Improving application responsiveness and I/O latency with the BFQ I/O scheduler

Paolo Valente

Department of Physics, Computer Science and Mathematics
Modena - Italy

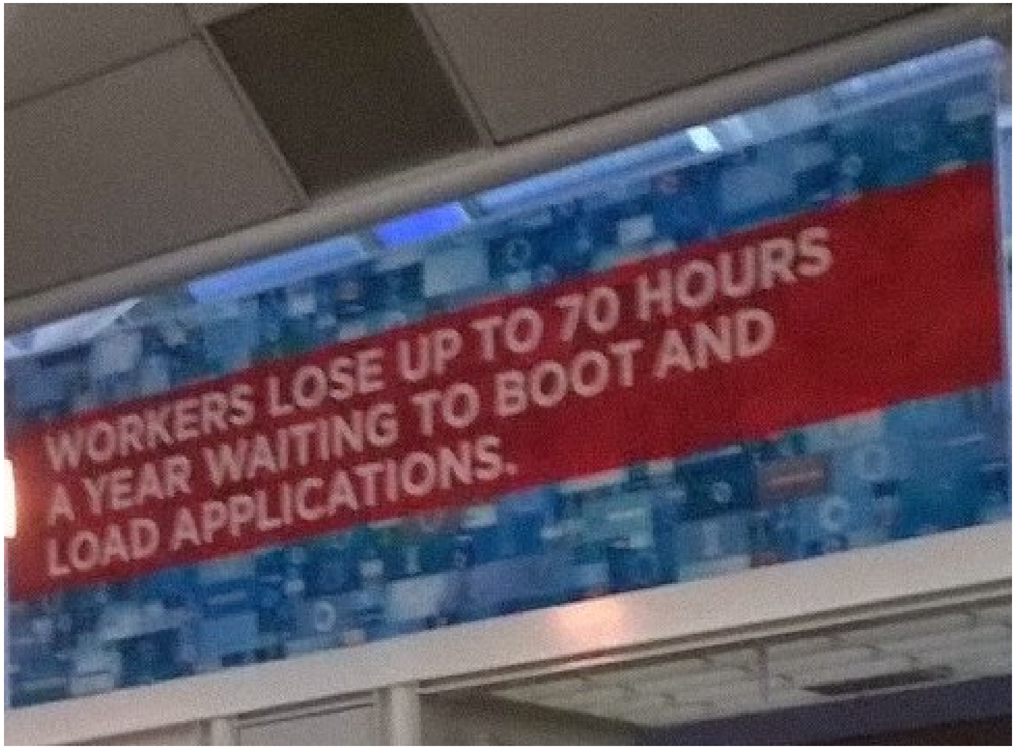
UNIMORE
UNIVERSITÀ DEGLI STUDI DI
MODENA E REGGIO EMILIA

P-SOCRATES

LinuxDay Pisa 2014

Budget Fair Queueing (BFQ) 1/2

- Storage-I/O scheduler
  - High responsiveness
  - Low latency for soft real-time (time-sensitive) applications, such as multimedia ones
  - High throughput
  - Desired throughput fraction guaranteed to each application
    - even if throughput fluctuates



Budget Fair Queueing (BFQ) 2/2

- Adopted in a number of distributions and kernel variants
- Submitted to *lkml* about four months ago: https://lkml.org/lkml/2014/5/27/314
  - Arranged a roadmap for possible inclusion
  - By replacing CFQ
- BFQ homepage: http://algogroup.unimore.it/people/paolo/disk_sched/

Contents of this presentation

1. Two demos of the performance of BFQ
    - Compared with CFQ, DEADLINE and NOOP
    - On an SSD and on an HDD
2. Some considerations about BFQ, fast devices and latency

Demo

- Links to the videos of the demos in BFQ homepage

  http://algogroup.unimore.it/people/paolo/disk_sched/

The trick ...

- Applications are launched quickly, and interactive and soft real-time applications enjoy a low latency because
  - BFQ privileges the I/O related to interactive or soft real-time tasks
- Hard part
  - Not losing throughput
  - Correctly detecting applications to privilege
  - Implementing all the logic cleanly

Speed and latency

- Because of its execution time, the current version of BFQ is likely to be a bottleneck on high-end, high-speed devices
  - “But little or no scheduling is needed on such devices”
    - “In fact, as the speed increases, latency problems will just go away”
  - True?

Speed and latency

- Not that sure
  - Device-related issues
  - Workload-related issues

Device-related key problem

- Devices usually **reorder** I/O requests
- Even if a device is very fast, but it
  - systematically serves many wrong requests
  - before serving the right ones,

  then responsiveness and latency for soft real-time applications are likely to be still bad
  - Exactly the cause of the problems shown in the demo
- Ordering might be controlled by passing in priorities
  - But this would hurt performance

Relation with new devices

- Speed will increase
- But expectedly through higher parallelism
  - Devices will be fed with more requests
  - Internal device schedulers may then happen to serve more wrong requests before the right ones
  - The *wrong-service-order* problem may remain unaltered
    - Or even get worse

Workload-related issues 1/2

- High-speed, costly devices make sense where high throughput is needed
- For example, where many instances of the same application need to be executed in parallel
  - Virtual machines in clouds
  - Instances of streaming servers in Video-on-Demand services

Workload-related issues 2/2

- For these applications
  - If the available throughput becomes *N* times as high as before
    - Also the number of instances that can be executed becomes *N* times as high
    - The per-instance throughput, and hence the request-completion latencies would then be about the same as before
  - In the end, latency issues are likely to remain about the same as before
    - Or may become even worse, because there would be more outstanding I/O requests

Future work on BFQ

- Dealing with millions of IOPS
    - Measuring the impact of BFQ
    - Investigating simpler variants of BFQ
        - Useful also if one may want to use BFQ as an internal scheduler in a device
            - This could enable low-latency guarantees to be provided with no or a negligible throughput penalty
- Guaranteeing high responsiveness and low latency also in virtualized environments

The end

Thanks for your attention

Questions?