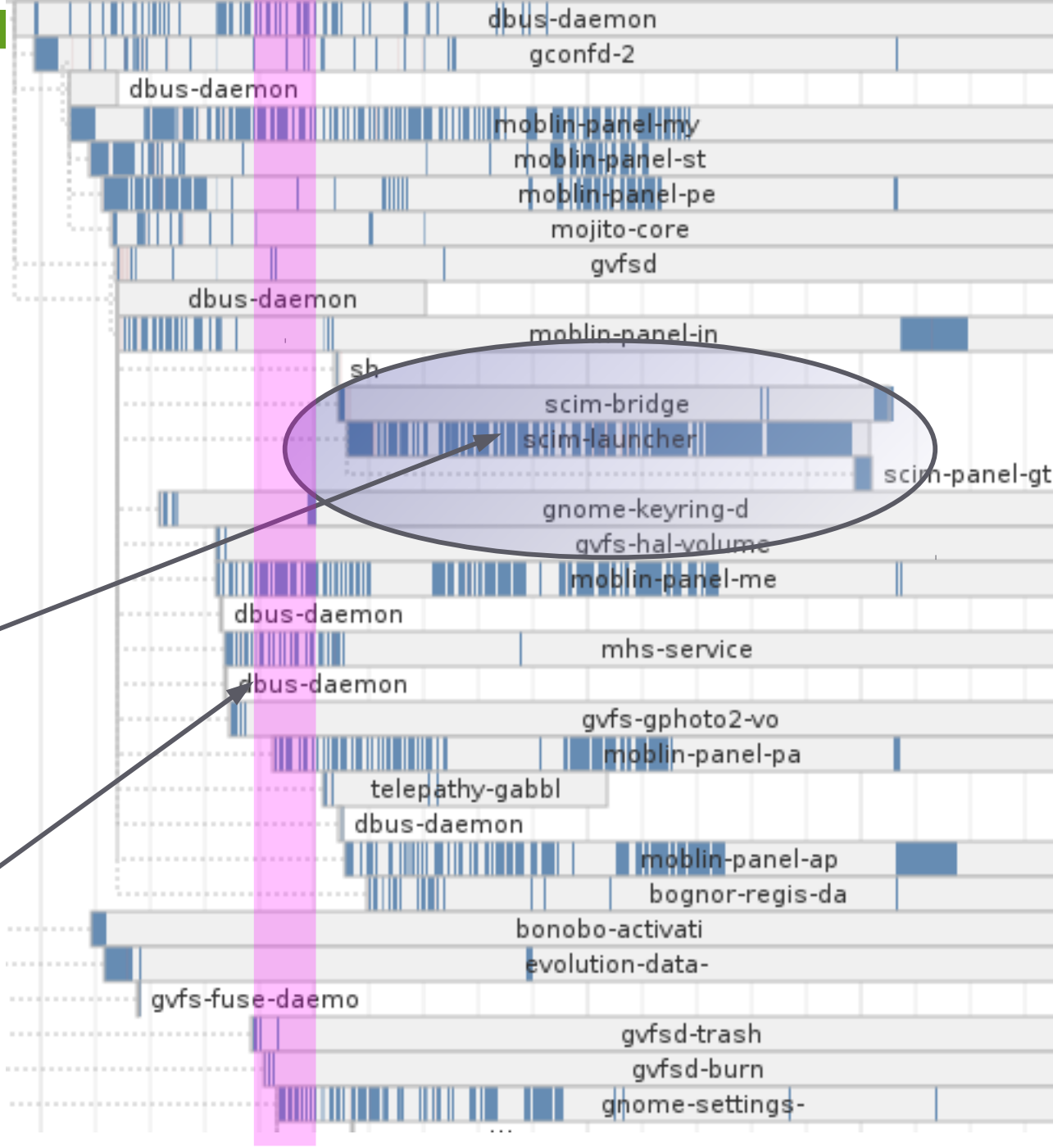
New



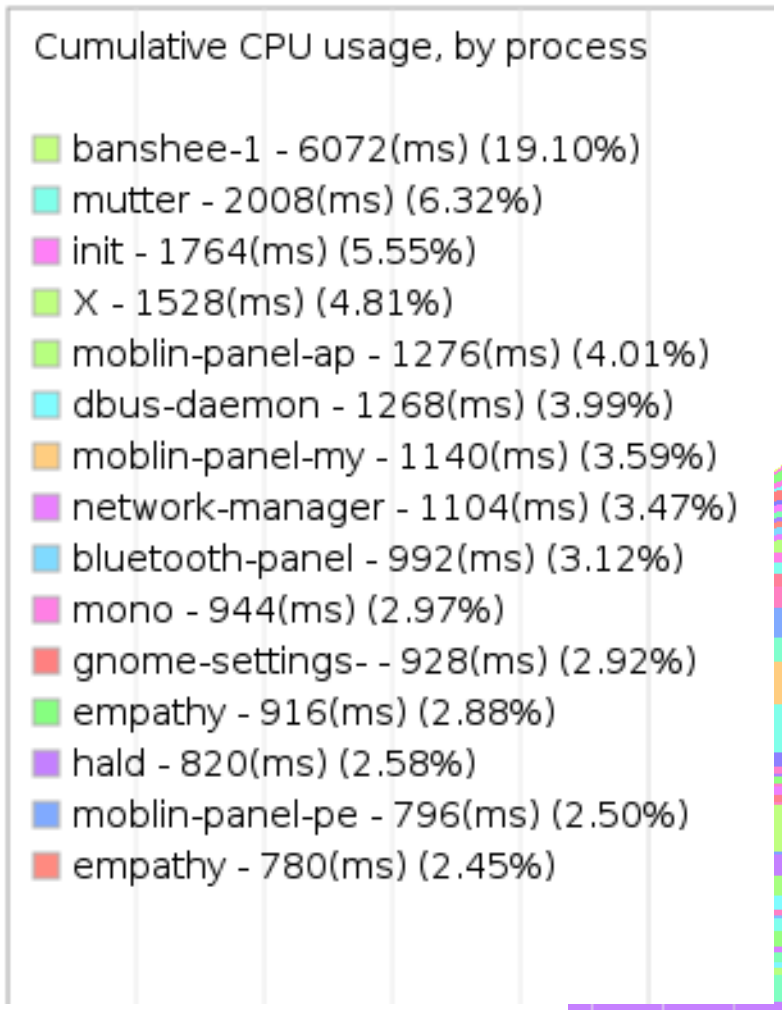
Which in bulk has lots of little lines:

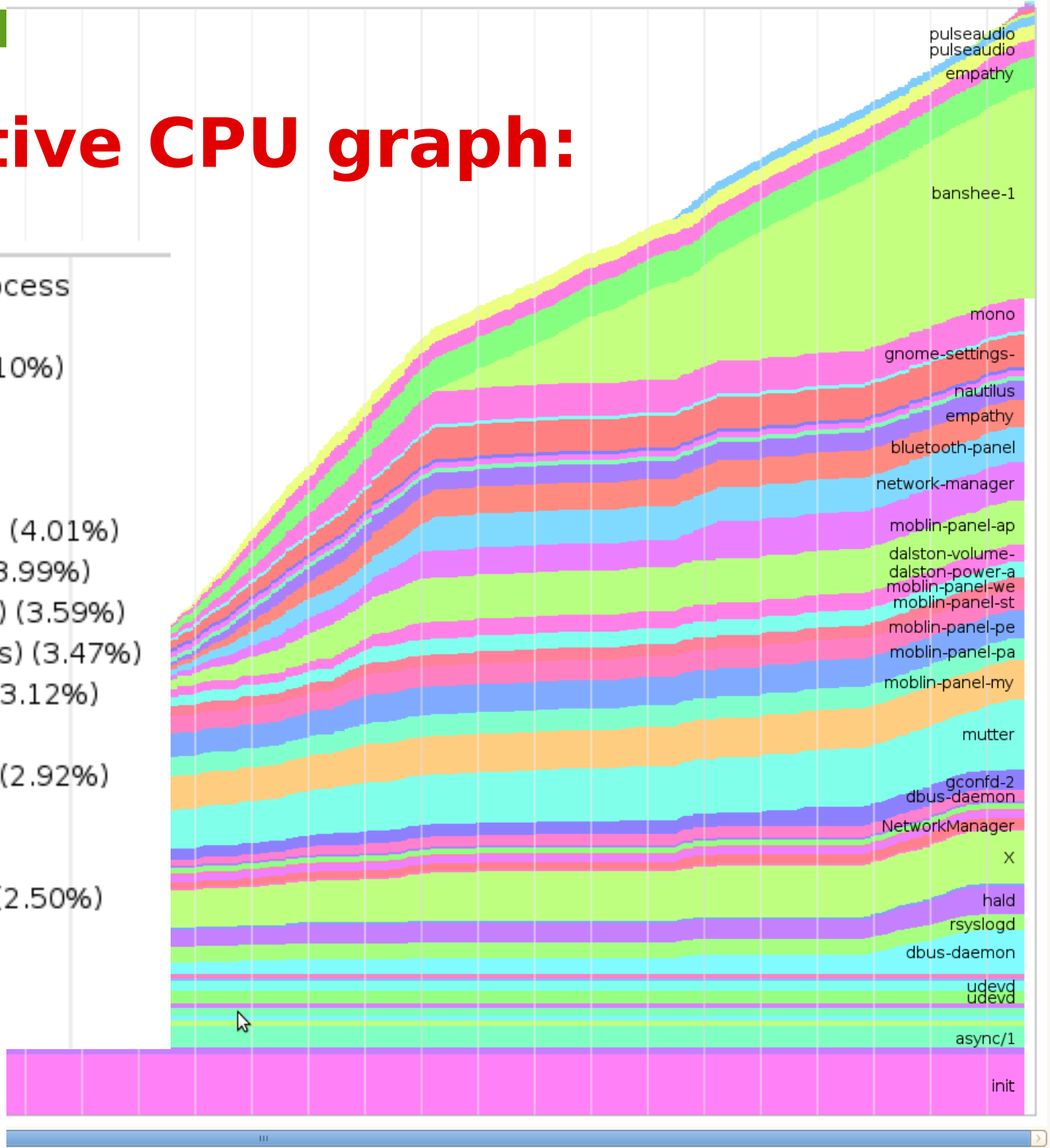
Who is a naughty process then ?

Killed feature to add alpha transparency based on %age of CPU used

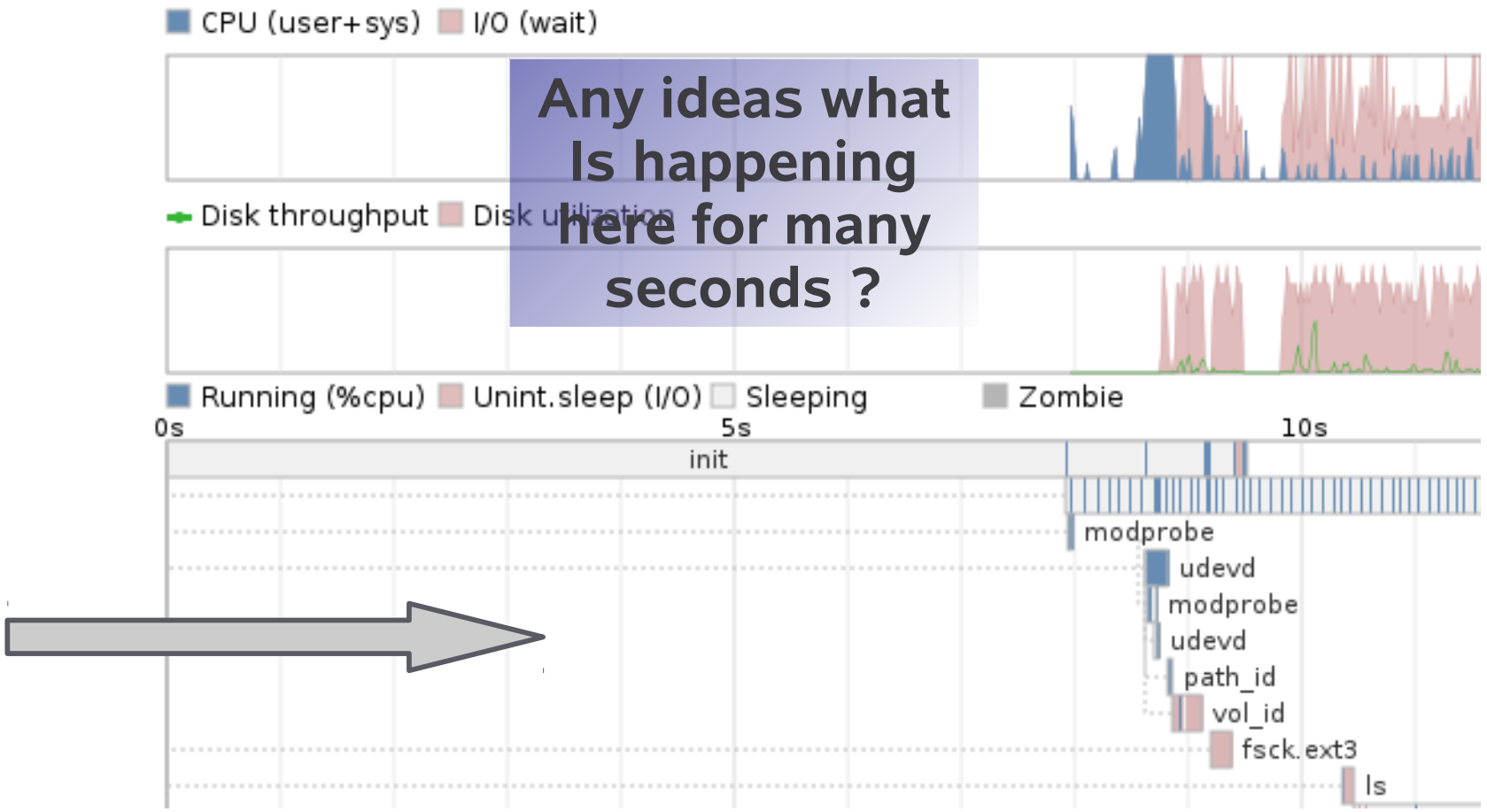


Cumulative CPU graph:

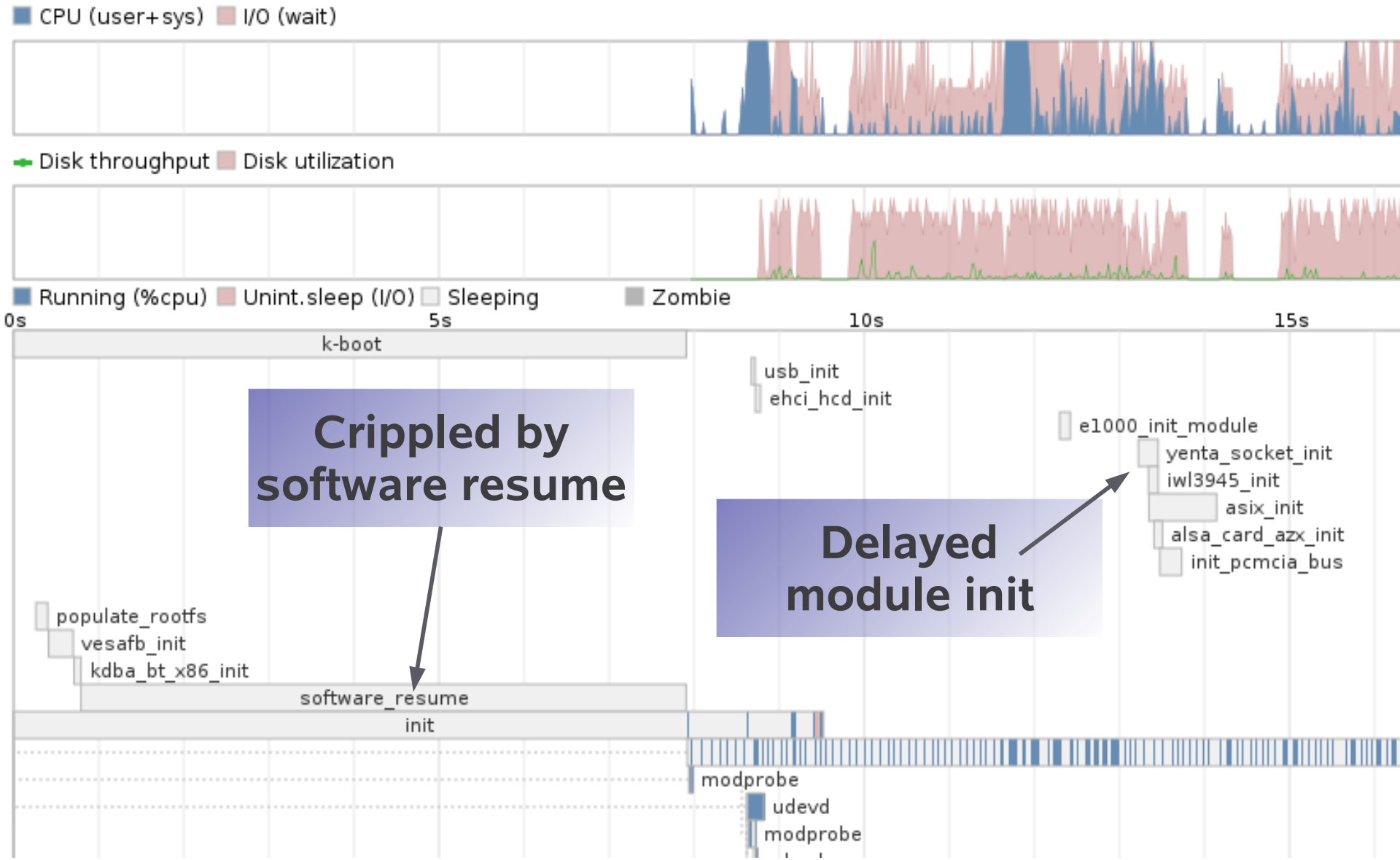




The icing – kernel boot-charting

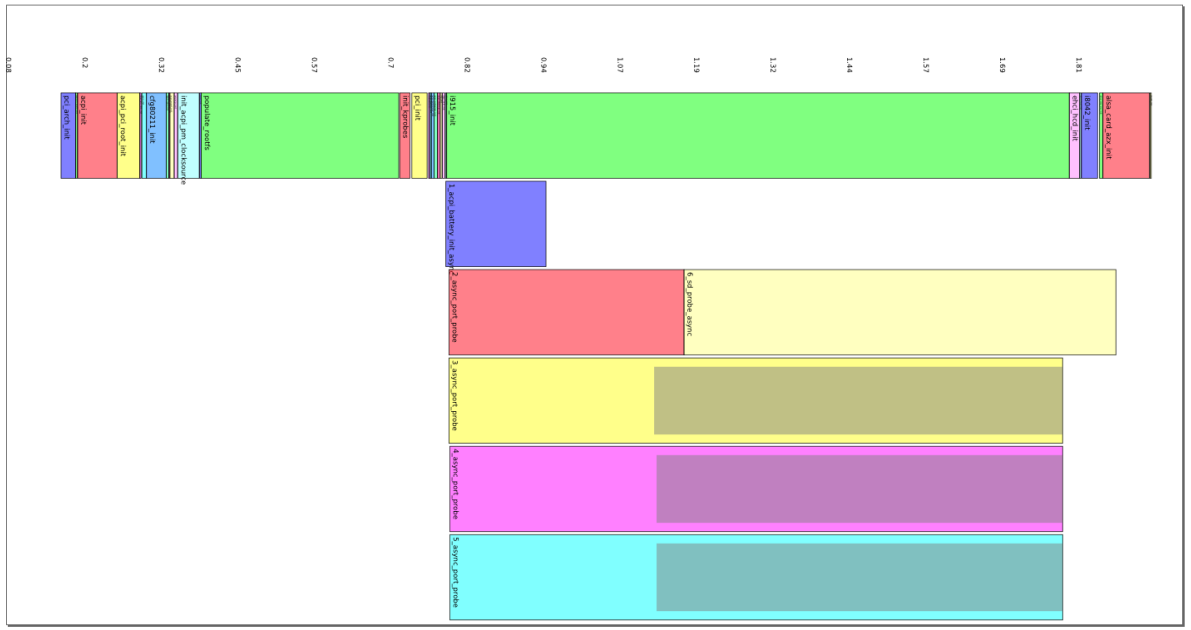


The icing – kernel boot-charting



More detailed kernel charting:

- Thanks to Arjan van de Ven
  - Stores, and parses dmesg output from the kernel
  - Kernel params: "`initcall_debug printk.time=y quiet`"
- In-kernel tool for better svg rendering:
  - dmesg | perl scripts/bootchart.pl > foo.svg

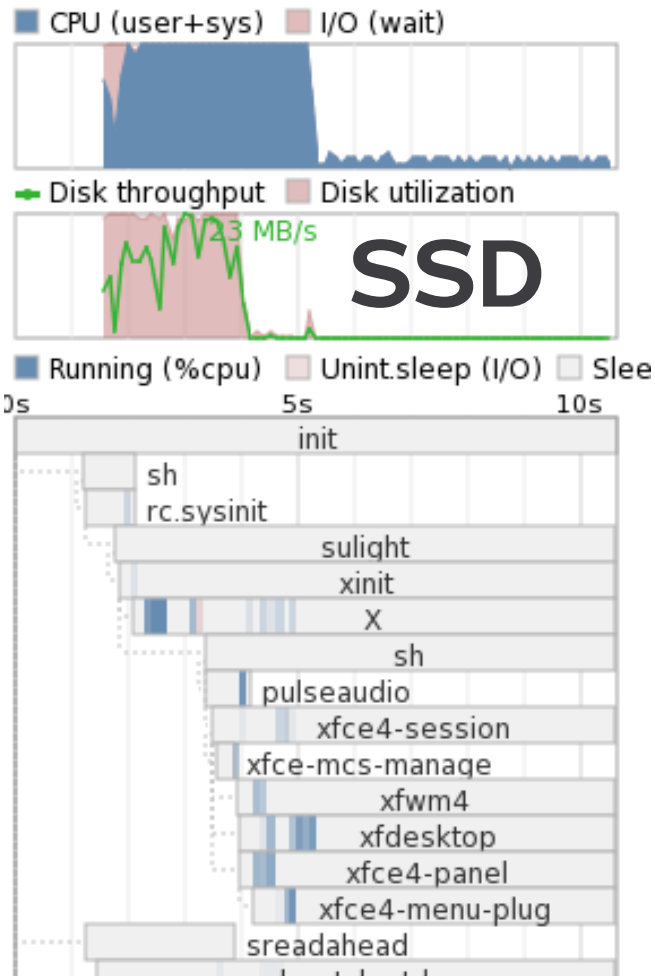


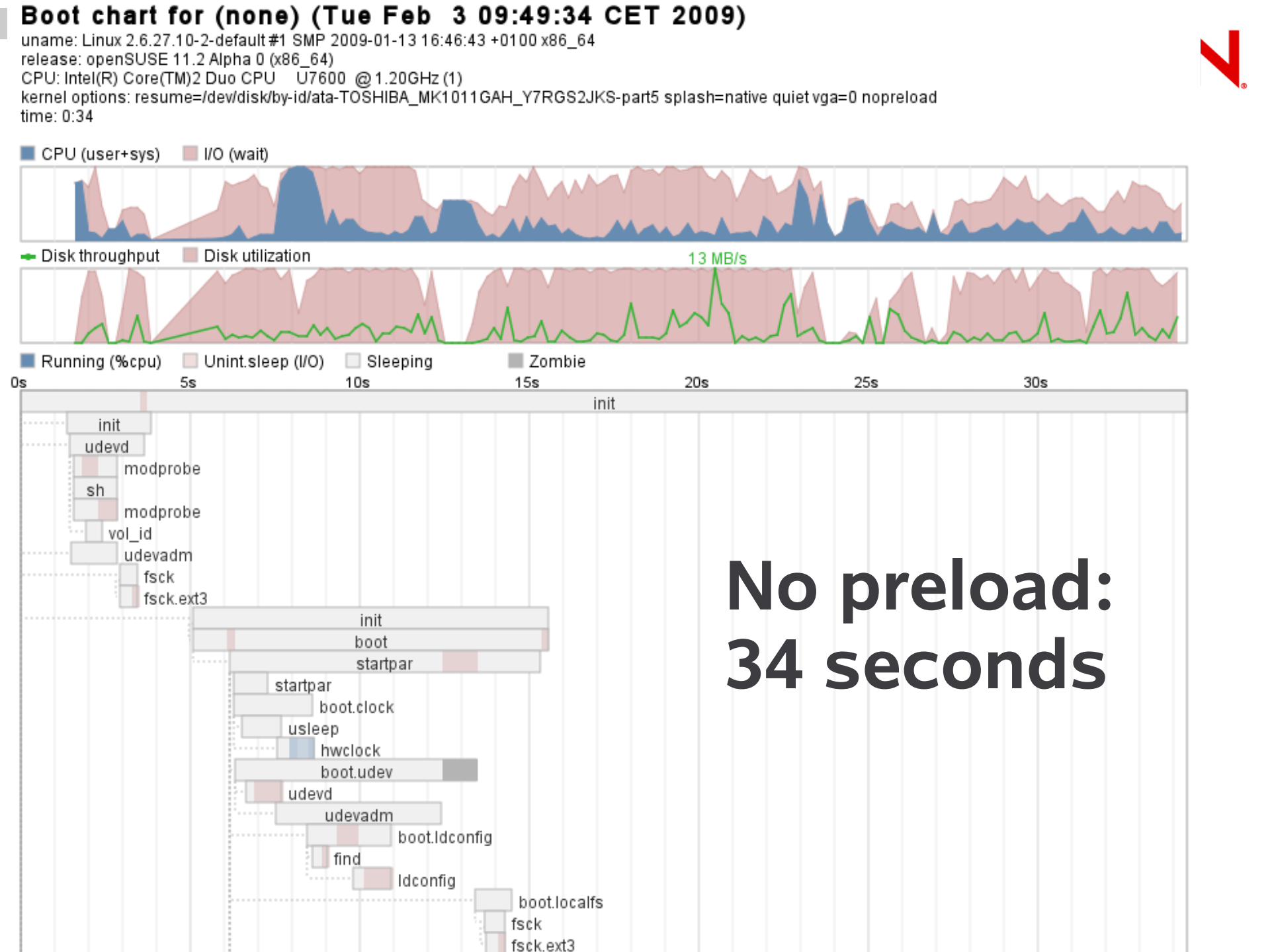
How to improve boot time ...

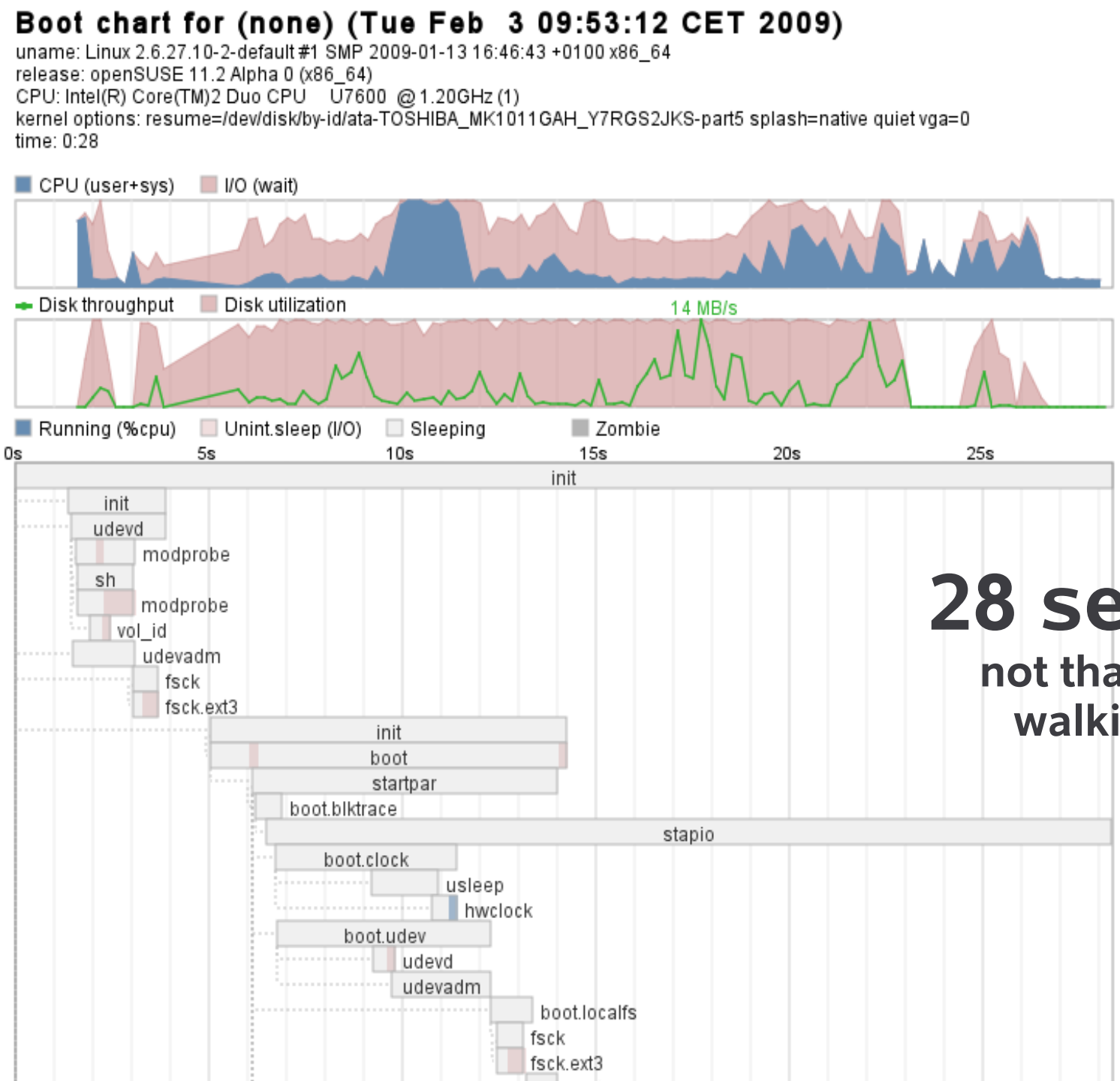
- Make everything asynchronous ?
  - Network Management arriving late
  - Udev, X, hal – races and issues
- Re-write all init scripts as a single shell script
  - No faster on SUSE (vs. insserv / parallelism)
  - Death by a thousand cuts in each component.
- Profile each piece and make it not suck:
  - modprobe: very slow: fixed, udev slow - accelerated.
  - Kernel module loading – monster lock contention: fixed
  - hal – doing millions of pointless allocate / frees
  - gconfd – under-performant parser, far too much XML
  - and it goes on: still plenty to do ...

I/O issues ...

- Interleave 'sleep' (or CPU) and 'read'




Boot chart for (none) (Tue Feb 3 09:49:34 CET 2009)
uname: Linux 2.6.27.10-2-default #1 SMP 2009-01-13 16:46:43 +0100 x86_64
release: openSUSE 11.2 Alpha 0 (x86_64)
CPU: Intel(R) Core(TM)2 Duo CPU U7600 @ 1.20GHz (1)
kernel options: resume=/dev/disk/by-id/ata-TOSHIBA_MK1011GAH_Y7RGS2JKS-part5 splash=native quiet vga=0 nopreload
time: 0:34
CPU (user+sys)
I/O (wait)
Disk throughput
Disk utilization
13 MB/s
Running (%cpu)
Unint.sleep (I/O)
Sleeping
Zombie
0s
5s
10s
15s
20s
25s
30s
init
init
udevd
modprobe
sh
modprobe
vol_id
udevadm
fsck
fsck.ext3
init
boot
startpar
startpar
boot.clock
usleep
hwclock
boot.udev
udevd
udevadm
boot.ldconfig
find
ldconfig
boot.localfs
fsck
fsck.ext3
No preload:
34 seconds



28 seconds:

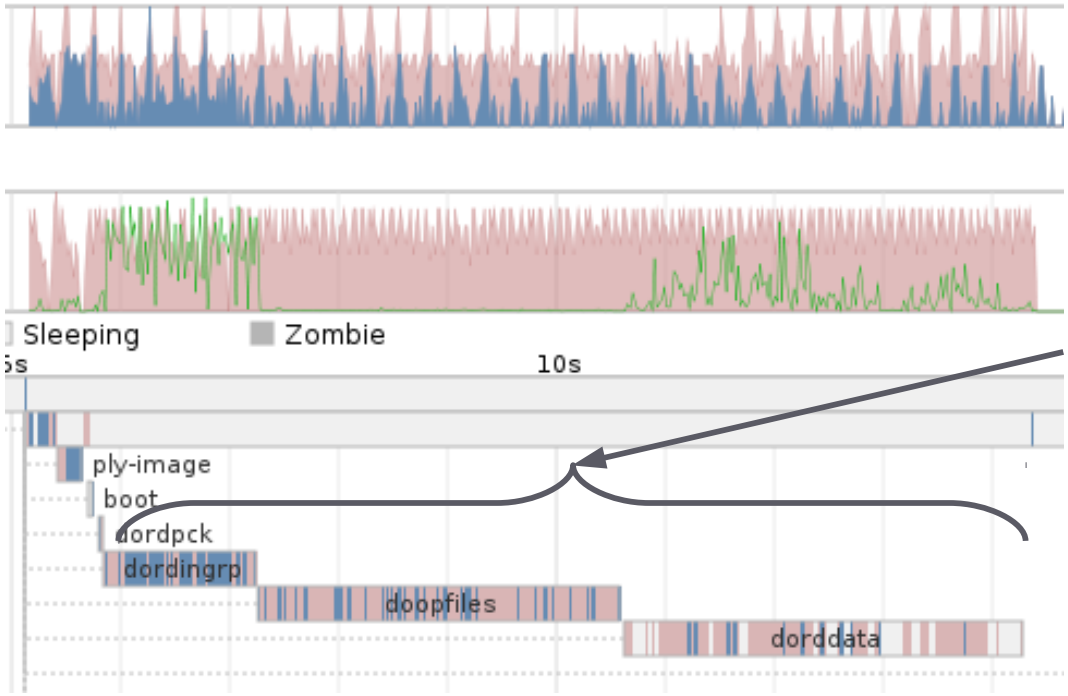
not that good, not a walking advert for 'preload'

Choice ! readahead(s)

- http://en.opensuse.org/Preload
  - uses system-tap to build a kernel module to trace I/O always.
- Other people uses a kernel patch of some sort: ftrace
  - https://fedorahosted.org/readahead/
  - http://sreadahead.googlecode.com/svn/trunk
    - Intel: ~dead. Optimised for Flash, not HDDs
  - https://launchpad.net/ureadahead
    - Cleaned up the sreadahead coding style, more flxible, generic, works for HDDs well.
    - Close to the © assignment disaster area: stay away !
    - **Blocking** reads – open ~100 files. sorts by file-system layout etc.
  - http://github.com/mmeeks/sreadahead
    - Cleaned up readahead, works better on HDD, sorts & tries to backtround

More tools and tricks:

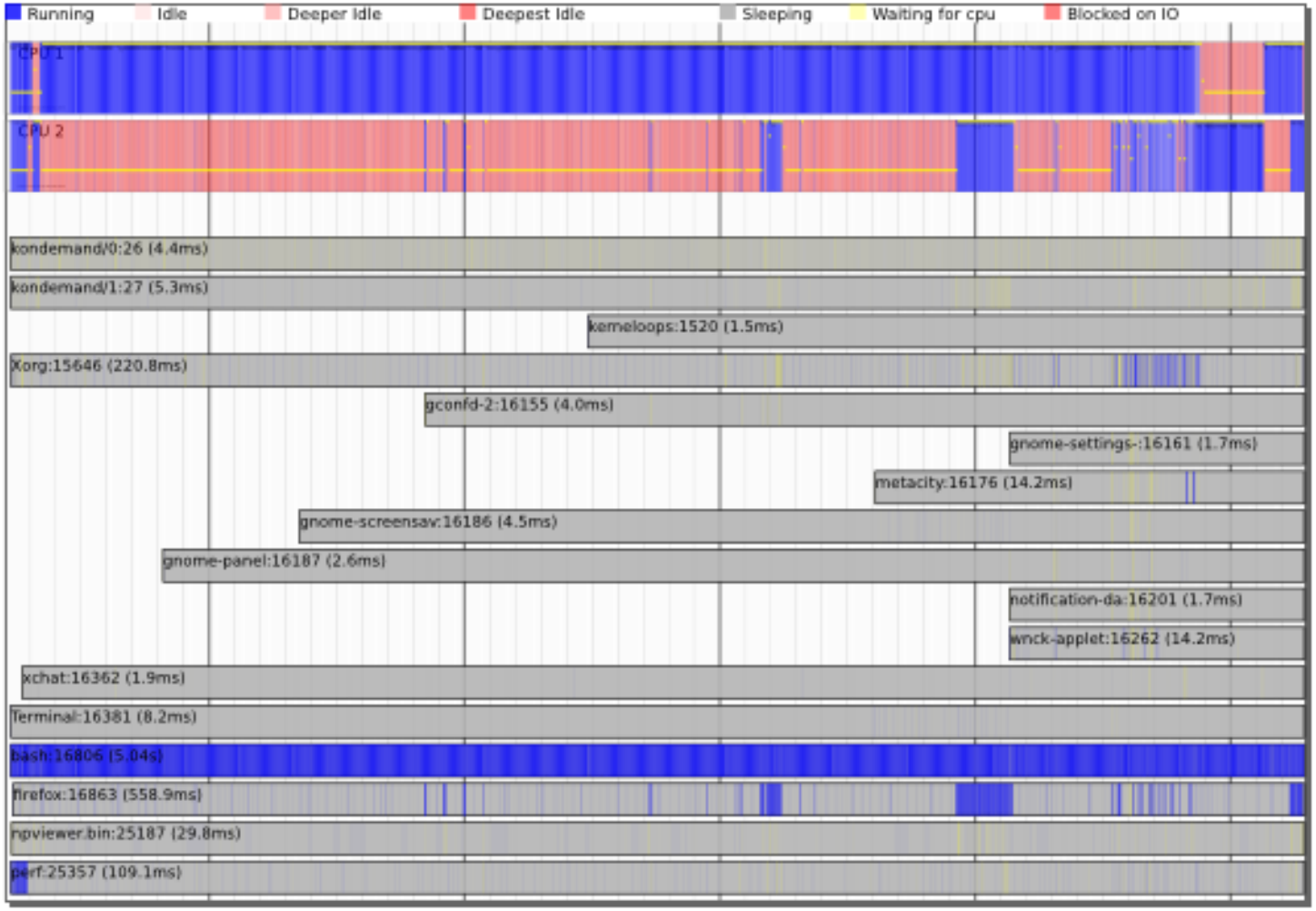
- A deeper dive:
  - The process is slow but where !?
  - `prctl(PR_SET_NAME, "HelloMu", 0, 0, 0);`



ureadahead

Timechart

- http://blog.fenrus.org/?p=5



Other bits ...

- Various magic scripts for **systemtap**
- **http://git.fedoraproject.org/git/?p=tuned.git**
- **http://git.fedoraproject.org/git/?p=tuned.git;a=blob_plai**
  - Various scripts of goodness
  - What processes are taking what time
- My sreadahead:
  - Disk I/O breakdown by directory and file
  - How much, and where does it all come from ?

```
/usr/lib/dri/i915_dri.so      2706
/usr/lib/dri                  2706
/etc                          2812
/lib                          3076
/usr/share                    4388
/usr/bin                      7098
/usr/lib                      32010
/usr                          45934
Total                         57789
```

Conclusion / Q&A

- Bootchart2 reaches places other boot charts cannot.
  - http://github.com/mmeeks/bootchart
  - Plenty more to do there, grab me afterward
    - python hackers ? ****Package Me !****
- It is possible to boot fast
  - How fast is unclear
  - Less is more – or less ?
- Moblin rocks ← gratuitous plug.
- Thanks – to all the people that did it [mostly not me]

Oh, that my words were recorded, that they were written on a scroll, that they were inscribed with an iron tool on lead, or engraved in rock for ever! I know that my Redeemer lives, and that in the end he will stand upon the earth. And though this body has been destroyed yet in my flesh I will see God, I myself will see him, with my own eyes - I and not another. How my heart yearns within me. - Job 19: 23-27