Boot Linux Faster

USTC

Fengguang Wu

AKA 2006 Linux kernel conference

Sep 16, 2006

WE NEED FAST BOOT

- **everybody likes it**
- **everybody cares**
- **we are put to shame by**
  - Windows with **Prefetcher**...
  - Mac OS X with **BootCache**...

the expert way

- less services
- less fonts
- less features
- light weight GUI

...

- `false && depmod -a`

keep ...

slim & fluent

the hacker way

```
kernel init=/bin/sh
```

the shortcut

suspend

&

resume

the comprehensive way

hey, let's do all possible

to ensure

fast boot with all the goodies!

Embedded Systems

CE Linux Forum

http://tree.celinuxforum.org/pubwiki/moin.cgi/BootupTimeResources
http://tree.celinuxforum.org/pubwiki/moin.cgi/BootupTimeReductionHowto

OLS2006

Improving Linux Startup Time Using Software Resume (and other techniques)
https://ols2006.108.redhat.com/reprints/kaminaga-reprint.pdf

Linux Bootup Time Reduction for Digital Still Camera
https://ols2006.108.redhat.com/reprints/park-reprint.pdf

Major Roadblocks

- **I/O seek frenzy**
  - 1k ~ 10k files on startup
  - 5k * 8ms = 40s
- **buggy apps**
  - silly spins / sleeps
    - IDE probe delays – up to 3s
  - Dave Jones: Why Userspace Sucks
- **CPU hogs**

General Solutions

`scheme`	`helps`
▪ bootchart	Analyze
▪ prelink	CPU
▪ parallelization	CPU / IO idles
▪ preload(focus)	I/O wait / seeks
▪ defrag	I/O seeks

Kernel Tricks

- **kernel options**
  - `kernel` **`quiet ide3=noprobe`**
  - `mount -o` **`noatime`**
- **`kexec` for fast reboot**
  - run a new kernel ***instantly***
- **parallel device initialization**
  - ongoing work by Greg KH

kexec mini HOWTO

- kernel `CONFIG_KEXEC=y`
- install `kexec-tools`
- hack `/etc/init.d/reboot`

```
# try kexec, then fall back to reboot
 /sbin/kexec -e
 /sbin/reboot -d -f -i
```

- load a kernel and reboot

```
 # kexec -l /vmlinuz \
         --append="$(</proc/cmdline)" -x
 # reboot
```

Prelinking

- **idea**
  - prelink(modify) ELF libs and apps
  - to speed up dynamic linking
- **benefits**
  - C++ apps linking to many libs
    - GNOME / KDE
  - load time not run time
  - CPU bound not I/O bound

Parallelization

Overlapping execution with I/O.

- **most distributions on the way**
- **SUSE already there**
  - LSB `initserv/chkconfig` tools
- **Ubuntu shows the future**
  - upstart

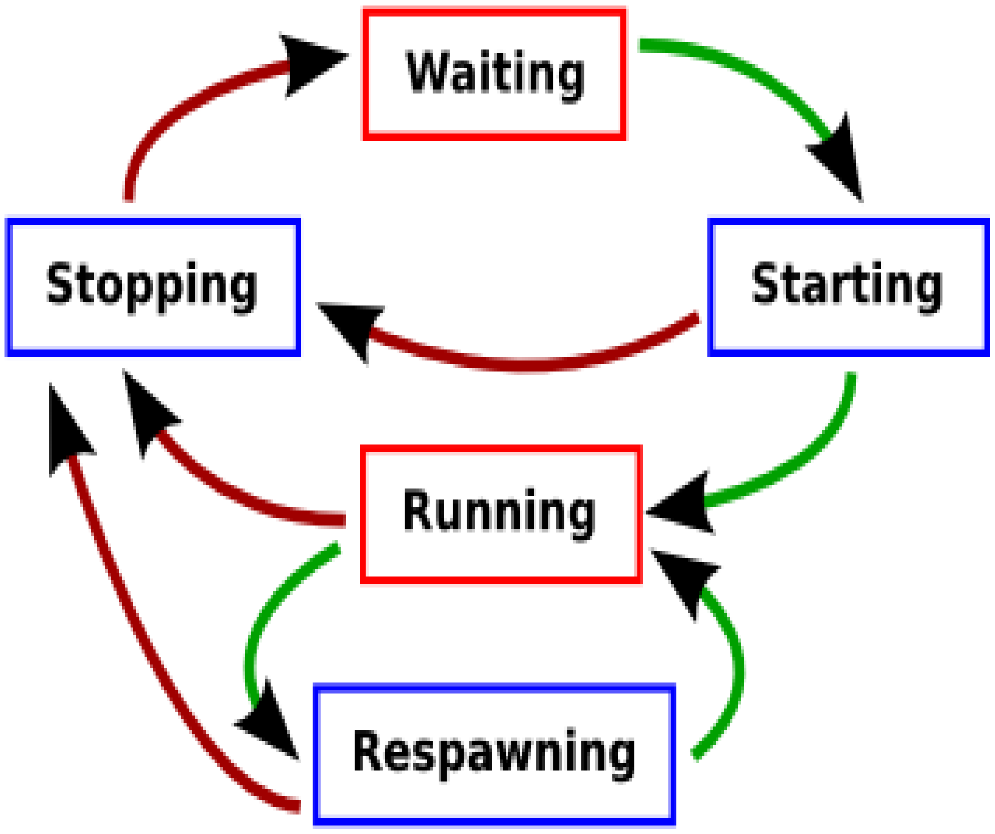
SUSE LSB boot

- **dependency based**
  - LSB standard header

```
### BEGIN INIT INFO
# Provides:         cron
# Required-Start: $remote_fs $syslog $time
# Should-Start:     $network sendmail postfix
# Required-Stop:    $remote_fs $syslog
# Default-Start:    2 3 5
# Default-Stop:     0 1 6
# Description:      Cron job service
### END INIT INFO
```

Ubuntu Upstart

- **event based**
- **components**
  - init
  - jobs
  - events
- **goal**
  - to replace init,
    and cron, inetd, ...



job life-cycle

Preload

- **startup is I/O bound**
- **goal**
  - reduce I/O wait
  - better I/O utilization
- **steps**
  - collect I/O trace
  - preload files
  - defrag files

Distribution Solutions

- **debian**
  - preload
- **ubuntu**
  - readahead
- **gentoo**
  - readahead-list-early
  - readahead-list
- **suse**
  - boot.preload_early
  - boot.preload
  - earlykdm
- **fedora**
  - readahead_early
  - readahead
  - xdm preload

Reports

- **preload.sf.net**
  - 65s to 85s reported by its author
  - 53s to 51s reported by Carlos Villegas
- **ubuntu readahead**
  - 49s to 49/50s from Carlos Villegas
- **suse boot.preload**
  - 42.2s to 41.9s from Fengguang Wu

not so favorable

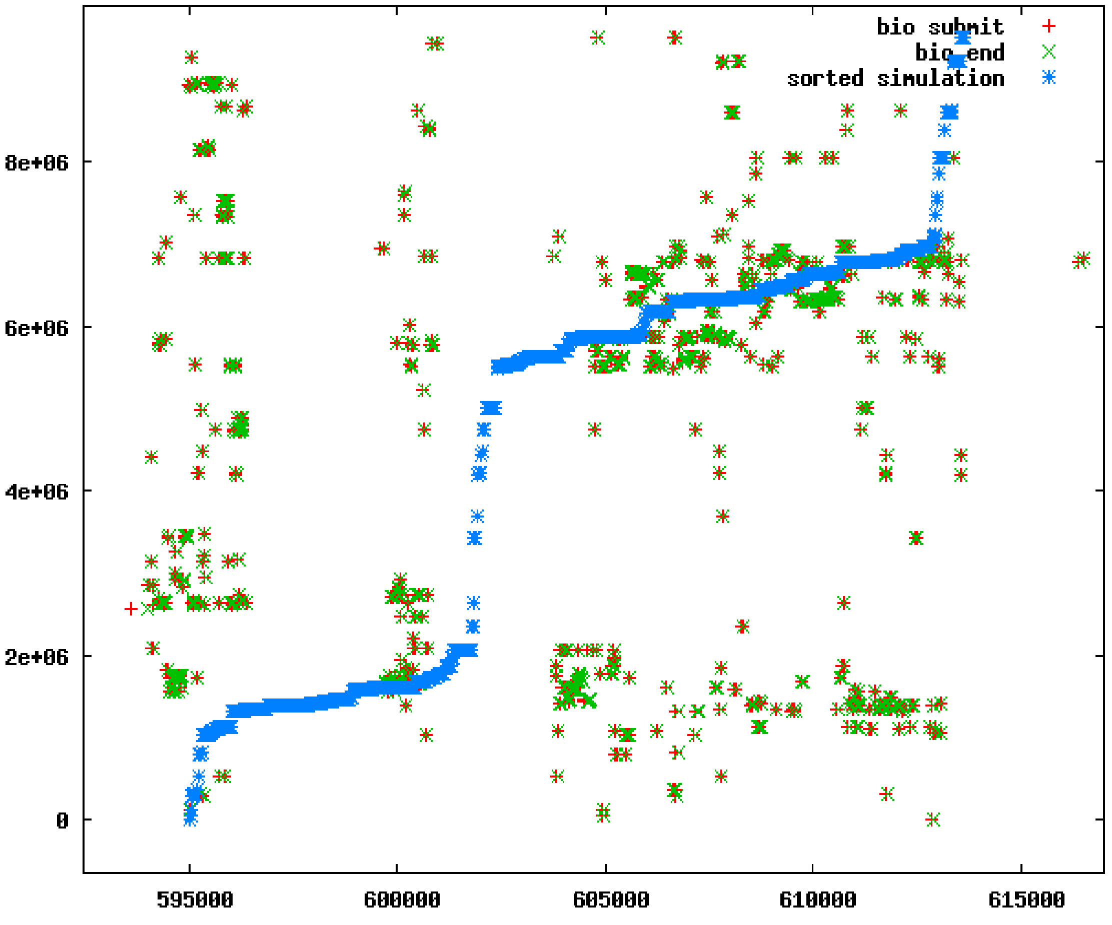
Ideal Preload

- complete in background
- read in order

Random

vs

Sorted



fcache by Jens Axboe

- **prime mode**
  - mirror read data to cache device
- **normal mode**
  - serve data from cache device
- **merits and demerits**
  - turn random accesses to linear ones
  - perfect layout, optimal I/O
  - not quite linux way

the Flavours

`fcache`

- perfect
- kernel solution
- specialized
- weird

  for enduser

the Flavours

`fcache`	`filecache`
perfect	good enough
kernel solution	kernel+userland
specialized	general purpose
weird for enduser	easy deployment

Proposed Solution

- **I/O trace**
  - `/proc/filecache` interface and tools
- **preload**
  - request for **all** files **ASAP**, so that I/O scheduler knows the **global picture**
  - schedule the **bulk** requests in kernel
- **data layout**
  - poor man's defragger for ext3

Collecting I/O Traces

▪ strace	SUSE
▪ LD_PRELOAD	gentoo-wiki.com
▪ /proc/<pid>/map	preload.sf.net
▪ fboot (kernel module)	Andrew Morton

`strace` Practice

To trace KDE startup initiated from KDM, we have to strace two `kdm` processes and the `X` process (PIDs 3703, 4100 and 9573):

```
# strace -f -F \
           -p3703 -p4100 -p9573 \
           -e trace=open \
           -o strace.kdm
```

/proc/filecache inodes view

```
# echo -n index > /proc/filecache
# cat /proc/filecache
```

```
# filecache 1.0
#    ino size cached cached% state refcnt dev          file
  472353    8      0       0  --    29     03:42(hdb2) /lib
 1205314   91     92     100  --    65     03:42(hdb2) /lib/ld-
2.3.6.so
   16289    8      0       0  --    38     03:42(hdb2) /etc
  472394    4      0       0  --    30     03:42(hdb2) /lib/tls
  233608 1242   1056      85  --    65     03:42(hdb2)
/lib/tls/libc-2.3.6.so
   65203  651    496      76  --    2      03:42(hdb2) /bin/bash
 1205315  261    160      61  --    10     03:42(hdb2)
/lib/libncurses.so.5.5
```

/proc/filecache
pages view

```
# echo -n /bin/bash > /proc/filecache
# cat /proc/filecache
```

```
# file /bin/bash
# flags R:referenced A:active U:uptodate D:dirty
W:writeback M:mmap
# idx   len     state   refcnt
0       46      RAU___  1
47      12      RAU___  1
60      13      RAU___  1
73      4       __U___  1
77      4       RAU___  1
81      2       __U___  1
83      6       RAU___  1
```

`filecache` Merits

- convenient
- no overhead
- shows cache usage
- shows fs metadata
- I/O trace for any task
  - take `snapshot0;` run task; take `snapshot1`
  - **`diff`** `snapshot1 snapshot0`

Preload Steps

```
setup queue parameters

for each fs:
  readahead fs metadata
  wait for fs mount
  readahead files in parallel

wait for IO complete
restore queue parameters
```

I/O Schedule

- **read-ahead requests**
  - `IOPRIO_CLASS_IDLE`
  - served on disk idle or in batch
  - need fix deadline and anticipatory
- **on pending read**
  - find the read-ahead request
  - queue it for submission
  - need fix all elevators

ext3 data layout

- **allocation group**
  - fs divided into equal sized groups
  - default to 80x128M for a 10G fs
- **allocate strategy (orlov)**
  - spread out "top-level" directories
  - ensure locality for normal inodes
  - makes unbalanced use of groups
    - `dumpe2fs /dev/hda2 | grep free`

ext3 top-level dirs

- **concept**
  - directories on the fs root
  - or: `chattr -T dir`
- **orlov**
  - find a moderate spare group
- **oldalloc**
  - find the most spare group (for all dirs)

ext3 poor man's defrag

```
mount -o remount,oldalloc /
mkdir /.defrag-habitat
mount -o remount,orlov /

for f in $files
do
  cp $f /.defrag-habitat/tmpf
  rm $f
  mv /.defrag-habitat/tmpf $f
done
```

Put it Together

- **elementary tools**
  - `/proc/filecache` snapshot

```
# filecache --snapshot --dump ./fc
# ls ./fc
bdev hdb2 hdb3
```

  - readahead files in parallel

```
# readahead-fs ./fc/hdb2
```

Put it Together

- **front-end tool**
  - collect I/O trace

```
# bootcache start firefox
$ firefox &
$ sleep 10s
# bootcache stop firefox
```

  - preload files

```
# bootcache preload firefox
$ firefox &
```

Let's go...

Debian:	95s to 68s	down 28%
SUSE:	49s to 50-60s	up 10%

What goes wrong?

lock contention

```
$ ps -C readahead-fs m -o \
    pid,tid,class,pri,pcpu,stat,wchan:14,comm

  PID   TID PRI %CPU STAT WCHAN          COMMAND
 9413     -   -  1.1 -    -              readahead-fs
    -  9413  24  0.1 Sl+  futex_wait     -
    -  9414  21  0.0 Dl+  sync_buffer    -
    -  9415  21  0.0 Dl+  real_lookup    -
    -  9416  21  0.0 Dl+  real_lookup    -
    -  9417  21  0.0 Dl+  sync_buffer    -
    -  9418  21  0.0 Dl+  real_lookup    -
    -  9419  21  0.0 Dl+  real_lookup    -
...
```

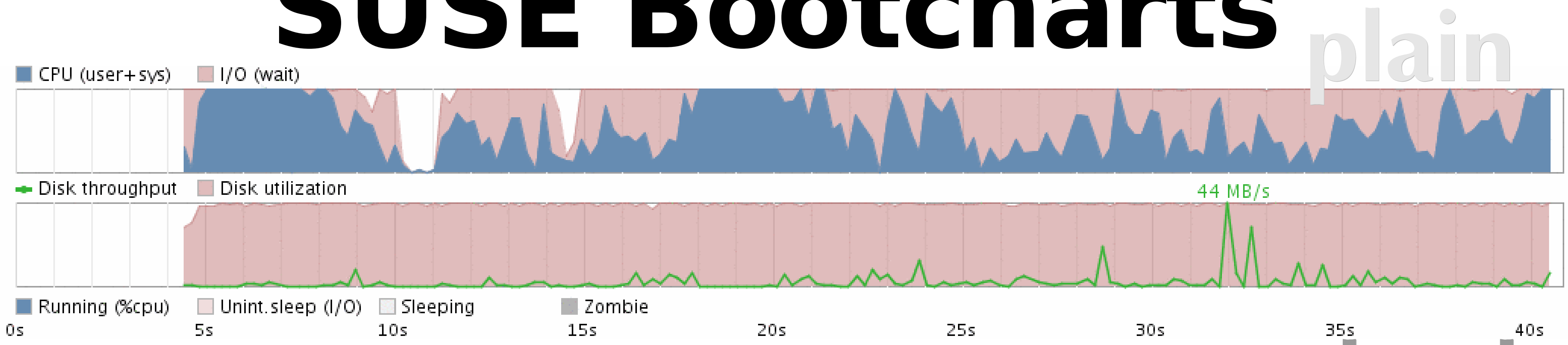
lock contention

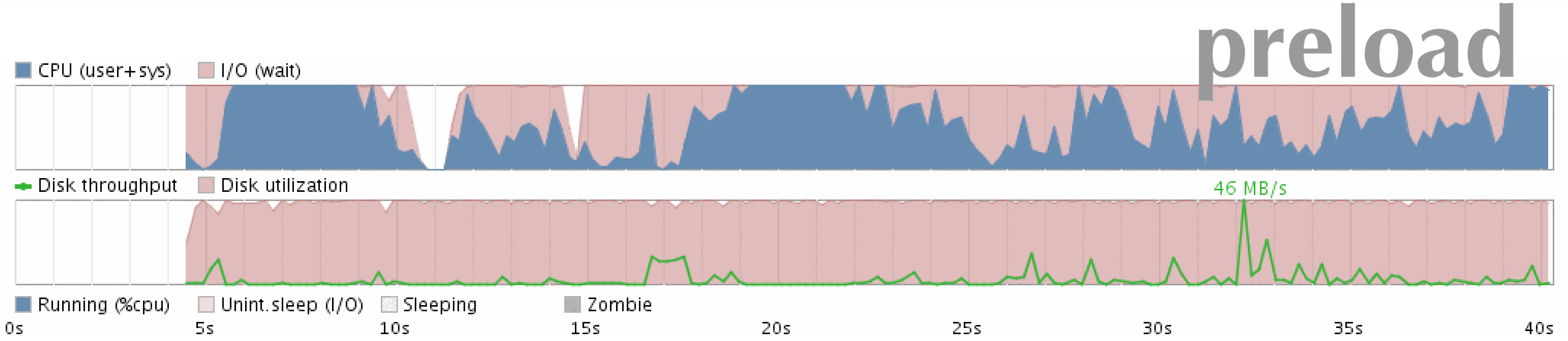
```
[<c01616c8>] sync_buffer+0x60/0x77
[<c03acbf6>] __wait_on_bit+0x58/0x61
[<c03acc7f>] out_of_line_wait_on_bit+0x80/0x88
[<c016175a>] __wait_on_buffer+0x31/0x33  <= wait I/O
[<c01ae0a5>] ext3_find_entry+0x16c/0x3d4
[<c01ae57d>] ext3_lookup+0x3c/0xe4
[<c016e214>] real_lookup+0xb5/0xd4  <= acquire mutex
[<c016e4c4>] do_lookup+0x94/0x9f
[<c016ec66>] __link_path_walk+0x797/0xe56
[<c016f369>] link_path_walk+0x44/0xba
[<c016f6b8>] do_path_lookup+0xe2/0x240
[<c016fb3b>] __user_walk_fd+0x48/0x5d
[<c0169dfb>] vfs_stat_fd+0x22/0x59
[<c0169e52>] vfs_stat+0x20/0x24
[<c016a530>] sys_stat64+0x1b/0x37
[<c0102cd3>] syscall_call+0x7/0xb
```

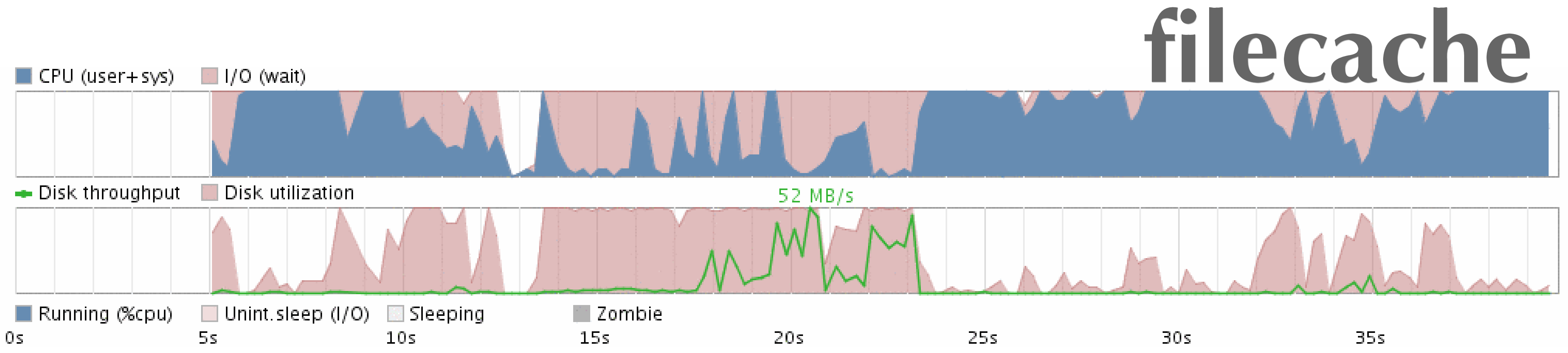
iopen

- **problem**
  - call sequence: `open(); read()`
  - stuck in `open()`, before requesting I/O
  - can't tell I/O elevator the global picture
- **open by inode**
  - ext3 inode number == offset inside fs
  - no path lookup
  - no lockup

SUSE Bootcharts







Conclusion

- **startup time remains the same**
- **still suffers from a lock contention**
  - leave as future work
- **seek frenzies reduced**
- **KDE startup is now CPU bound**
  - time to show off your dual core CPU ;-)
  - I'll go for `prelink` ^_^

Thank you.

Acknowledgment

Ongoing Projects

Google Summer of Code 2006

- Improve the Debian Boot Process
  http://initscripts-ng.alioth.debian.org/

- Rapid linux desktop startup through pre-caching
  http://code.google.com/p/pagecache-tools/

upstart: init daemon replacement
http://www.netsplit.com/blog/work/canonical/upstart.html

Resources

bootchart: Boot Process Performance Visualization
http://www.bootchart.org

Analyzing and Improving GNOME Startup Time
http://www.gnome.org/~lcolitti/gnome-startup/analysis/

10 Things Apple Did To Make Mac OS X Faster
http://www.kernelthread.com/mac/apme/optimizations/

kernel facilities for cache prefetching
http://marc.theaimsgroup.com/?t=114655640400004&r=2&w=2

Linux: Boot Time Speedups Through Precaching
http://kerneltrap.org/node/2157