Improving Performance using the LINUX IO Scheduler

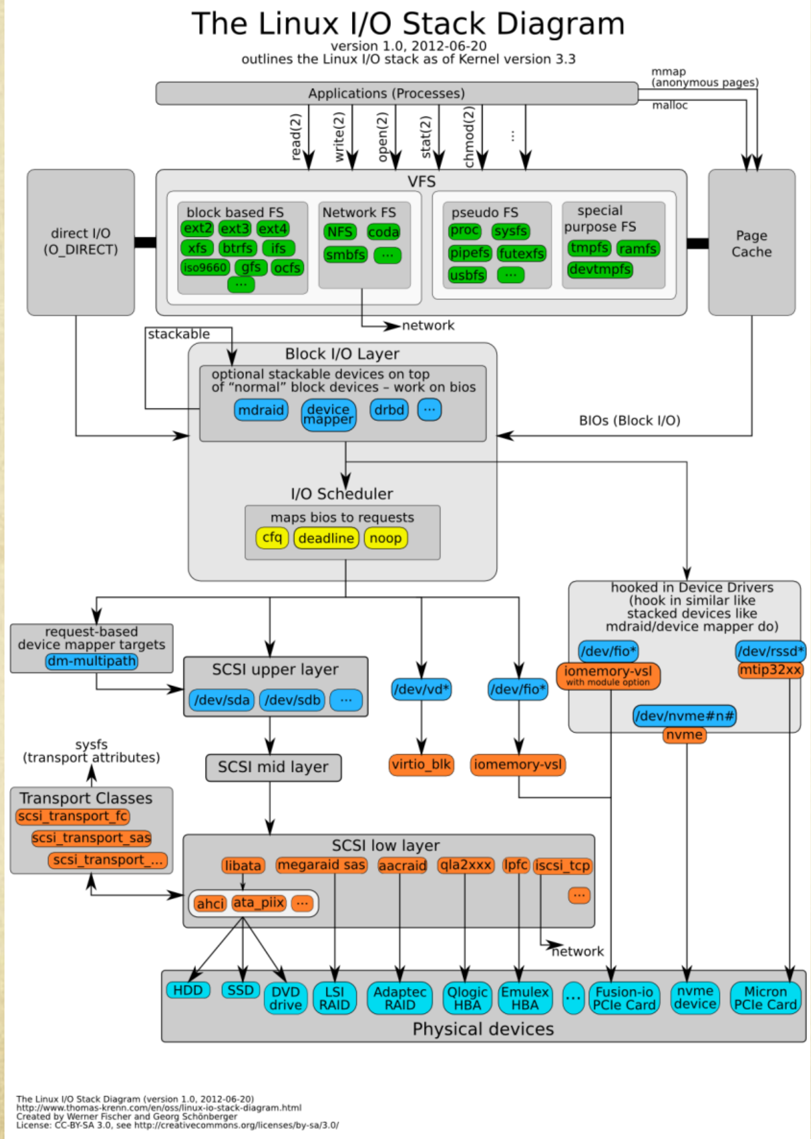
Shaun de Witt
STFC

ISGC2016

Role of the Scheduler

- Optimise Access to Storage
  - CPU operations have a few processor cycles (each cycle is < 1ns)
  - Seek operations take about ~> 8ms (25 million times longer)
- Reorder IO requests to minimise seek time
- Two main operations – *sorting* and *merging*
  - Sorting arranges block reads sequentially
  - Merging makes two adjacent block reads into one

Where the Scheduler Sits



The Linux I/O Stack Diagram (version 1.0, 2012-06-20)
http://www.thomas-krenn.com/en/oss/linux-io-stack-diagram.html
Created by Werner Fischer and Georg Schönberger

© Werner Fischer & Georg Schonberger
https://www.thomas-krenn.com/en/wiki/Linux_Storage_Stack_Diagram

Available Schedulers

- Linux Kernel >= 2.4 has 4 schedulers
  - CFQ (default)
  - Deadline
  - Anticipatory
  - Noop

Completely Fair Queueing

- Each IO device gets its own queue and each queue gets its own timeslice
- Scheduler reads each queue in round-robin until end of timeslice
- Reads prioritised over writes

Deadline

- Three queues
  - Elevator queue containing all sorted read requests
  - Read-FIFO queue containing time ordered read requests
  - Write-FIFO queue containing time ordered write requests
- Each request in FIFO queue has an ‘expiration time’ of 500 ms
- Normally requests are served from the elevator queue, but as requests expire in the FIFO queue, these queues will take precedence
- Gives good avaerage latency but can reduce global throughput

Anticipatory

- Similar to deadline scheduler
  - But with some anticipatory knowledge – expectation of a subsequent read
- Read requests serviced within deadline, but then pauses for 6ms, waiting for a subsequent read of the next block
- In general, this works well if most reads are dependent (e.g. reading sequential blocks in a streaming operation)
  - But may be poor for vector reads

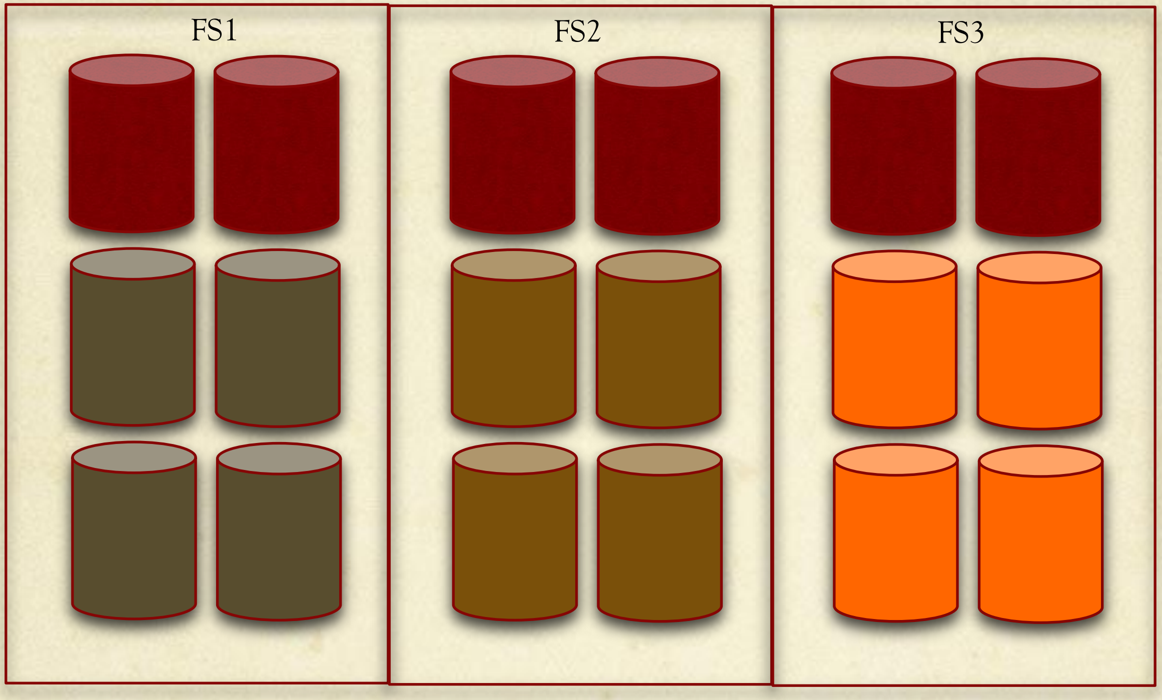
NOOP

- Simplest scheduler
- Only does basic merging
    - NO SORTING
- Recommended for non-seeking devices (e.g. SSDs)

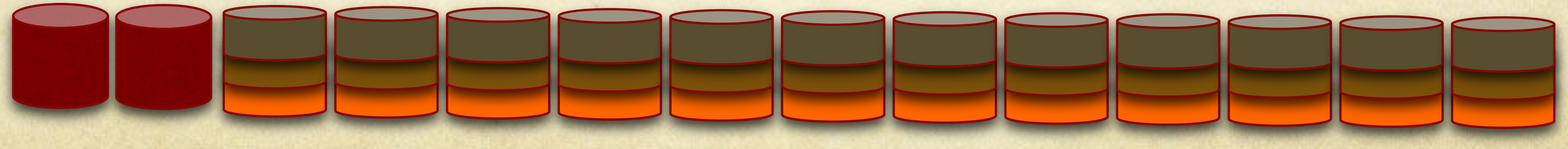
Hardware Set-up at RAL

- Diskservers configured in RAID6 or RAID60 with 3 partitions
  - Allows for two independent disk failures without losing data
- Uses **hardware RAID controllers**
  - Various vendors dependent on procurement
- Unusual Horizontal striping
  - Minimise RAID storage overhead

RAID Configuration



‘Typical’ RAID6
Configuration



RAL RAID6 Configuration

Impact of Using Hardware RAID

- RAID controller has fairly big cache
  - Able to write quickly for large periods of time
- But now CFQ sees only one device (or one for each filesystem)
- Scheduler sorts iops assuming a well defined disk layout
  - But RAID controller has a different layout and will resort it
- Not normally a significant overhead, until the iops become large and effectively random
  - Double sorting and different cache sizes between kernel and RAID controller lead to massive inefficiencies
  - Almost all application time spent in WIO

What made RAL look at this

© Alamy

CMS Problems

- Accessing disk-only storage under LHC Run 2 Load
- All disk servers (~20) running at 100% WIO
  - Only certain workflows (pileup)
  - Job efficiencies running <30%
  - Other Tier 1 sites also had problems, but were much better (~60% CPU efficiency)
  - 'Normal' CPU efficiencies run >95%

Solutions...

- Limit number of transfers to each disk server
  - Jobs timeout waiting for storage system scheduling
  - Means few jobs run well, but most fail
- Limit number of pileup jobs on the processing farm
  - No reliable mechanism for identifying them
- Add more disk servers
  - Would work, but would have cost implications

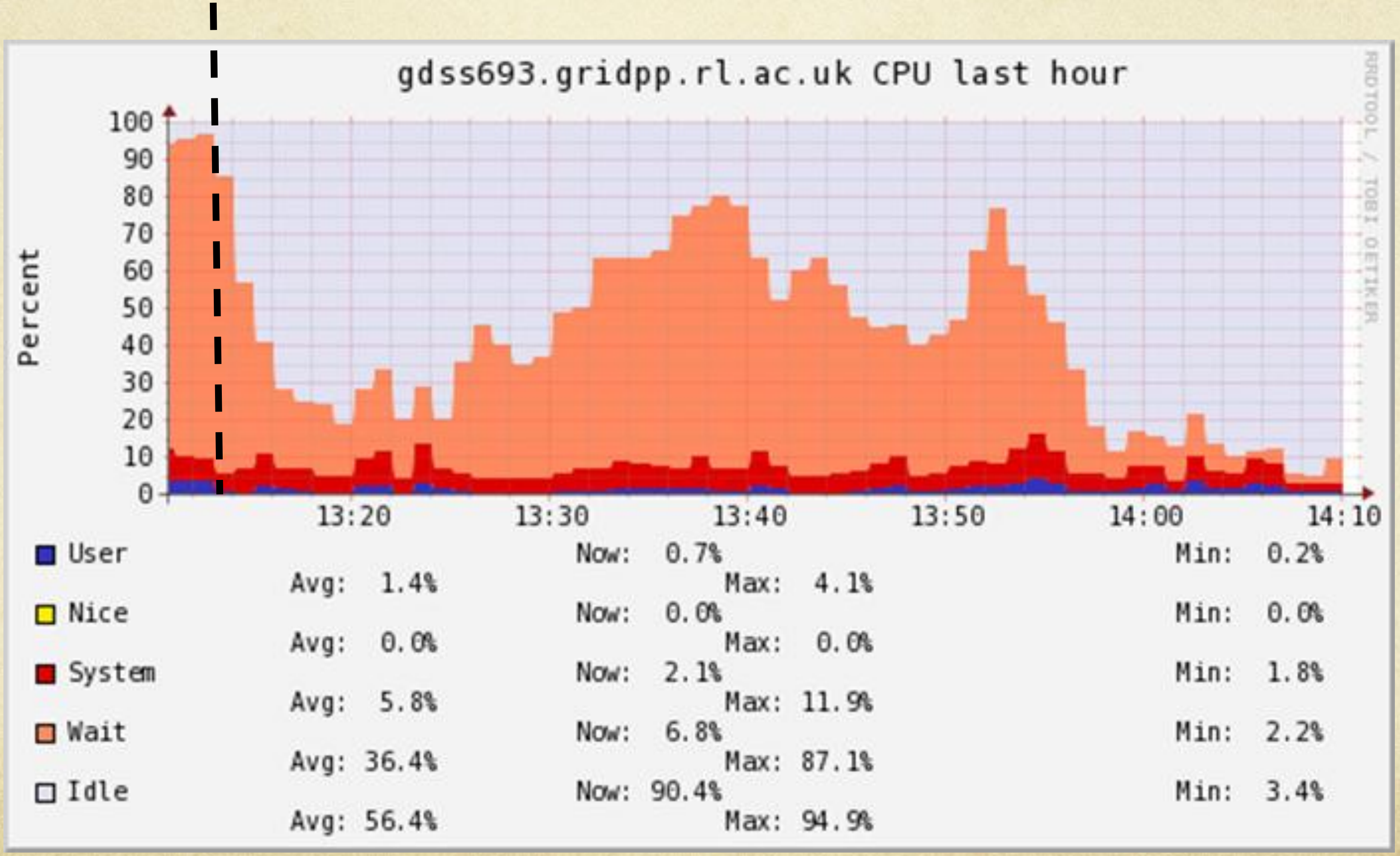
Finally…

- Try switching scheduler…
  - Can be done and reverted ‘on the fly’
    - Does not require a reboot
  - Can be applied on a single server to investigate the impact
- Preemptive CAVEAT
  - Don’t try this at home!
  - Software RAID and different RAID controllers will behave differently
  - Only do this if either
    - Your storage system is already ‘overloaded’
    - You have tested the impact with your hardware configuration

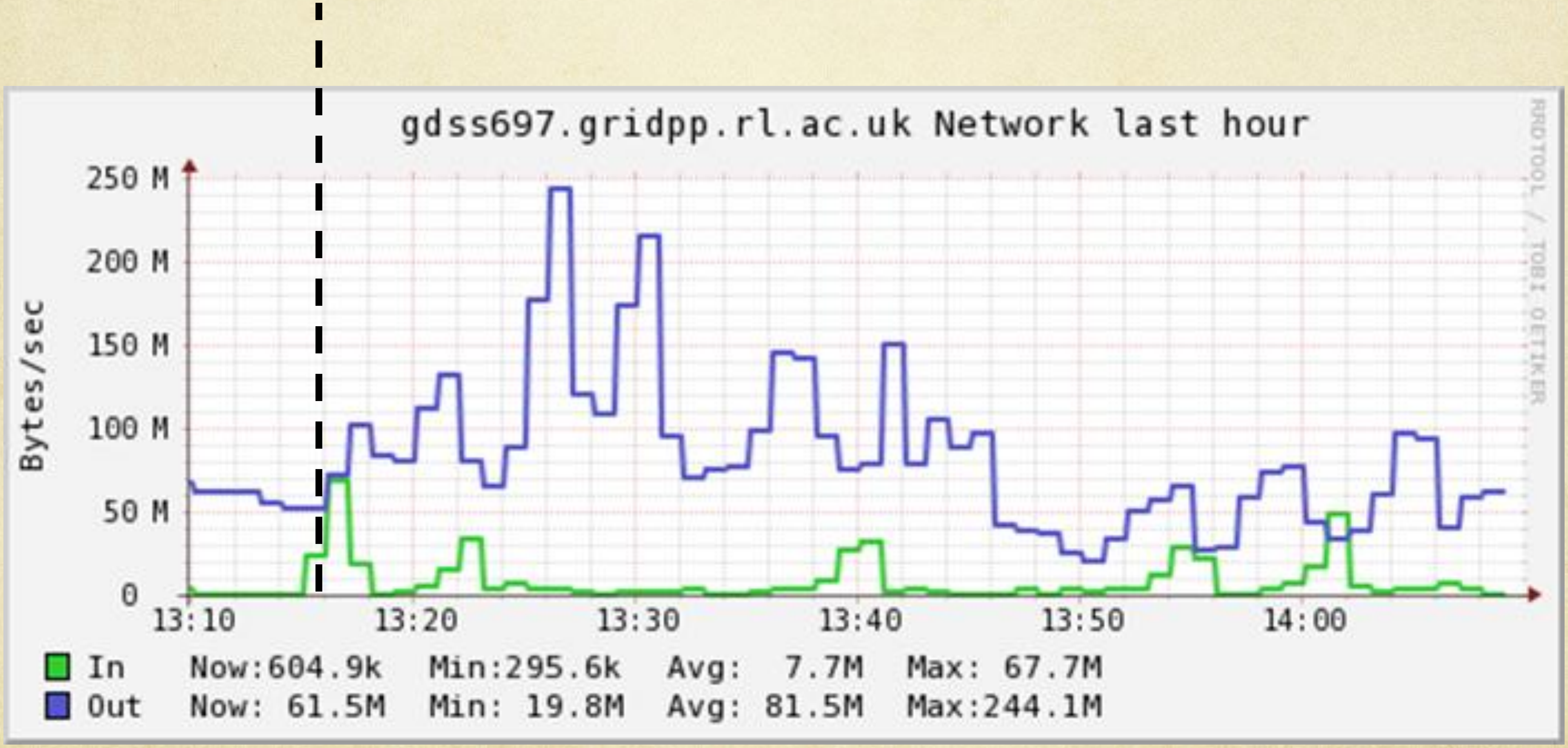
Anecdotal Impact

- Switched to NOOP scheduler
  - WIO reduced from 100% to 65%
  - Throughput increased from 50Mb/s to 200Mb/s
- Based on making a change to 1 server under production workload

Anecdotal Impact



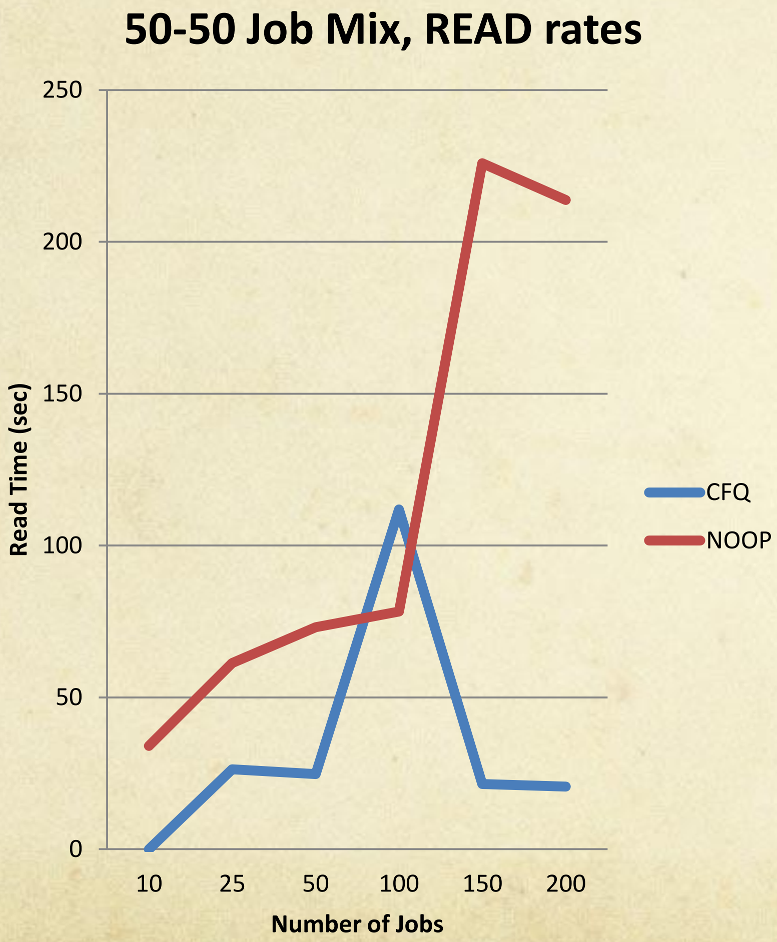
Anecdotal Impact

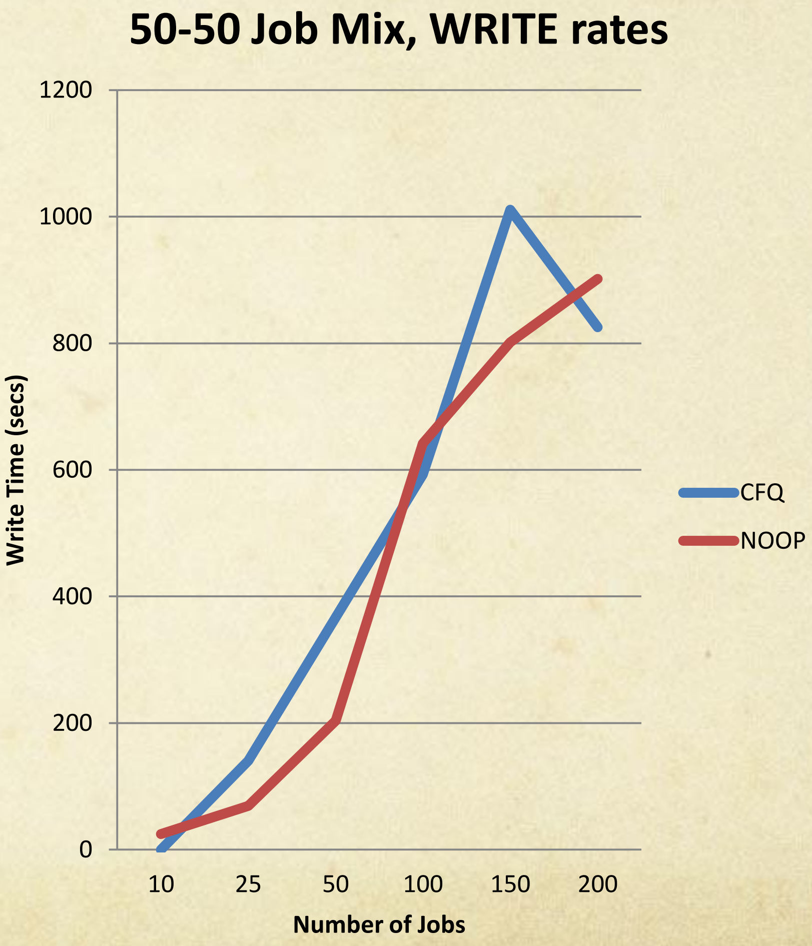


Attempt to Quantify

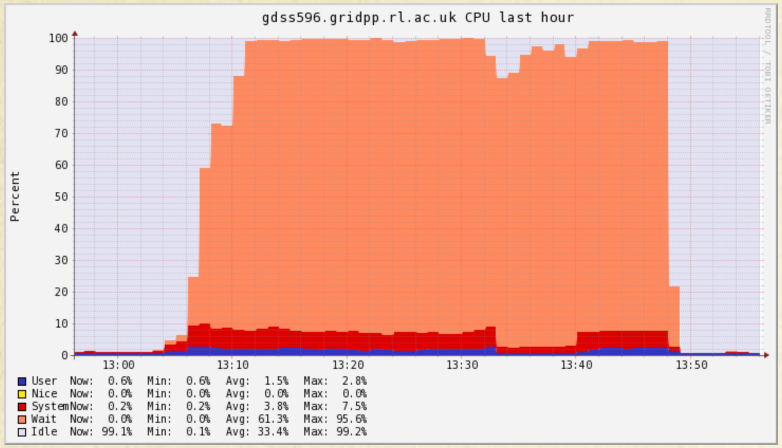
- Heavy WIO reproduced by
  - Multiple jobs reading/writing whole files (1GB random data)
  - Different R/W mixing
  - Using xrootd protocol
- 200 different files for reading (try to remove any effect of caching reads)

50-50 Job Mix

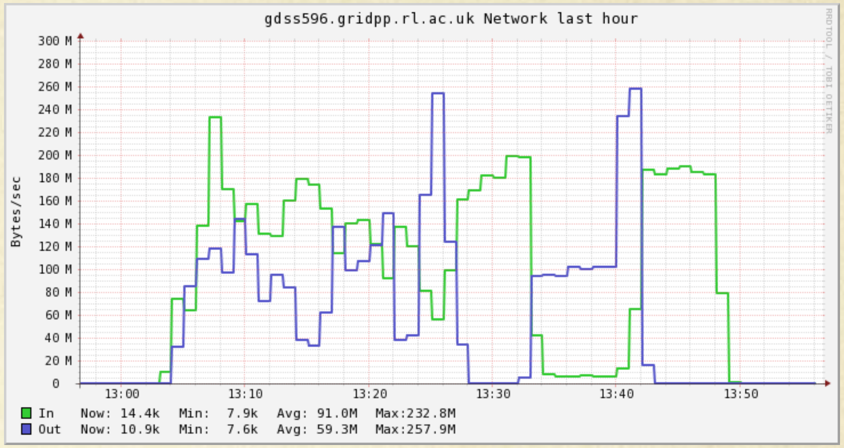

50-50 Job Mix, READ rates
Read Time (sec)
250
200
150
100
50
0
10
25
50
100
150
200
Number of Jobs
CFQ
NOOP

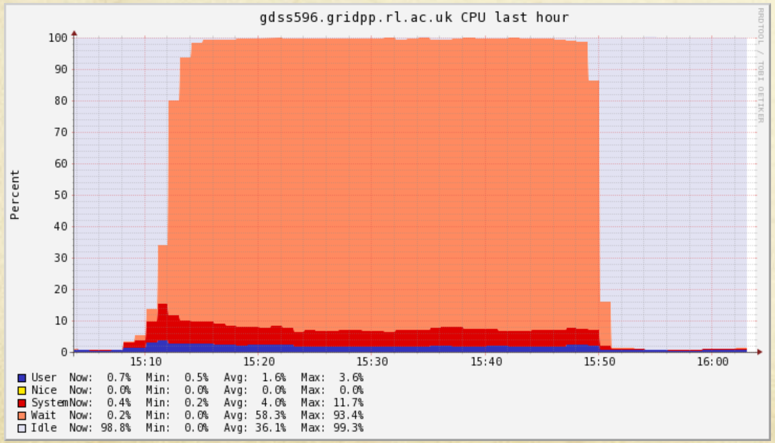

50-50 Job Mix, WRITE rates
Write Time (secs)
1200
1000
800
600
400
200
0
10
25
50
100
150
200
Number of Jobs
CFQ
NOOP

50-50 Job Mix (100 jobs)

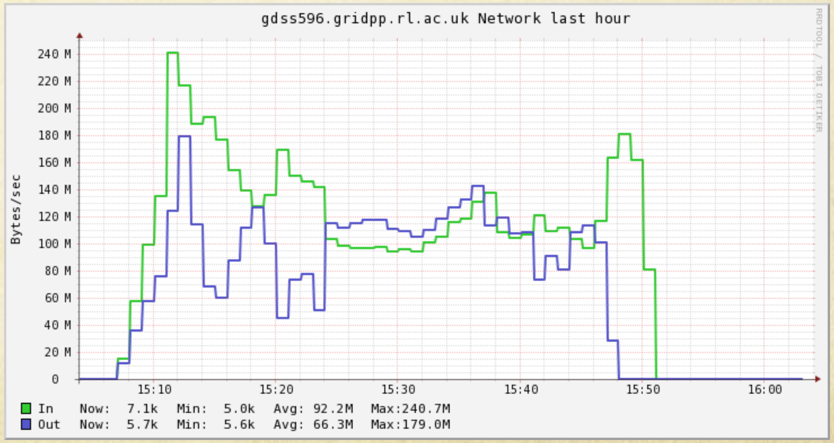


CFQ

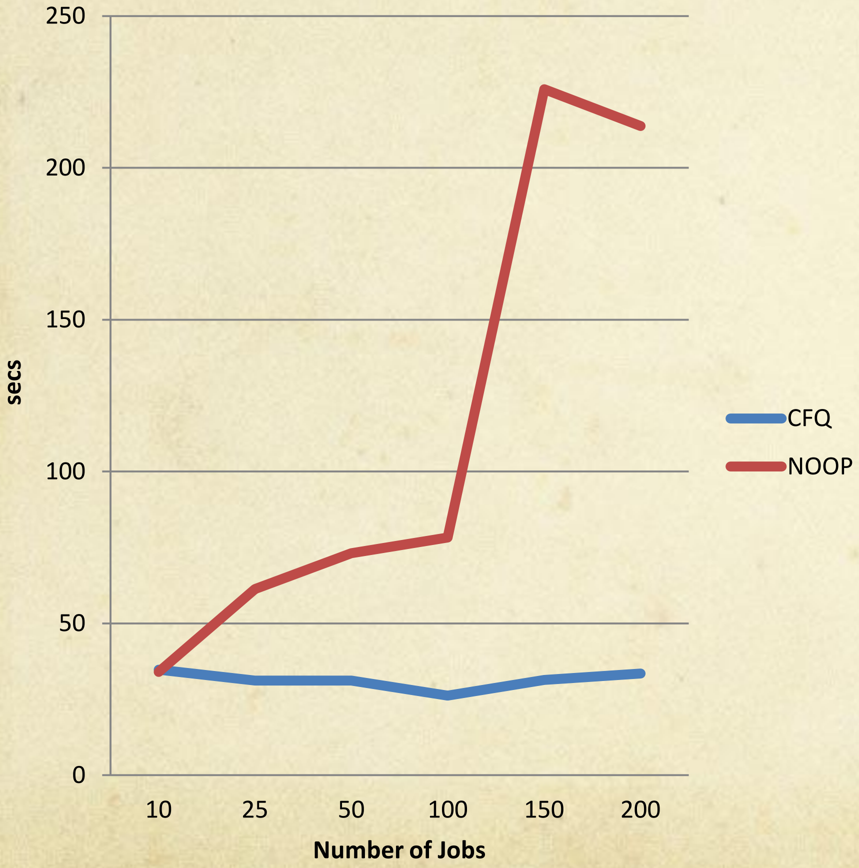




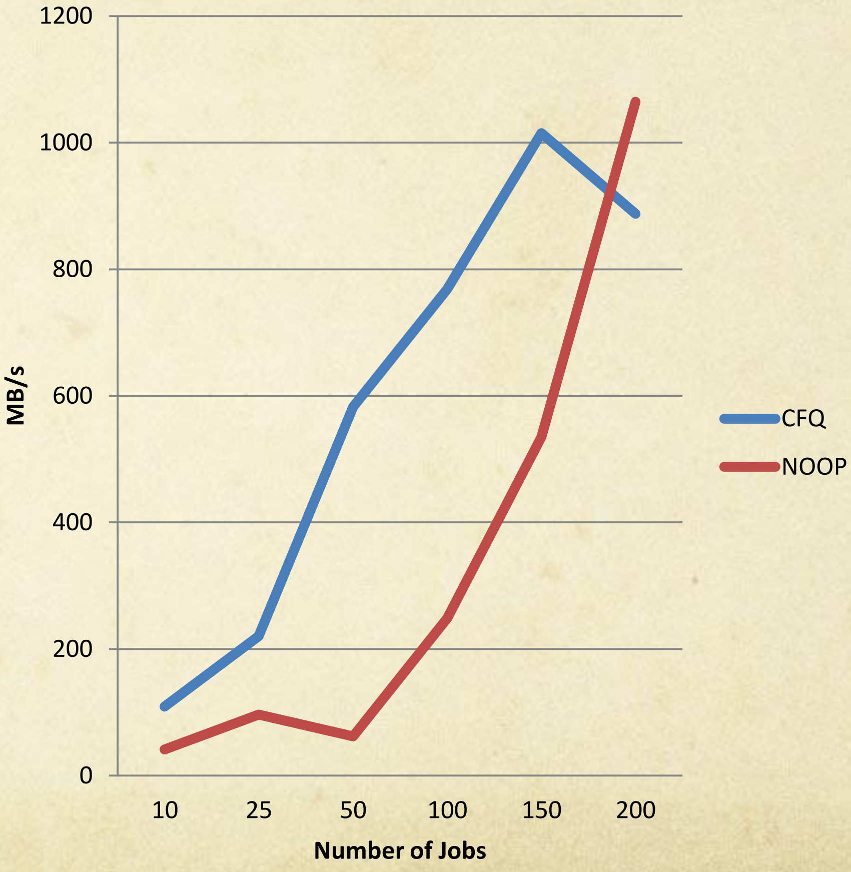
NOOP



Read Dominated (80-20) Job Mix

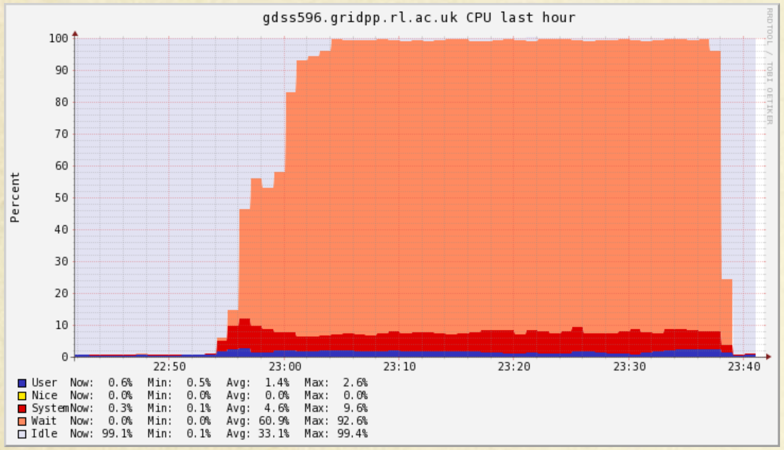


80-20 Job Mix, READ rates



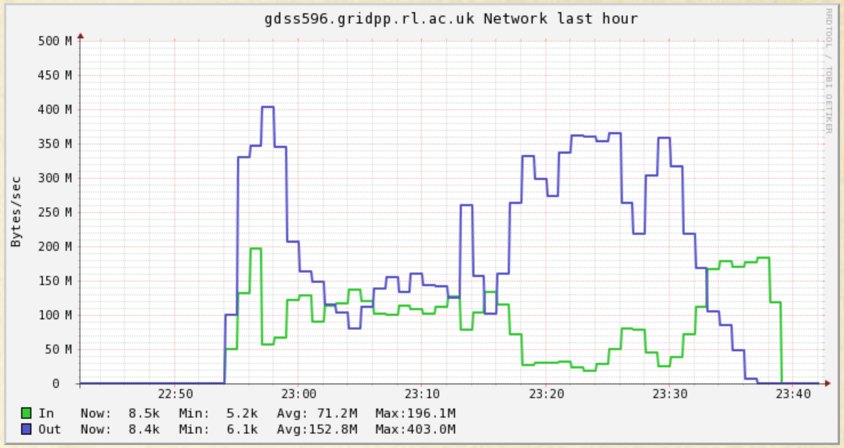
80-20 Job Mix, WRITE rates

Read Dominated Job Mix

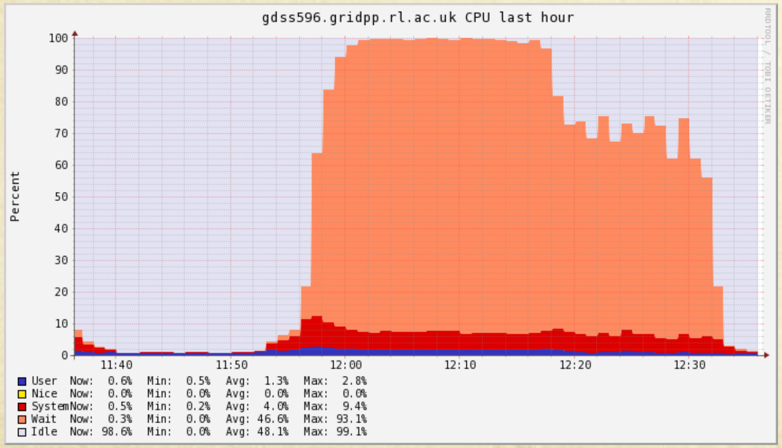


```
■ User   Now:   0.6%   Min:   0.5%   Avg:   1.4%   Max:   2.6%
■ Nice   Now:   0.0%   Min:   0.0%   Avg:   0.0%   Max:   0.0%
■ SystemNow:   0.3%   Min:   0.1%   Avg:   4.6%   Max:   9.6%
■ Wait   Now:   0.0%   Min:   0.0%   Avg:  60.9%   Max:  92.6%
□ Idle   Now:  99.1%   Min:   0.1%   Avg:  33.1%   Max:  99.4%
```

CFQ

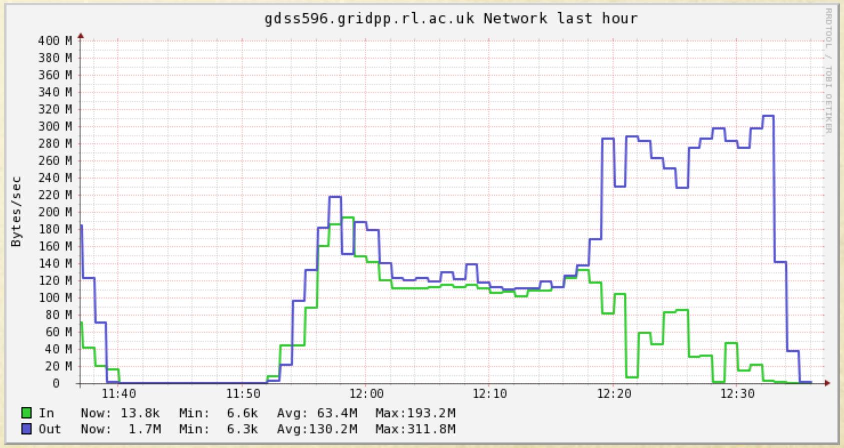


```
■ In    Now:   8.5k   Min:   5.2k   Avg: 71.2M   Max:196.1M
■ Out   Now:   8.4k   Min:   6.1k   Avg:152.8M   Max:403.0M
```



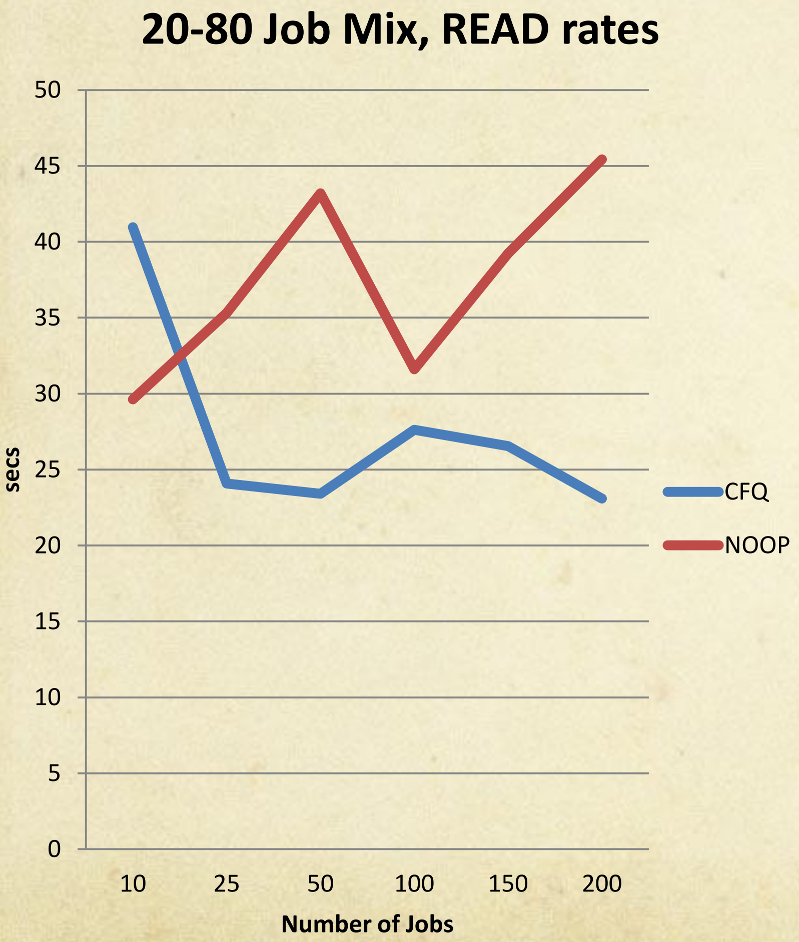
```
■ User   Now:   0.6%   Min:   0.5%   Avg:   1.3%   Max:   2.8%
■ Nice   Now:   0.0%   Min:   0.0%   Avg:   0.0%   Max:   0.0%
■ SystemNow:   0.5%   Min:   0.2%   Avg:   4.0%   Max:   9.4%
■ Wait   Now:   0.3%   Min:   0.0%   Avg:  46.6%   Max:  93.1%
□ Idle   Now:  98.6%   Min:   0.0%   Avg:  48.1%   Max:  99.1%
```

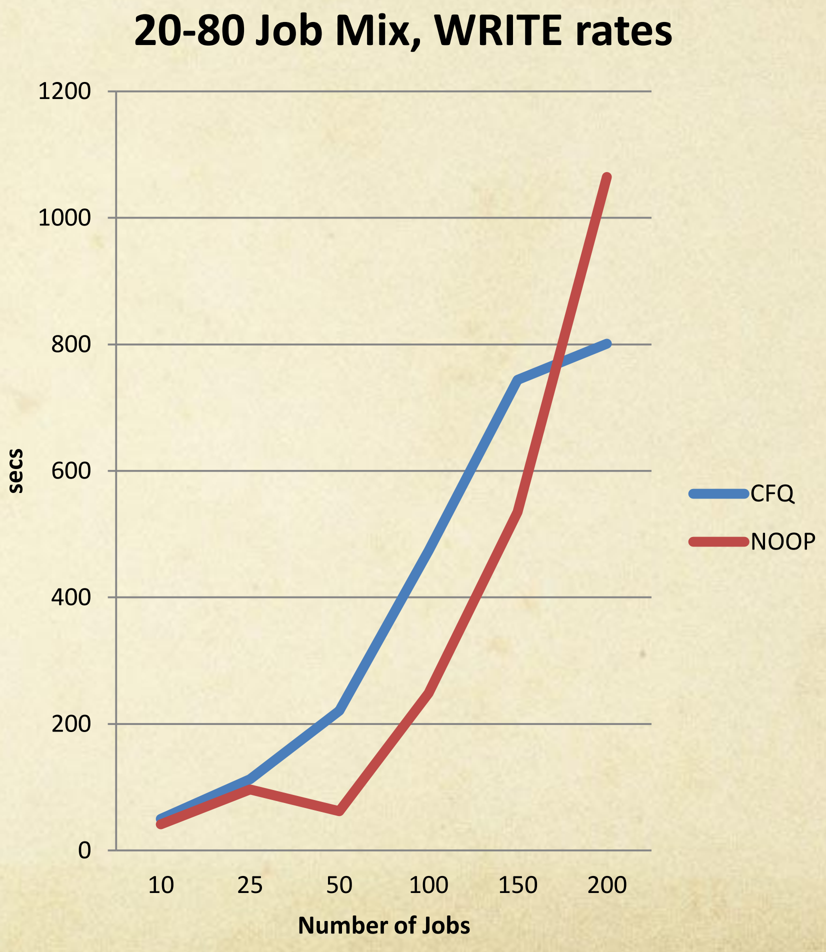
NOOP



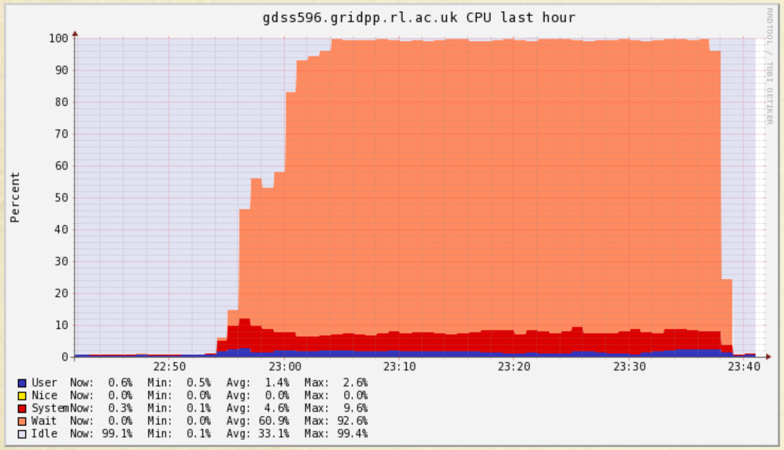
```
■ In    Now:  13.8k   Min:   6.6k   Avg: 63.4M   Max:193.2M
■ Out   Now:   1.7M   Min:   6.3k   Avg:130.2M   Max:311.8M
```

Write-Dominated (20-80) Job Mix

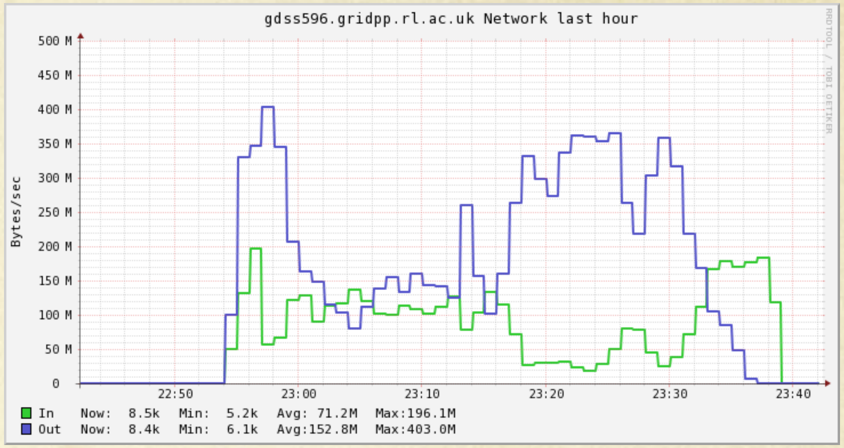


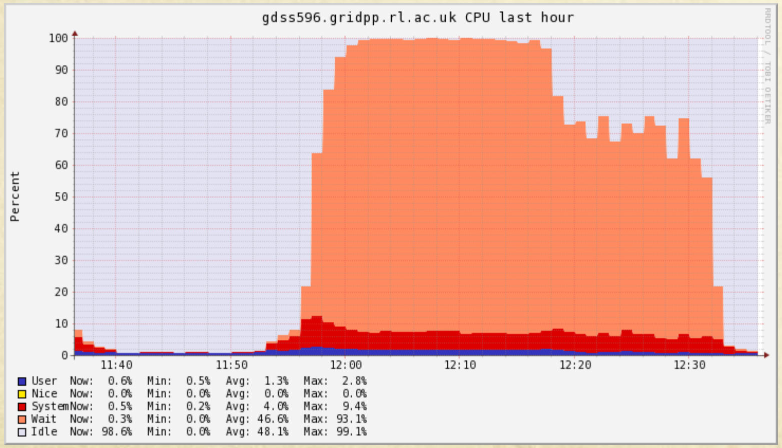


Read Dominated Job Mix

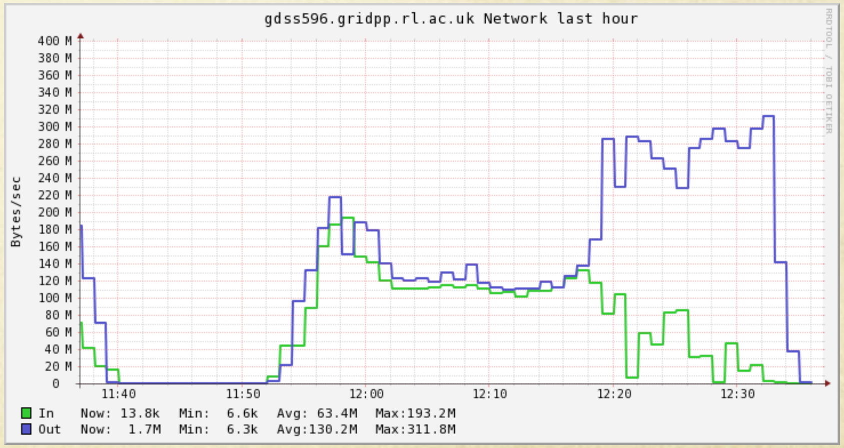


CFQ





NOOP



Summary & Conclusion

- Attempted controlled tests showed no significant benefit of using NOOP or CFQ schedulers
  - ATLAS problems bear this out
- … or to spin results positively
  - There is no detriment observed to using NOOP over CFQ
- Left with the question what is it about the pile-up jobs that causes such a bad impact on the disk servers, and why does NOOP give such a big gain
- Questions – Because I still have many!