REGIONALES RECHENZENTRUM ERLANGEN [RRZE]

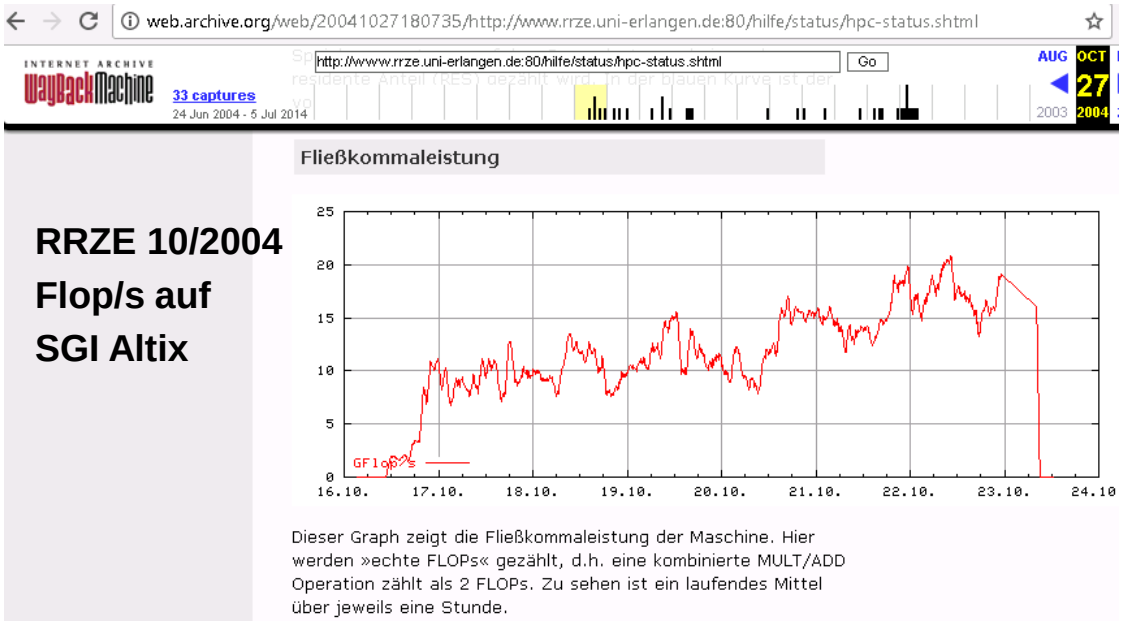
Demo RRZE ~~Ganglia~~

Proofs-of-concept for using RRDs from system monitoring for job-based views

Thomas Zeiser, RRZE
hpc@fau.de

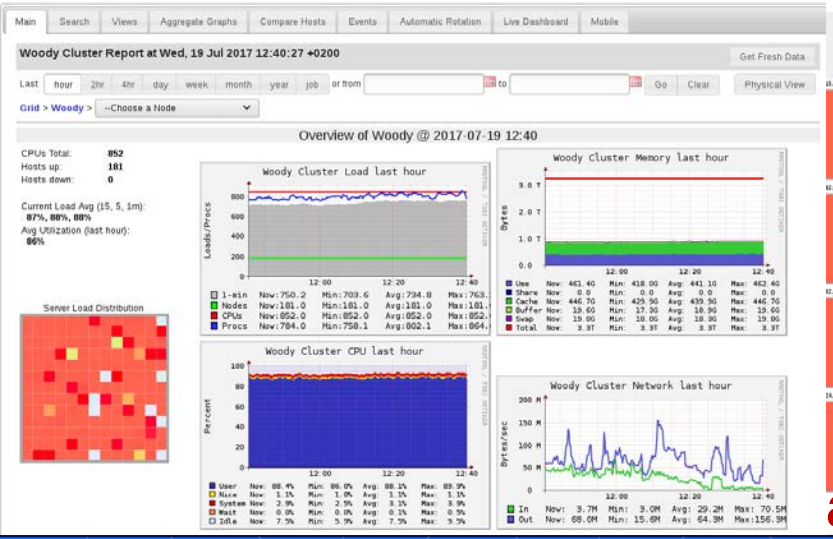
How it all started …

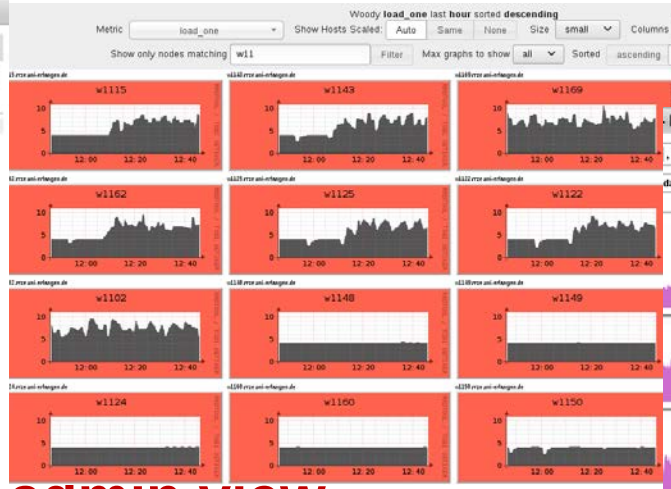
- **2003:** 1st HPC-Linux-Cluster @ RRZE
  - with stateless nodes, with Ganglia for system monitoring
- **2004?:** additional metrics feed to ganglia or directly to RRDs
  - /usr/sbin/gangliametrics.pl (in the meantime ~1000 lines)
  - long time only used for system-wide aggregated graphs and health-checks



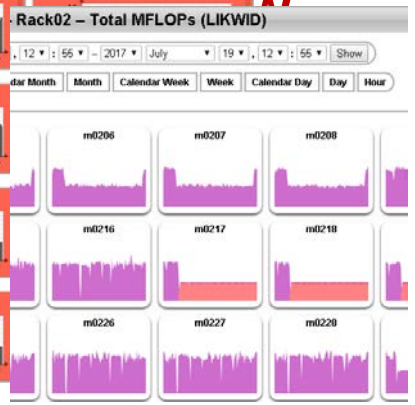
How it all started …

- **2003:** 1st HPC-Linux-Cluster @ RRZE
  - with stateless nodes, with Ganglia for system monitoring
- **2004?:** additional metrics feed to ganglia or directly to RRDs
  - /usr/sbin/gangliametrics.pl (in the meantime ~1000 lines)
  - long time only used for system-wide aggregated graphs and health-checks
  - but no real insights due to missing connection to jobs, job start, …





admin view



How it all started … with quick hacks as proofs-of-concept (which now celebrate their 5th anniversary and are in daily use)

- **2003:** 1st HPC-Linux-Cluster @ RRZE
  - with stateless nodes, with Ganglia for system monitoring
- **2004?:** additional metrics feed to ganglia or directly to RRDs
  - /usr/sbin/gangliametrics.pl (in the meantime ~1000 lines)
  - long time only used for system-wide aggregated graphs and health-checks
  - but no real insights due to missing connection to jobs, job start, …
- **2012:** show-job-counters.pl (~400 lines) **=> live admin view**
  job-statistic.pl (~300 lines) **=> post-mortem user view**
- **2015:** instantaneous cluster roofline plot **=> live admin view**
  (~100 lines Python + 400 lines HTML/Javascript + 2 JS libs)
- **2016:** extension for Megware's ClustWare & slurm as sources
- **2017:** special GPU views (~125/400 lines PHP)
  => **fixed time live admin view**

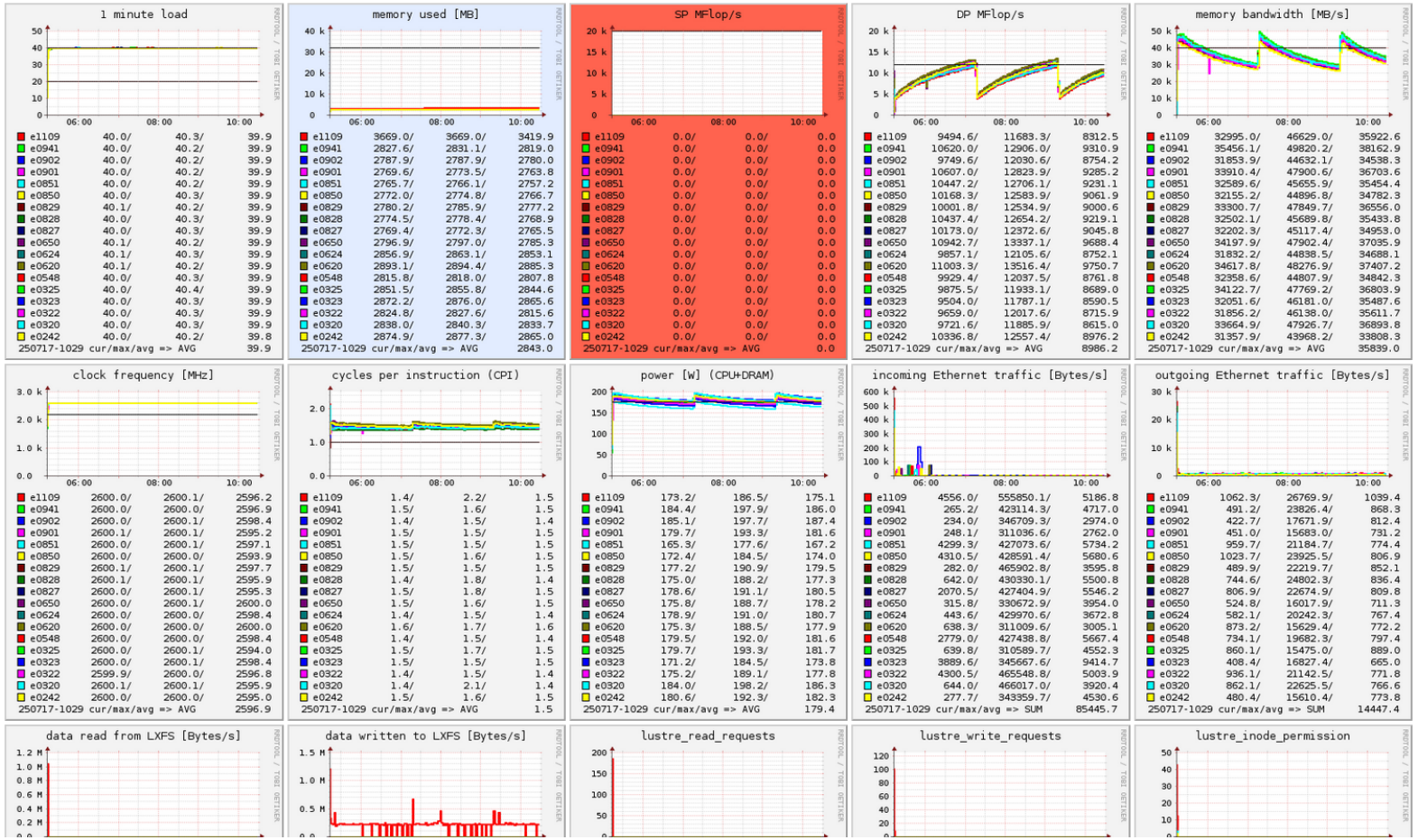
additional job-based view

Let’s have a demo: (1) port-mortem user view

https://www.hpc.rrze.uni-erlangen.de/HPC-Status/job-info.php

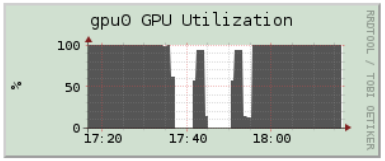
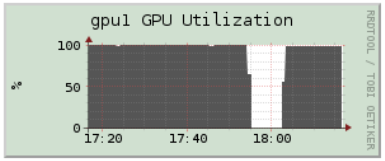
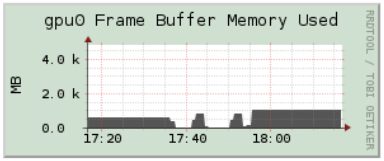
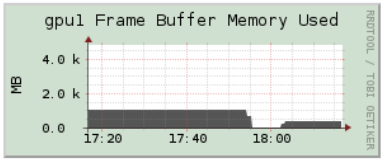
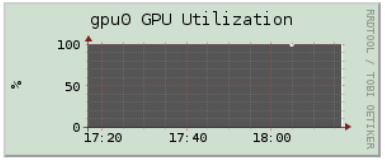
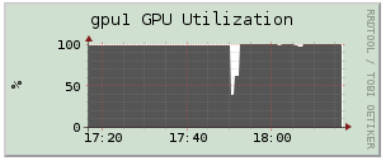
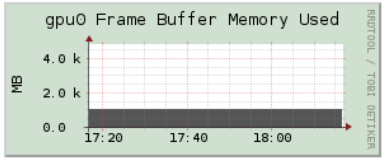
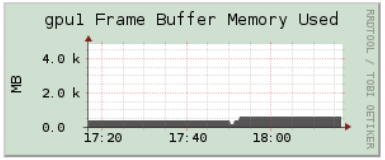
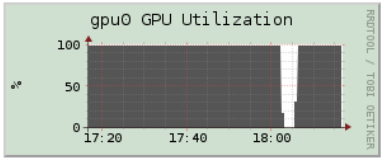
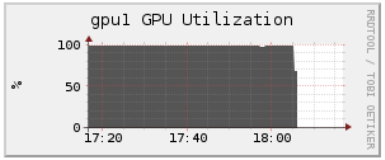
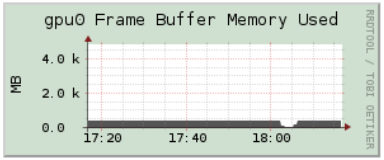
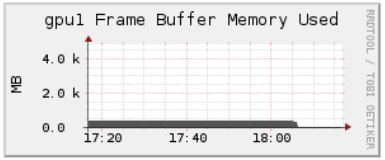
```
Clearing buffers and caches on the nodes.
Power management available, enabling ondemand governor
End of prologue: Tue Jul 18 10:30:01 CEST 2017

... user output from the job script ...

Starting epilogue... Wed Jul 19 10:25:23 CEST 2017
=== JOB_STATISTICS ===
=== current date     : Wed Jul 19 10:25:23 CEST 2017
= Job-ID             : 77498.eadm
= Job-Name           : 363KCO2
= Queue              : work
= PBS_O_WORKDIR      : /home/hpc/hpc0000h/363KCO2
= Requested resources: nodes=4:ppn=40,walltime=23:55:00
= Used resources     : walltime=23:55:19
= Node list          : e0550,e0546,e0545,e0544
= AccessKey          : 0bb83d4
======================
Power management available, setting all CPUs to ondemand govenor
```

Let’s have a demo: (2) live admin view (Jobs)



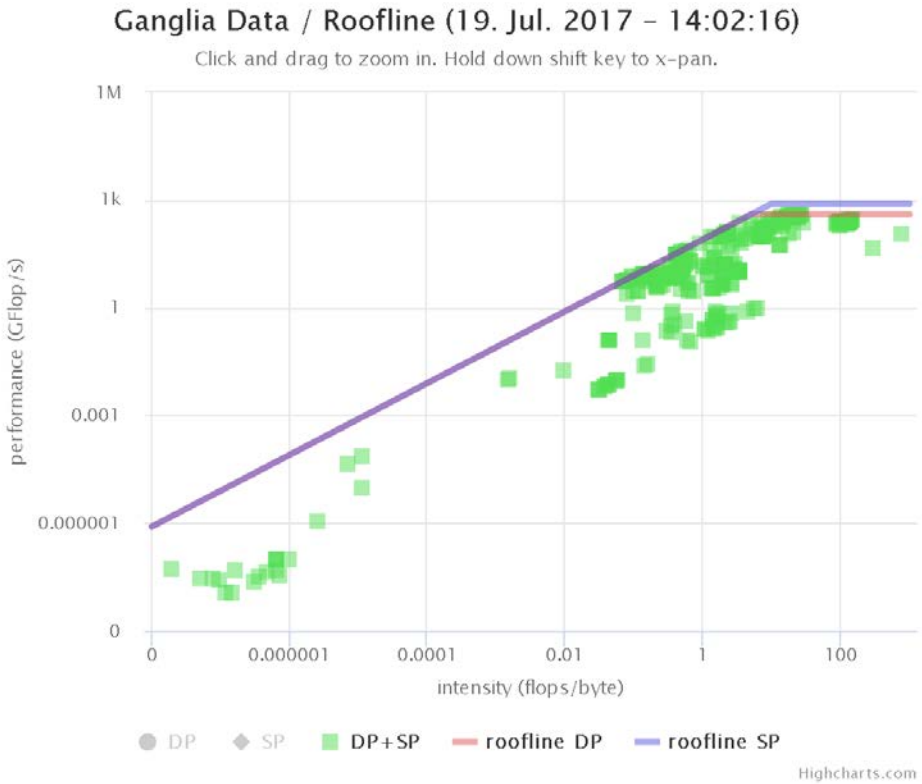
Let’s have a demo: (2) live admin view (GPUs)

group	capn	mfbi	sum
running GPUs	4	14	18
remaining running GPU-walldays	3.5	3.1	
queued GPUs			0

GTX-980 nodes	gpu0+GPU+Utilization	gpu1+GPU+Utilization	gpu0+FB+Memory+Used	gpu1+FB+Memory+Used
tg001 (0-3@140797) (4-7@140798)				
tg002 (0-3@140762) (4-7@140796)				
tg003 (0-3@140799)				

Let’s have a demo: (3) instantaneous roofline

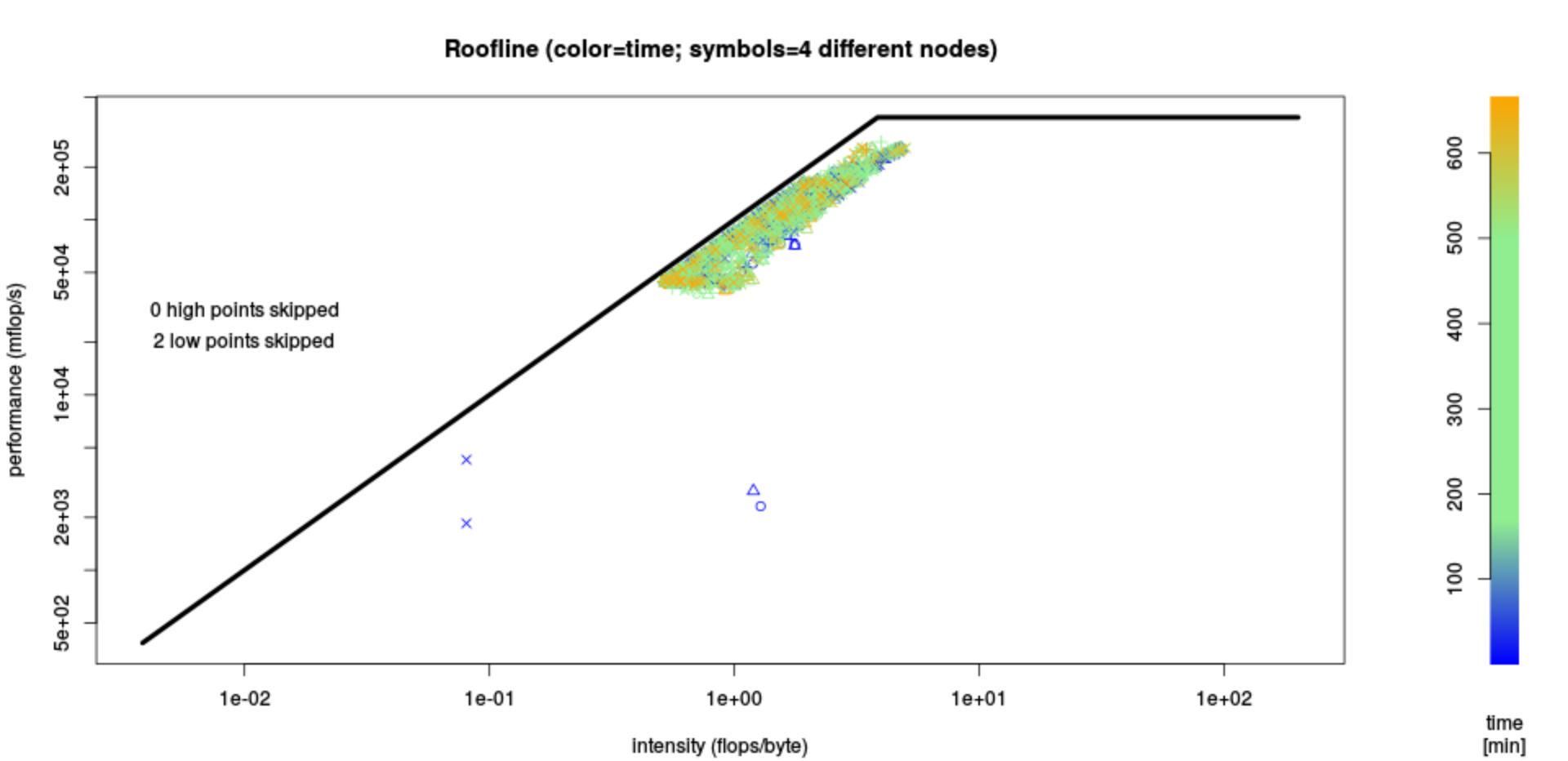
- only instantaneous data (i.e. from the last collector run) shown
- connection to other views
- but now filter for job / user implemented (yet)

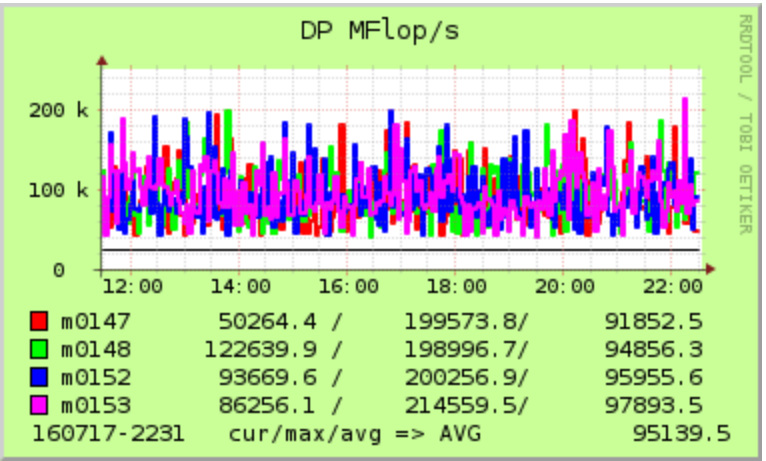


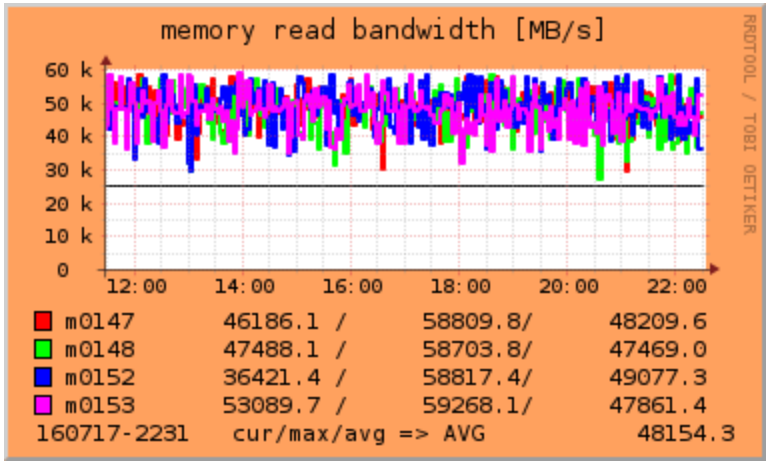
Let’s have a demo: (4) post-mortem roofline

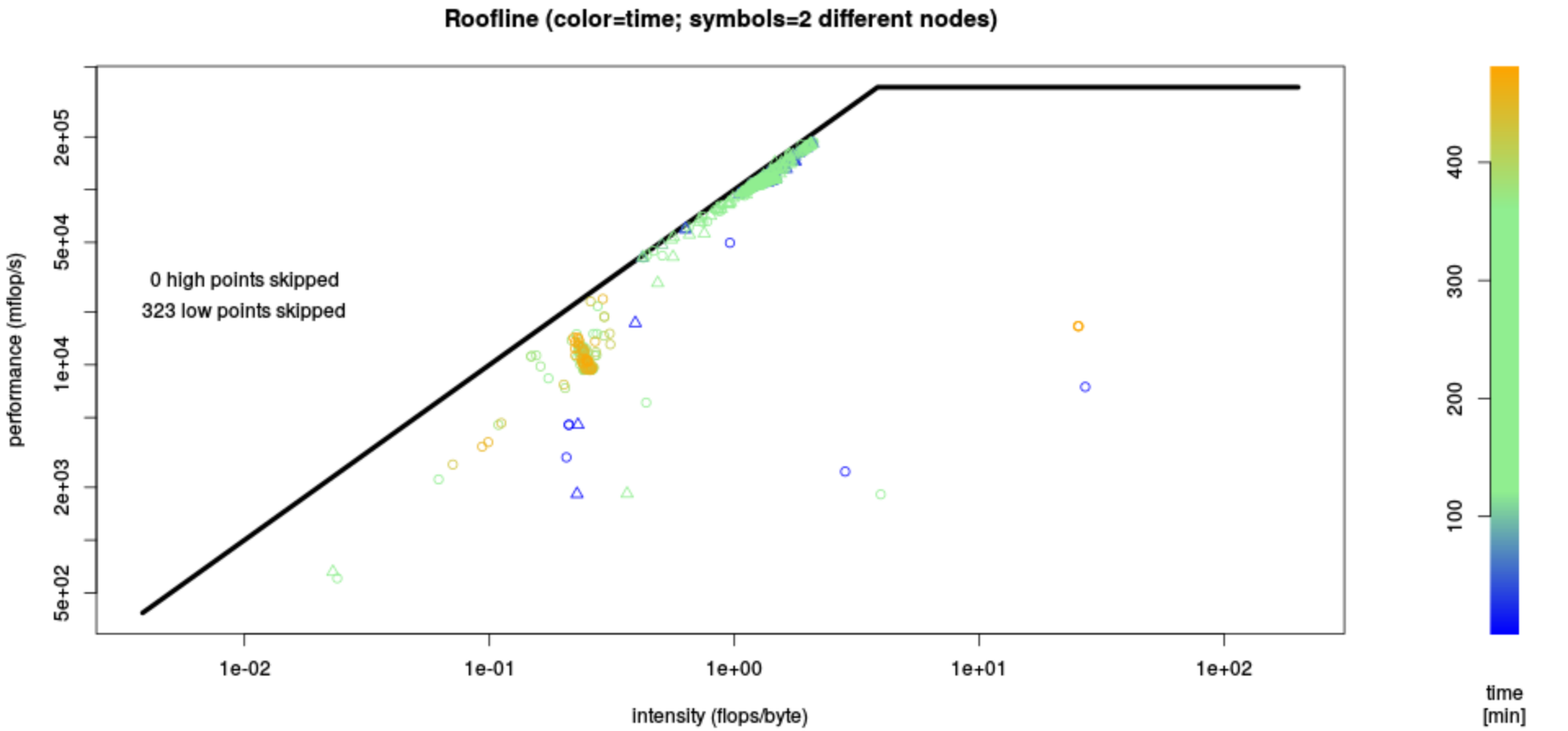
- Roofline *time evolution of a single job*
  - color = time
  - symbols = different nodes

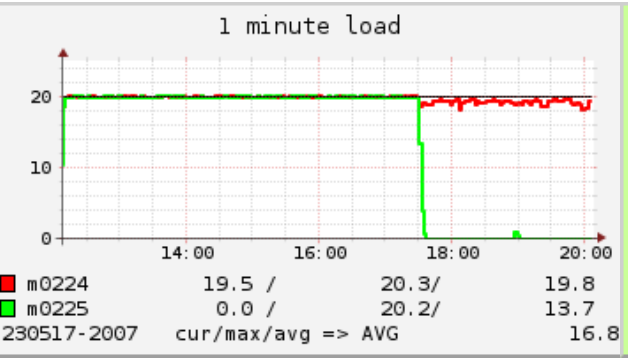
Two examples …

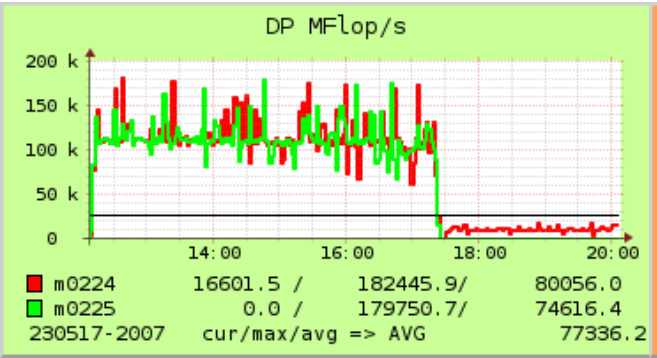

Roofline (color=time; symbols=4 different nodes)
performance (mflop/s)
5e+02
2e+03
1e+04
5e+04
2e+05
0 high points skipped
2 low points skipped
1e-02
1e-01
1e+00
1e+01
1e+02
intensity (flops/byte)
100
200
300
400
500
600
time
[min]

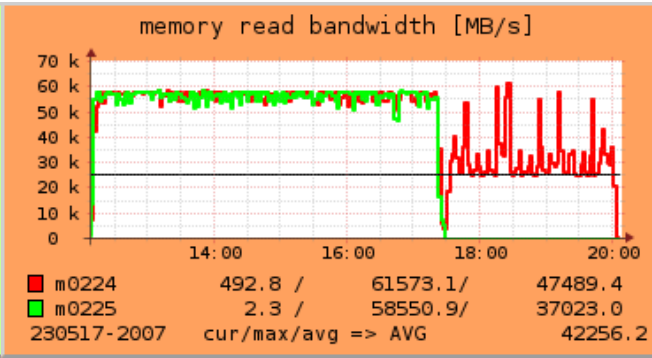

DP MFlop/s
200 k
100 k
0
12:00
14:00
16:00
18:00
20:00
22:00
RRDTOOL / TOBI OETIKER
m0147 50264.4 / 199573.8/ 91852.5
m0148 122639.9 / 198996.7/ 94856.3
m0152 93669.6 / 200256.9/ 95955.6
m0153 86256.1 / 214559.5/ 97893.5
160717-2231 cur/max/avg => AVG 95139.5


memory read bandwidth [MB/s]
60 k
50 k
40 k
30 k
20 k
10 k
0
12:00
14:00
16:00
18:00
20:00
22:00
RRDTOOL / TOBI OETIKER
m0147 46186.1 / 58809.8/ 48209.6
m0148 47488.1 / 58703.8/ 47469.0
m0152 36421.4 / 58817.4/ 49077.3
m0153 53089.7 / 59268.1/ 47861.4
160717-2231 cur/max/avg => AVG 48154.3


Roofline (color=time; symbols=2 different nodes)
performance (mflop/s)
5e+02
2e+03
1e+04
5e+04
2e+05
0 high points skipped
323 low points skipped
1e-02
1e-01
1e+00
1e+01
1e+02
intensity (flops/byte)
100
200
300
400
time
[min]


1 minute load
20
10
0
14:00
16:00
18:00
20:00
m0224 19.5 / 20.3/ 19.8
m0225 0.0 / 20.2/ 13.7
230517-2007 cur/max/avg => AVG 16.8


DP MFlop/s
200 k
150 k
100 k
50 k
0
14:00
16:00
18:00
20:00
m0224 16601.5 / 182445.9/ 80056.0
m0225 0.0 / 179750.7/ 74616.4
230517-2007 cur/max/avg => AVG 77336.2


memory read bandwidth [MB/s]
70 k
60 k
50 k
40 k
30 k
20 k
10 k
0
14:00
16:00
18:00
20:00
m0224 492.8 / 61573.1/ 47489.4
m0225 2.3 / 58550.9/ 37023.0
230517-2007 cur/max/avg => AVG 42256.2

The technique behind

No additional data collected –
only new view on the existing RRDs from system monitoring!

1. Get start time and nodes of a job from batch system
2. Call "**rrdtool**" with appropriate command line and you have the images => only some glue code, no real magic

```
/usr/bin/rrdtool graph  -  -w 250 -t "mem_free" -s -6324s   DEF:e0539=/var/lib/ganglia/rrds/emmy/e0539/mem_free.rrd:sum:AVERAGE
DEF:e0538=/var/lib/ganglia/rrds/emmy/e0538/mem_free.rrd:sum:AVERAGE
DEF:e0537=/var/lib/ganglia/rrds/emmy/e0537/mem_free.rrd:sum:AVERAGE
DEF:e0527=/var/lib/ganglia/rrds/emmy/e0527/mem_free.rrd:sum:AVERAGE
DEF:e0520=/var/lib/ganglia/rrds/emmy/e0520/mem_free.rrd:sum:AVERAGE  LINE2:e0539#ff0000:'e0539' GPRINT:e0539:LAST:'%9.1lf/'
GPRINT:e0539:MAX:'%9.1lf/' GPRINT:e0539:AVERAGE:'%9.1lf\n' LINE2:e0538#00ff00:'e0538' GPRINT:e0538:LAST:'%9.1lf/'
GPRINT:e0538:MAX:'%9.1lf/' GPRINT:e0538:AVERAGE:'%9.1lf\n' LINE2:e0537#0000ff:'e0537' GPRINT:e0537:LAST:'%9.1lf/'
GPRINT:e0537:MAX:'%9.1lf/' GPRINT:e0537:AVERAGE:'%9.1lf\n' LINE2:e0527#ff00ff:'e0527' GPRINT:e0527:LAST:'%9.1lf/'
GPRINT:e0527:MAX:'%9.1lf/' GPRINT:e0527:AVERAGE:'%9.1lf\n' LINE2:e0520#00ffff:'e0520' GPRINT:e0520:LAST:'%9.1lf/'
GPRINT:e0520:MAX:'%9.1lf/' GPRINT:e0520:AVERAGE:'%9.1lf\n' CDEF:sum=e0539,e0538,+,e0537,+,e0527,+,e0520,+ CDEF:avg=sum,5,/
COMMENT:`date +%d%m%y-%H%M`' cur/max/avg => AVG' GPRINT:avg:AVERAGE:'%10.1lf'
```

- **For admin view:** simple cgi-script and ***dynamic*** images
- **For post-mortem user view:** generate ***static*** images during epilogue and save the images (and raw data) for some time (ensures that high-res RRD data is still available; no need for users to have access to RRD files)