bootchart2, e4rat and readahead

can we speed up the boot process?

Nick Morrott

October 2012

Motivations

- **monitor HDD/SSD performance**
- **increase boot speed**
- identify bottlenecks
- identify unused services

before we start...

- hardware used:
  - IBM/Lenovo X60s laptop (1.66GHz)
  - Western Digital Scorpio Blue HDD (80GB)
  - Samsung 830 series SSD (128GB)
- software used:
  - Debian GNU/Linux testing (wheezy)
  - 3.2.0 kernel

monitoring performance

what tools do you need?

- **bootchart2** is a kernel data collection tool (written in C)
- **pybootchartgui** is an interactive boot process viewer (python/GTK2)
- to install:

```
# apt-get install bootchart2 pybootchartgui
```

bootchart2 usage

- to grab kernel performance data, append the following to the kernel command line (e.g. 'e'dit an entry in the grub menu):

  quiet initcall_debug printk.time=y init=/sbin/bootchartd

- results stored in /var/log/bootchart.tgz
- no kernel modifications needed

pybootchartgui usage

- once boot data has been collected, you can analyse it interactively in the GUI:

  $ pybootchartgui -i

- generate visuals of the boot process:

  $ pybootchartgui -f [png|svg|pdf] -o PATH

- grab the boot time:

  $ pybootchartgui --boot-time

increasing boot speed

speeding things up

boot times can be reduced by:

- increasing raw disk I/O performance
- optimising caching/placement of files
- starting only the services you need
- altering the order services are started
- altering the way services are started

increasing raw disk I/O

- **problem:** boot process produces a lot of random reads and writes (many small files)
- simple ways to increase disk I/O:
  - using a mechanical disk with faster rpm

    4,200 < 5,400 < 7,200 < 10,000
  - using a solid state disk (SSD)
  - using a faster disk interface

    1.5Gb/s < 3Gb/s < 6Gb/s

optimising file caching/placement

- rotating disks have inherent latency whilst repositioning read/write heads between reads
- caching or optimising the physical placement of files should therefore reduce seek times

  (Note that SSDs have uniform access times across the device)

readahead

- aims to optimise boot process using a readahead cache (no physical reordering of files on disk or ext4 dependency)

  # apt-get install readahead-fedora

- comprises **readahead-collector** which generates list of files used during boot, and **readahead** which populates cache of required files early in boot sequence

readahead usage

- to trigger an update manually (e.g. after significant changes to services):

  ```
  # touch ./readahead_collect
  ```

  (reboot the system to regenerate file list)

- run the readahead-collector tool monthly via cron to keep the list of cached boot files optimised over time

e4rat

- moves boot files into contiguous sequence of blocks, allowing fast sequential reads
- loads boot files into readahead cache, massively increasing cache hit rate
- requires ext4 and 2.6.31+ kernel (uses the ext4 online defragmentation ioctl)
- can make significant improvements to boot speed (from 45s to 15s in example data)

e4rat usage

- **e4rat-collect**
  - add 'init=/sbin/e4rat-collect' to kernel cmd
- **e4rat-realloc**
  - switch to runlevel 1
  - e4rat-realloc /var/lib/e4rat/startup.log
- **e4rat-preload**
  - add 'init=/sbin/e4rat-preload' to kernel cmd

init/service configuration

- Dependency-based sysvinit configuration aims to ensure services do not block needlessly
- new init daemon replacements (e.g. Upstart, systemd) use architectures designed to increase concurrency and/or start services when required

init/service configuration

- Debian determines complex service dependencies automatically using insserv, generating symlinks in /etc/rcS.d
- it is possible to manually reorder services but managing dependencies can be complex and configs overwritten
- symlinks processed lexicographically

identifying bottlenecks

spotting CPU bottlenecks

- bootchart2 includes output showing processes in order of CPU usage
- makes it easy to see processes which are taking a lot of CPU (differences between HDD/SSD likely small)
- **udevd** and **modprobe** are likely to be amongst the most CPU intensive (module loading/device mgt)

spotting I/O bottlenecks

- bootchart2 also includes output showing processes in order of I/O usage
- analysis here is likely to show larger differences between rotating HDDs and SSDs, especially when a process produces lots of random reads/writes
- **modprobe** is likely to be the most I/O intensive process

unused services

unused services

- checking the output of bootchart2 you may notice services being started that you don't recognise and/or that are not used
- these may include:
  - default services installed during installation
  - unused services installed long ago

unused services

- if you don't recognise a service:
  - check its manpage / package details

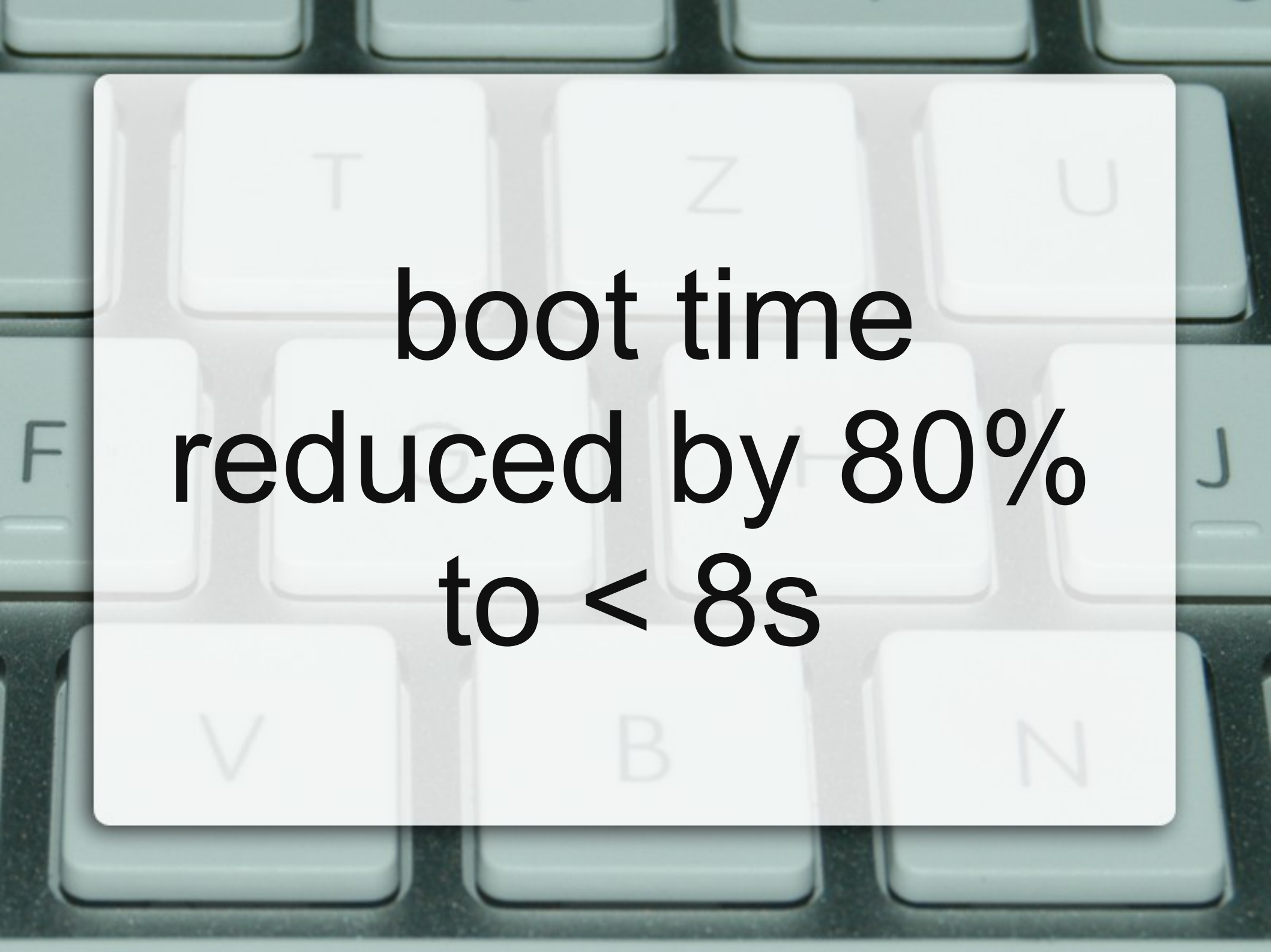
```
$ man <servicename>
$ dpkg -S /path/to/service
```

- If you decide it's definitely surplus to requirements, uninstall via your package manager

conclusions

conclusions

- opportunity to look under the hood at the boot process in detail
- very straightforward data collection
- clear visual output
- results of installing an SSD?


boot time
reduced by 80%
to < 8s

performance analysis

- timing shows grub -> display manager

	HDD	SSD
readahead (disabled)	39.18s	8.05s
readahead (enabled)	33.03s	**7.84s**

- most effective change was installing SSD
- readahead most effective on HDD

links

- bootchart2 / pybootchartgui

  https://github.com/mmeeks/bootchart
- readahead

  https://fedorahosted.org/readahead/
- e4rat

  http://e4rat.sourceforge.net/