A solution to high latencies caused by I/O

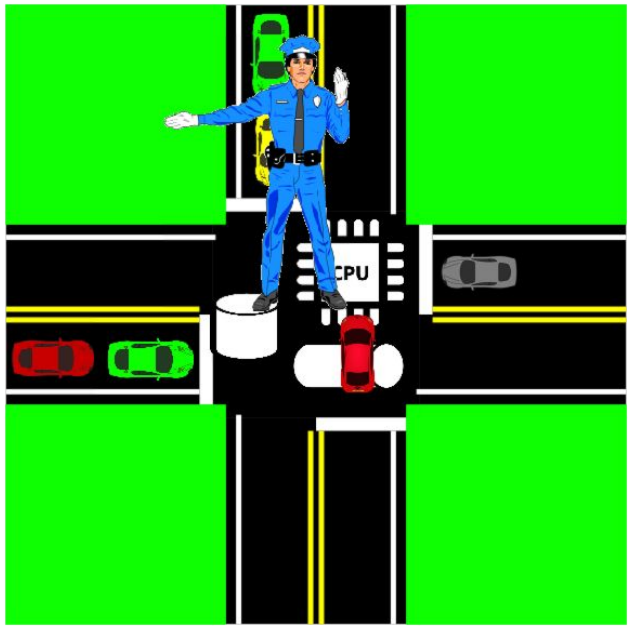
Paolo Valente, Assistant professor, *Linaro*

Content

- What's the job of an I/O scheduler?
- High latencies caused by I/O, and the *bfq* solution
  - Hikey (LeMaker) board: Android and Debian
  - Google Pixel 2
  - Pogoplug v4 and a laptop, with 4-8GB SD Cards
  - Tests in progress with other boards
- Questions
  - We don't bump into these issues with our devices and services, don't we?
  - Why does only BFQ work?

I/O scheduler

- Decides the order in which I/O requests are to be served
- So as to guarantee:
  - High I/O throughput
  - Low latency
    - High responsiveness - Low lag
  - Fairness
  - ...
- Component in the block layer (legacy blk, blk-mq)
  - blk: *noop, deadline, cfq, (bfq)* blk-mq: *none, mq-deadline, kyber, bfq*



The two worlds

- Main classes of Linux-based OSes for embedded systems
  - Android — No in-tree bfq, no blk-mq support in mmc
  - Linux distributions — From 4.16: bfq, blk-mq support in mmc
- For Android
  - HiKey (LeMaker) Board
  - Google Pixel 2
- For Linux distributions
  - HiKey (LeMaker) Board
  - Pogoplug v4
  - Dell Laptop with SD-Cards from 256MB to 8GB

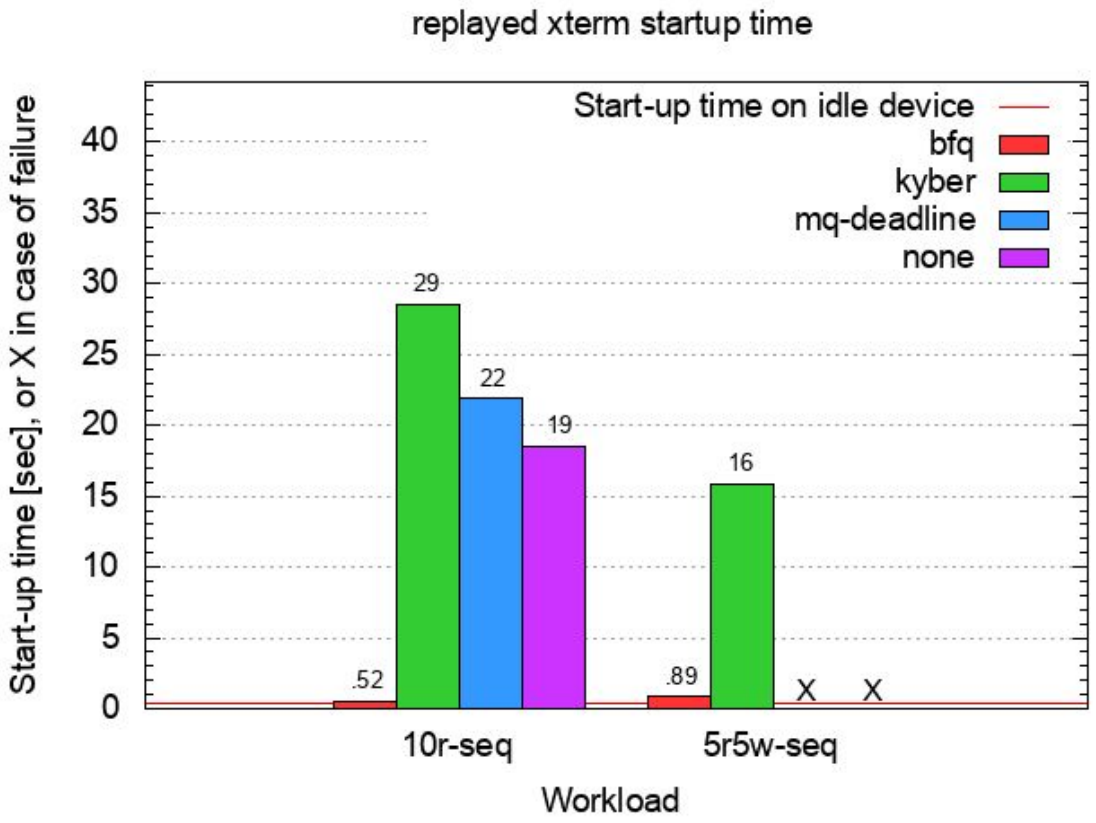
Android

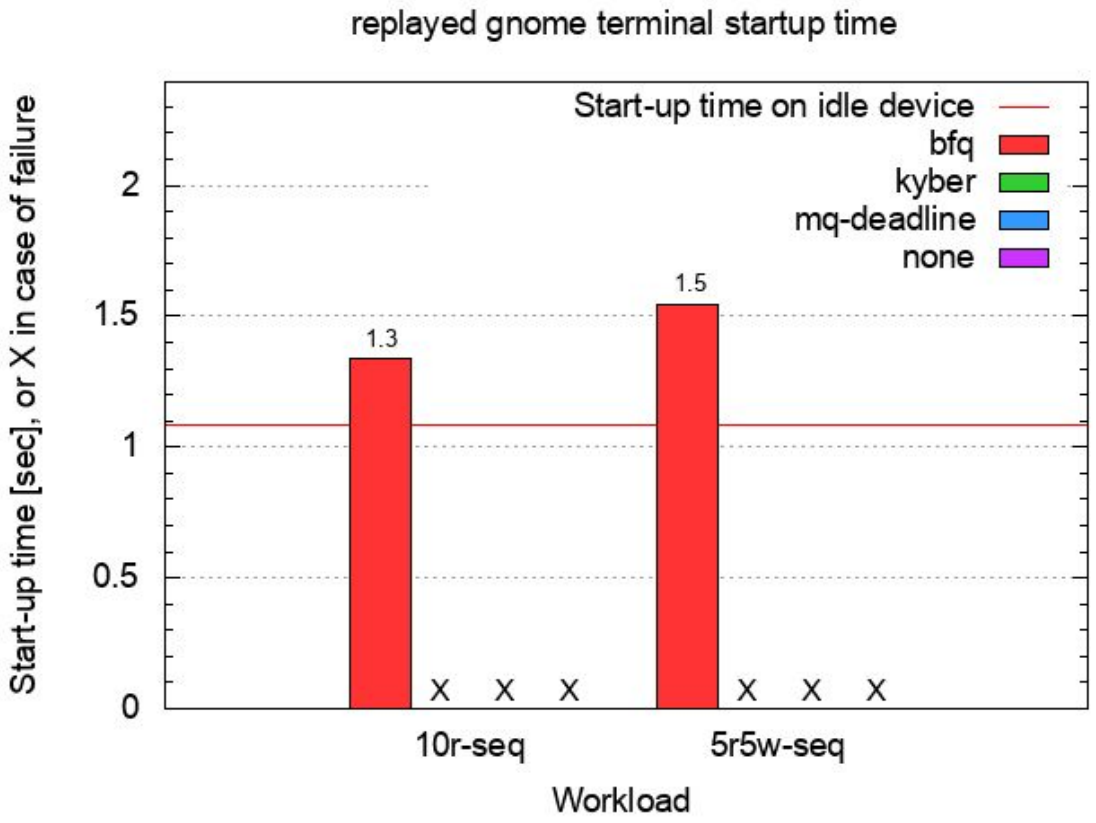
- HiKey (LeMaker) board, with *out-of-tree bfq*
  - Demo showing problems and a solution
    https://youtu.be/ANfqNiJVoVE
- Google Pixel 2
  - Hard to find fast networks in Italy, for a real test with updates
  - Resorted to testing with just file copies in the background
    - Lighter, but intense I/O in common with updates
    - High lag with file copy ⇒ high lag with app updates on a fast network
  - Demo:
    https://youtu.be/Ai3EPDpdsvY

Linux distros: Hikey with Debian

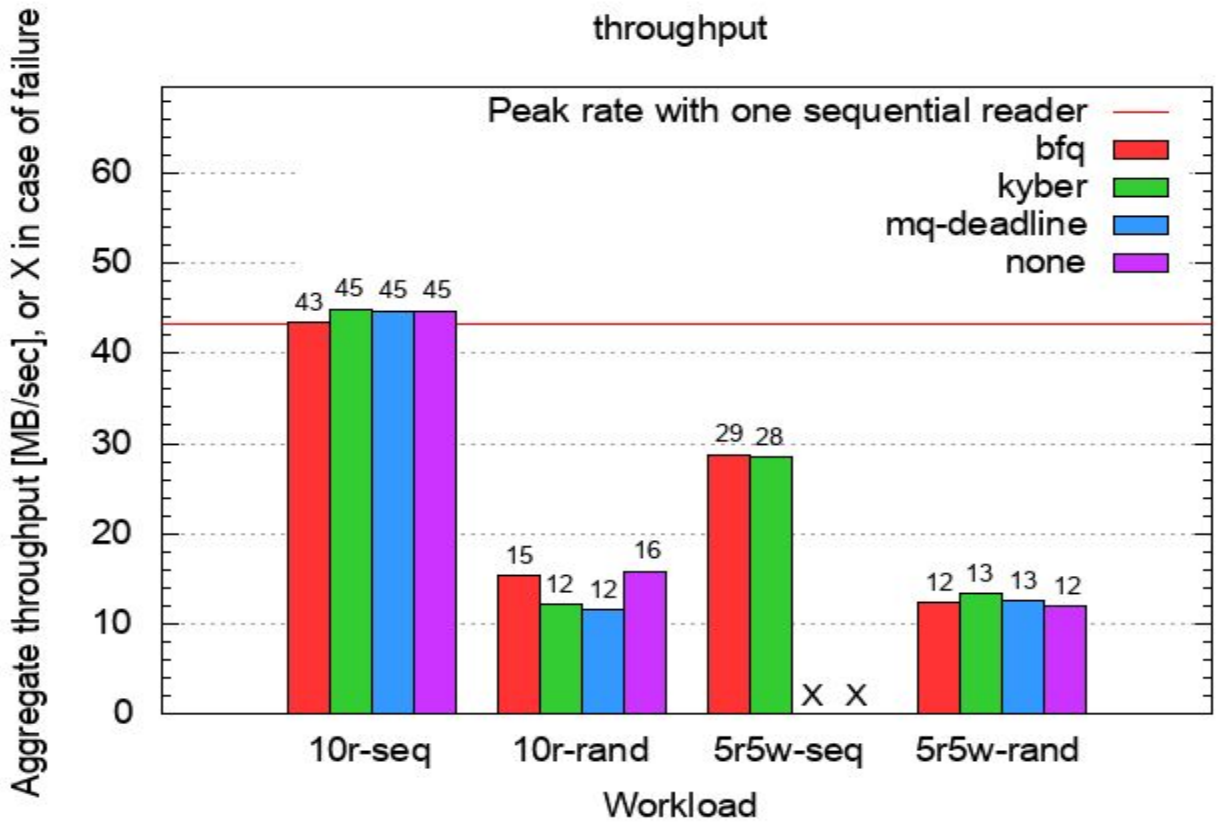
- Tests performed with the *S benchmark suite*: https://github.com/Algodev-github/S
- Demo: https://youtu.be/gyM_JJtIvP0
- Then more thorough results through graphs
  - Start-up times for *xterm*, *gnome-terminal* and *LibreOfficeWriter*
    - *LibreOfficeWriter* not shown: same results as with *gnome-terminal*
  - I/O throughput

Hikey with Debian: start-up times





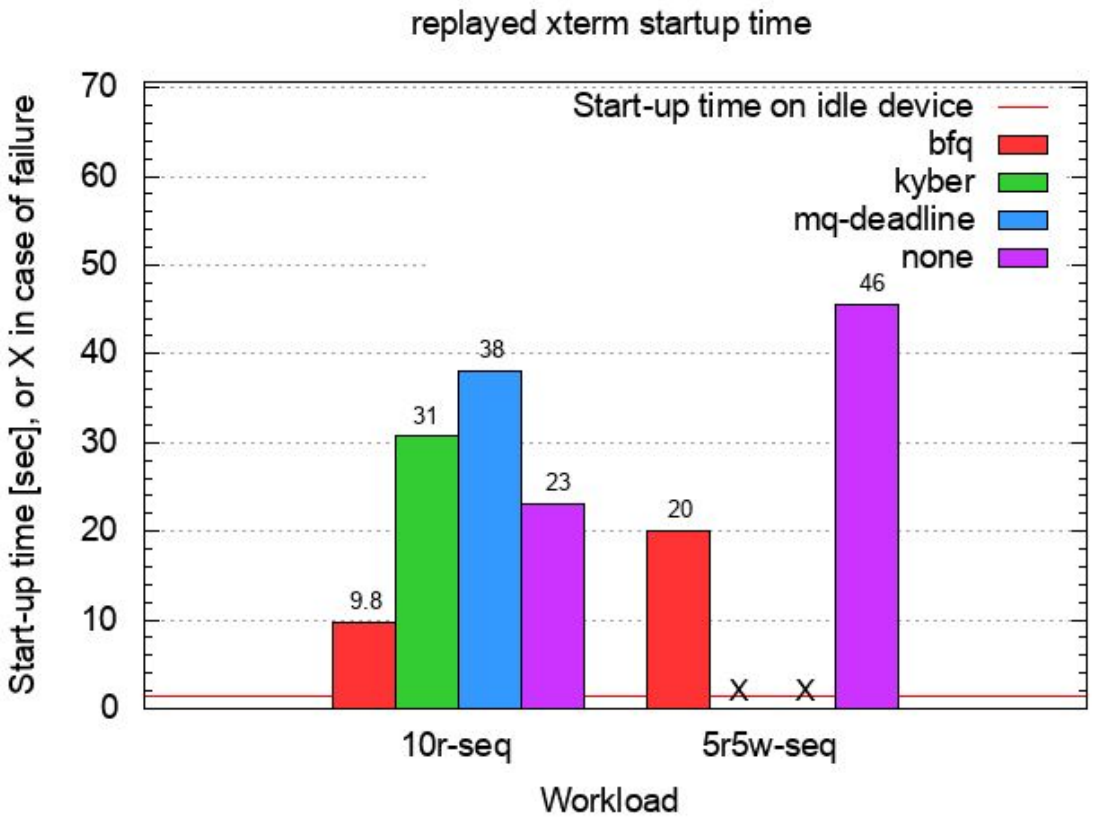
Hikey with Debian: throughput



Pogoplug v4 with Arch Linux

- Tests run by Linus Walleij from Linaro
  - With an 8GB *Kingston SDHC Card*
- Interesting case: *bfq* was not that good either
- Showing results only for *xterm* start up
  - Other test cases are very little informative

Pogoplug v4 with Arch Linux: start-up times



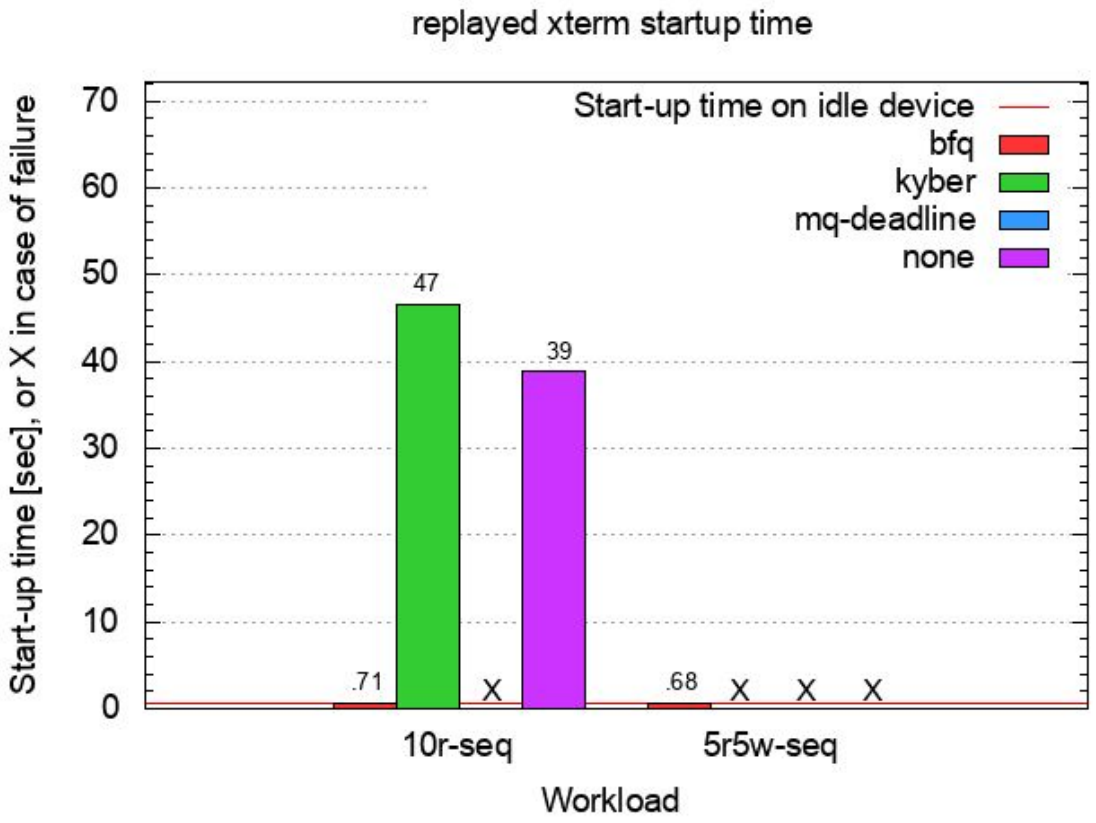
Pogoplug v4 with Arch Linux: comments

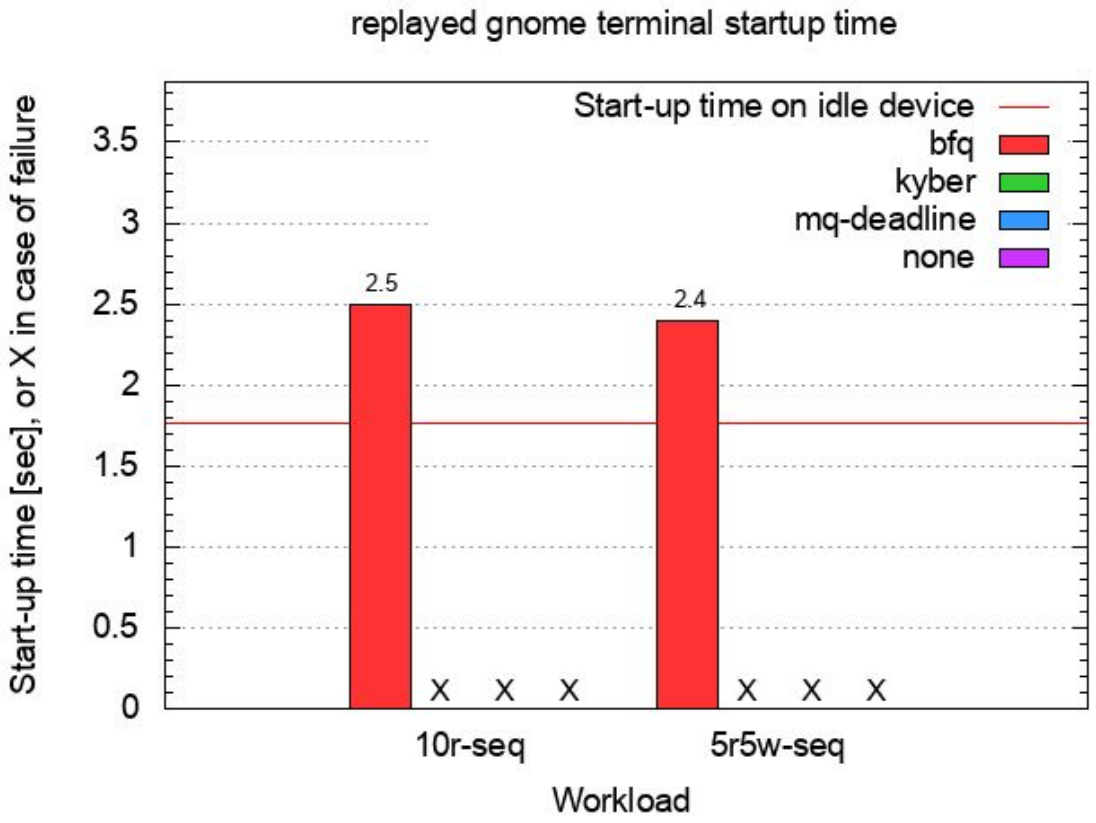
- Doubt
  - Some other hardware resource influenced results
- To clear this doubt: test with the same SD-Card, but
  - mounted this time on the SDHCI host controller of a Dell Laptop

Dell laptop with Fedora

- Tests run by Linus Walleij from Linaro
- Three SD-Cards tested, with equivalent results
  - 256MB Lexar SD Card
  - 8GB Kingston SDHC Card
  - 4GB Toshiba SDHC Card
- Relative performance of all schedulers but *bfq* is much worse than that with the HiKey
- Thus, for brevity, reporting
  - Only *xterm* and *gnome-terminal* start-up times
  - Only for the 8GB Kingston SDHC Card

Dell laptop with Fedora: start-up times





Second part: two questions

- We don’t bump into these latency issues with devices and services, don’t we?
  - And when we really don’t see these issues, why?
- Why does only *bfq* work?

A matter of probability: the bright side

- Given the terrible results shown, why are our devices perfectly responsive in normal usage?
  - At least most times
- Because storage is almost always **underutilized**
- We may run into troubles, if, in parallel,
  - one or more large files are being read or written,
  - a tree of source files is being compiled,
  - a software update is in progress,
  - indexing daemons are scanning/updating
  - ...

A matter of probability: the downside

- If your use case is beyond average, then the behavior of the system usually changes dramatically
  - It may become very frustrating and at times impossible to use
  - No practical solution, apart from bfq, where available and known
- Users often just think this is normal and somewhat inevitable
- But it is not
  - Some companies have turned these problems into an opportunity
  - Proposing devices that guarantee an (allegedly ?) much better user experience, with no or very rare lag issues

Responsiveness of services

- Hard to guarantee acceptable latency with a system that may be subject to intense I/O
- Typical solution: **overprovisioning**
  - Guarantee that the system is always underutilized
  - Evident high cost in terms of resources, energy and money
- Or/plus some ad-hoc (non-public) scheduling/tuning
  - Typically very rigid
  - Fails if the workload changes w.r.t. to that for which the system has been fine-tuned

How does *bfq* make it?

- Through a combination of three main techniques
  - *Accurate proportional-share of the throughput*
    - Shares proportional to weights (assigned to single processes or groups)
  - *Detection of I/O to privilege*
    - Raise the weight of the processes to privilege
  - *I/O-dispatch plugging* (a.k.a. device idling)
    - Do not dispatch other I/O while the process in service has momentarily no pending I/O, if the process does sync I/O
    - Prevents low-weight I/O from cutting in front of high-weight sync I/O inside the storage device
    - This may lower throughput. Challenge: plug for the minimum time needed


Embedded Linux
Conference
North America
OpenIoTSummit
North America
THE
LINUX
FOUNDATION