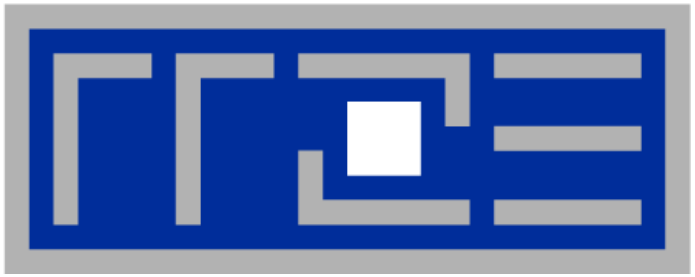


Monitoring, Accounting und Nutzerverwaltung auf den HPC-Systemen des RRZE

Georg Hager, Jan Treibig, Michael Meier, Thomas Zeiser, Markus Wittmann, Holger Stengel

HPC Services

Regionales Rechenzentrum Erlangen (RRZE)

Friedrich-Alexander-Universität (FAU) Erlangen-Nürnberg

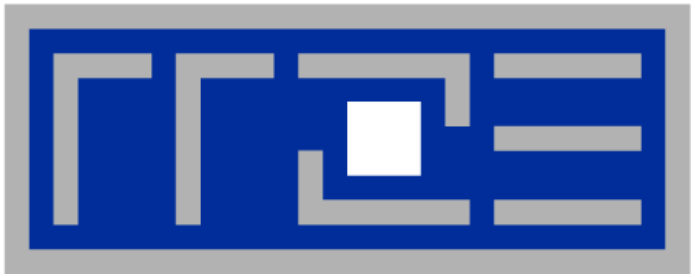
ZIH Dresden, 25.08.2011

Abstract

Monitoring, Accounting und Nutzerverwaltung sind zentrale Komponenten im Betrieb einer HPC-Infrastruktur. Der Vortrag stellt die am Regionalen Rechenzentrum Erlangen (RRZE) heute genutzten Werkzeuge vor: Tools für das Auslesen von Hardware Performance Countern auf x86-Systemen zum Zwecke der Analyse einzelner Applikationen und zum Monitoring des Gesamtsystems, eine flexible Datenbank-basierte Lösung für das Rechenzeit-Accounting, und die Nutzerverwaltung für HPC unter den Randbedingungen eines universitätsweiten IdM-Systems.

Outline

- **Resource accounting and load monitoring**
  - Architecture
  - Tools and interfaces
  - Miscellaneous
- **Performance monitoring / analysis**
  - LIKWID – Lightweight performance tools
  - likwid-perfctr in detail
  - Use cases
- **User management**
  - HPC clusters at RRZE
  - Connection with the FAU IdM system



Resource accounting, load monitoring

Resource accounting/Load monitoring

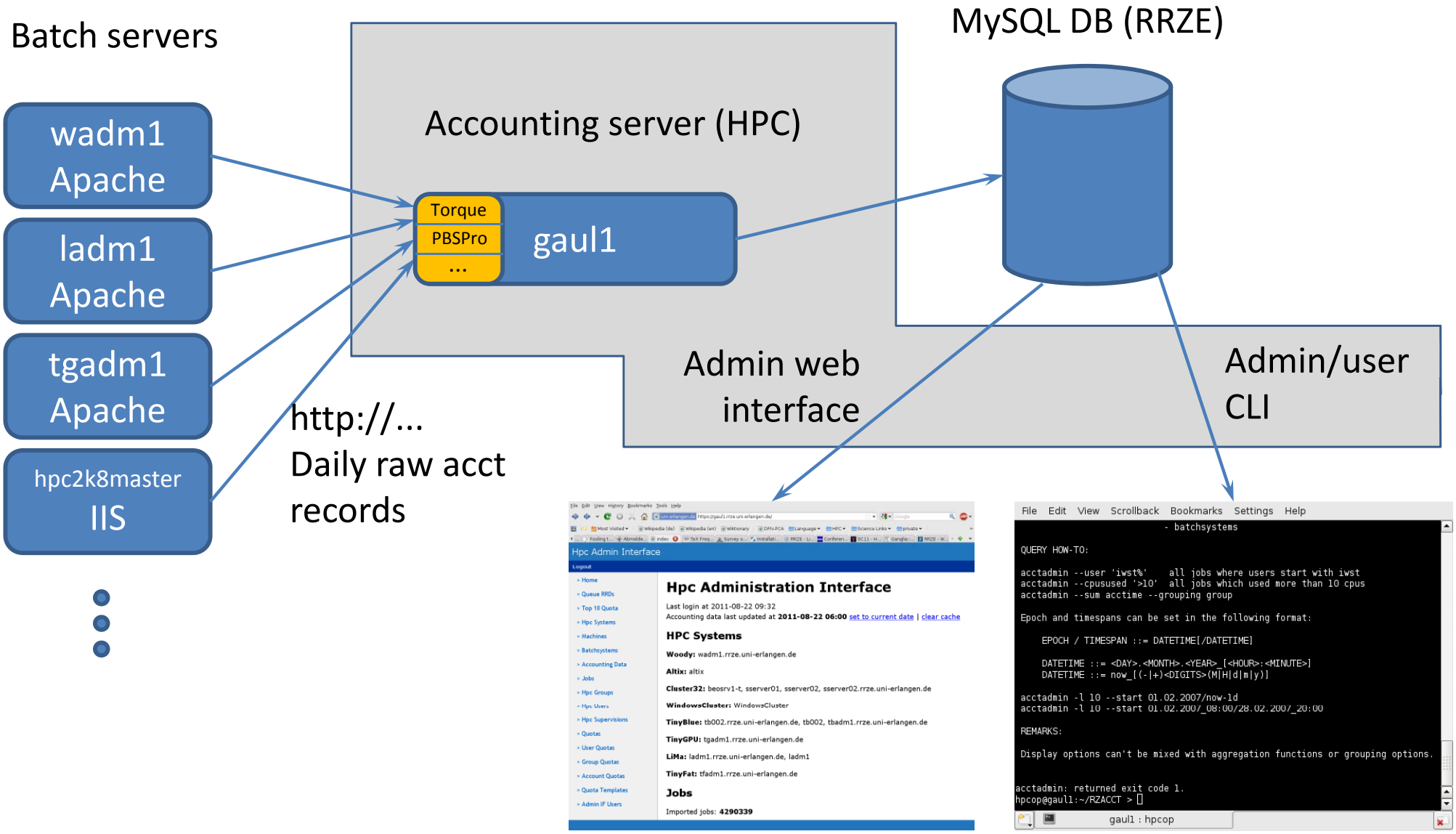
- **CPU time**
  - Used for billing (if applicable) and informational purposes
  - FAU projects and associated universities get CPU time for free
  - DFG projects need to coordinate with RRZE before submitting grant proposal – but CPU time is still free
  - Commercial projects are handled by special contracts
    - Official price list exists and serves as a starting point for negotiations
- **Load monitoring**
  - Informational purposes, online display of system load
- **Disk space**
  - Monitored (mostly), but no billing
  - Informational purposes
  - Controlled by
    - quotas (long-term storage)
    - high watermark deletion (parallel filesystems)

CPU time accounting – History

- **Basic system developed 2006/2007**
  - Student project with Ohm University of Applied Sciences, Nuremberg
  - Since November 2007 all CPU time accounting at RRZE is based on this software
- **Administrative web interface added in 2008/09**
- **User functionality in production since end of 2010**

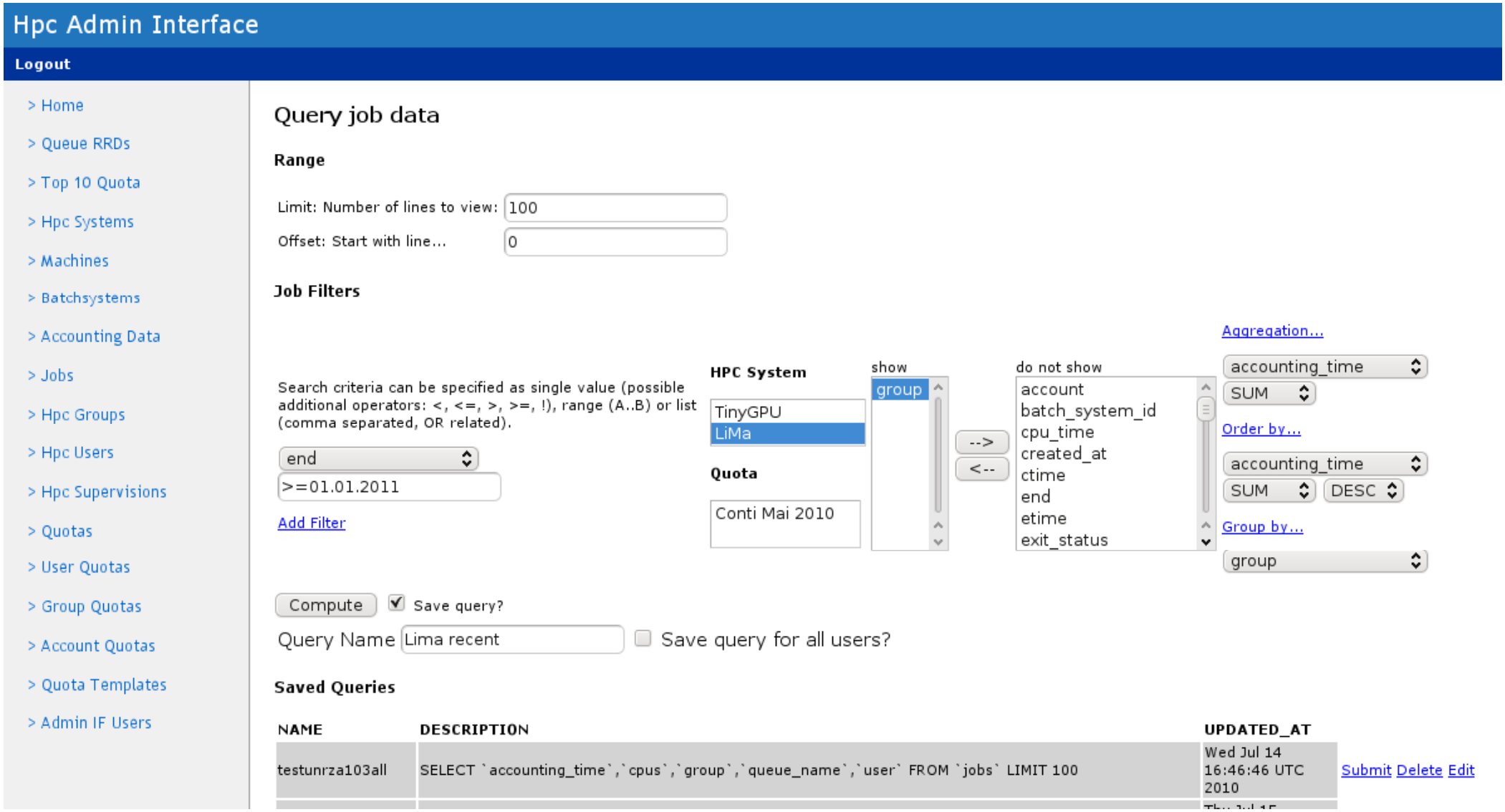
- **Low-level components, DB interface implemented in Python**
- **Database: Firebird → SQLite → MySQL**
- **Higher functions (web interface, statistics output) implemented with Ruby on Rails**

CPU time accounting – Basic architecture



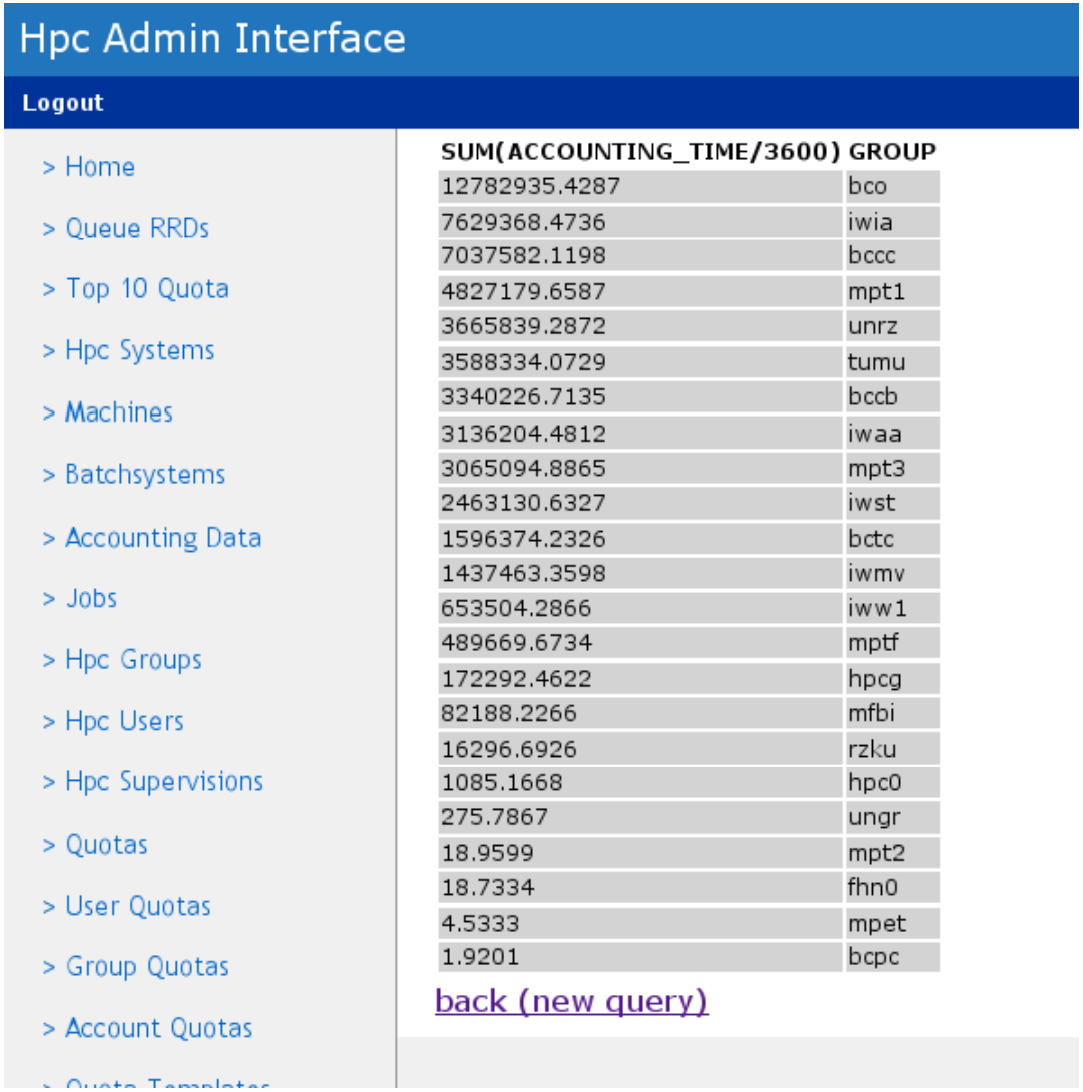
CPU time accounting – Admin web interfaces

- **A job data query**



CPU time accounting – Admin web interface

- **Result**



CPU time accounting – Admin CLI

- **Same query as on previous slide on the command line:**

```
$ acctadmin.sh --end 01.01.2011/now --hpcsystem Lima \
      --grouping group --sum acctime
```

- **Other examples:**

```
$ acctadmin.sh --end 01.01.2010/31.12.2010 --hpcsystem Woody \
      --user unrz55 --sum acctime

$ acctadmin.sh --end 01.01.2011/now --hpcsystem Lima \
      --cpus ">255" --sum acctime
```

- **DB queries are used to provide monthly data to RRZE controller for billing**

CPU time accounting – User “CLI”

- **We provide daily updated per-user stats**
- **AU = “Accounting Unit”**

  T_i: HW thread hours used on system i since start of accounting period (Oct 1st)

  P_i: Official CPU time fee in € per thread-hour on system i

$$AUs_i = T_i \times P_i$$

- This is purely informational for most users/groups
- AUs may translate to € due in some cases, if charges apply

- AUs are only calculated for the current accounting period (starting Oct 1st)

CPU time accounting – User “CLI”

Group summary data

User-specific data

```
unrz55@woody2:~ > showacct.sh
Generated:    2010-11-29 06:09
Last Updated: 2010-11-29 06:00
= Group: unrz ==========================================================
system                last month       last year  current period          AUs
Woody                      381 h         55976 h           771 h       77 AUs
Altix                        0 h             2 h             0 h        0 AUs
Cluster32                    0 h          8010 h             0 h        0 AUs
WindowsCluster               0 h          1468 h            10 h        0 AUs
TinyBlue                 17850 h       1212996 h        100727 h     3777 AUs
TinyGPU                  10545 h         59273 h         22570 h      846 AUs
 sum                                                                 4701 AUs

- User: unrz55 (Dr. Georg Hager) ---------------------------------------
system                last month       last year  current period          AUs
Woody                        3 h          3432 h           195 h       19 AUs
Altix                        0 h             0 h             0 h        0 AUs
Cluster32                    0 h             0 h             0 h        0 AUs
WindowsCluster               0 h             0 h             0 h        0 AUs
TinyBlue                   610 h         33420 h           642 h       24 AUs
TinyGPU                     17 h            33 h            17 h        1 AUs
 sum                                                                    44 AUs
```

now – 30d

now – 365d

since Oct 1st

CPU time accounting – User “CLI”

Head of group/chair may assign a supervisor role:

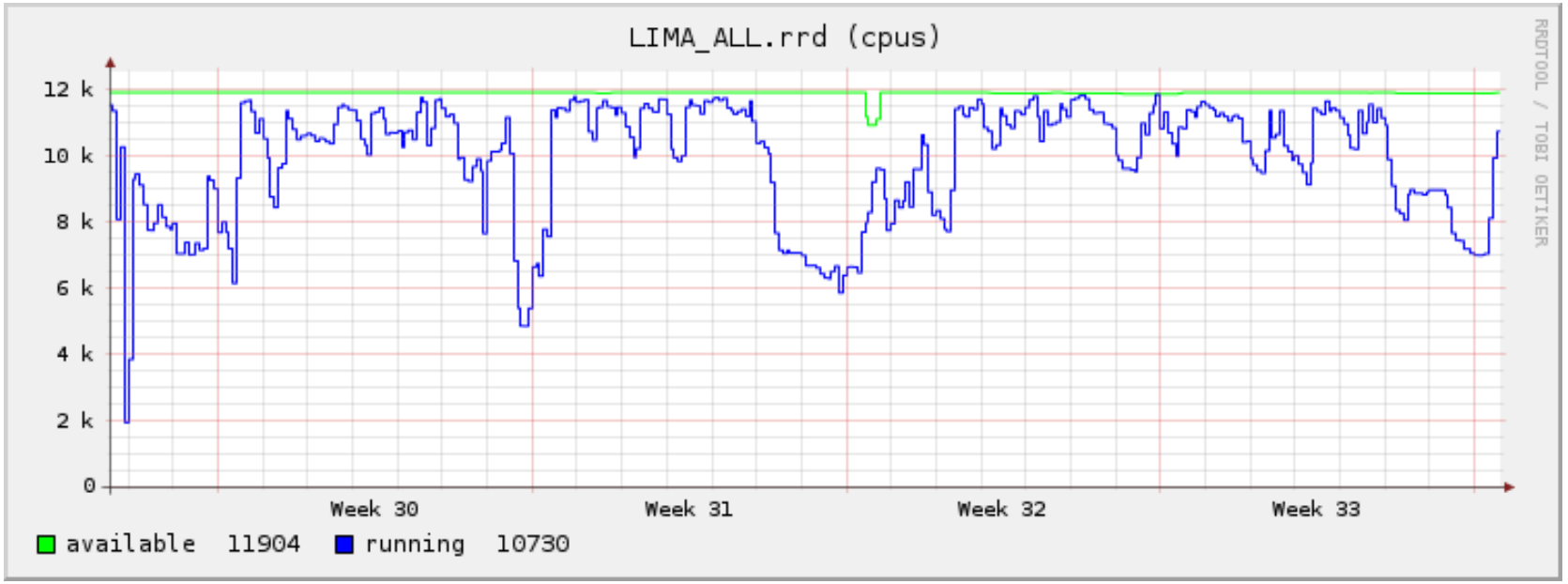
```
unrz55@woody2:~ > showacct.sh -s
Generated:    2010-11-29 06:10
Last Updated: 2010-11-29 06:00
= Group: unrz ========================================================
system                last month       last year  current period          AUs
Woody                      381 h         55976 h             771 h        77 AUs
Altix                        0 h             2 h               0 h         0 AUs
Cluster32                    0 h          8010 h               0 h         0 AUs
WindowsCluster               0 h          1468 h              10 h         0 AUs
TinyBlue                 17850 h       1212996 h          100727 h      3777 AUs
TinyGPU                  10545 h         59273 h           22570 h       846 AUs
 sum                                                                    4701 AUs

- User: el28axut (Birk Mueller) ----------------------------------------
system                last month       last year  current period          AUs
Woody                        0 h             0 h               0 h         0 AUs
Altix                        0 h             0 h               0 h         0 AUs
Cluster32                    0 h             0 h               0 h         0 AUs
WindowsCluster               0 h             0 h               0 h         0 AUs
TinyBlue                     0 h             0 h               0 h         0 AUs
TinyGPU                      0 h            59 h               0 h         0 AUs
 sum                                                                      0 AUs

- User: unrz143 (Dr. Thomas Zeiser) ------------------------------------
system                last month       last year  current period          AUs
Woody                      329 h         11882 h             330 h        33 AUs
Altix                        0 h             1 h               0 h         0 AUs
Cluster32                    0 h           934 h               0 h         0 AUs
WindowsCluster               0 h             0 h               0 h         0 AUs
TinyBlue                     0 h        806862 h             402 h        15 AUs
TinyGPU                   2931 h         12896 h            4393 h       165 AUs
 sum                                                                    213 AUs
...
```

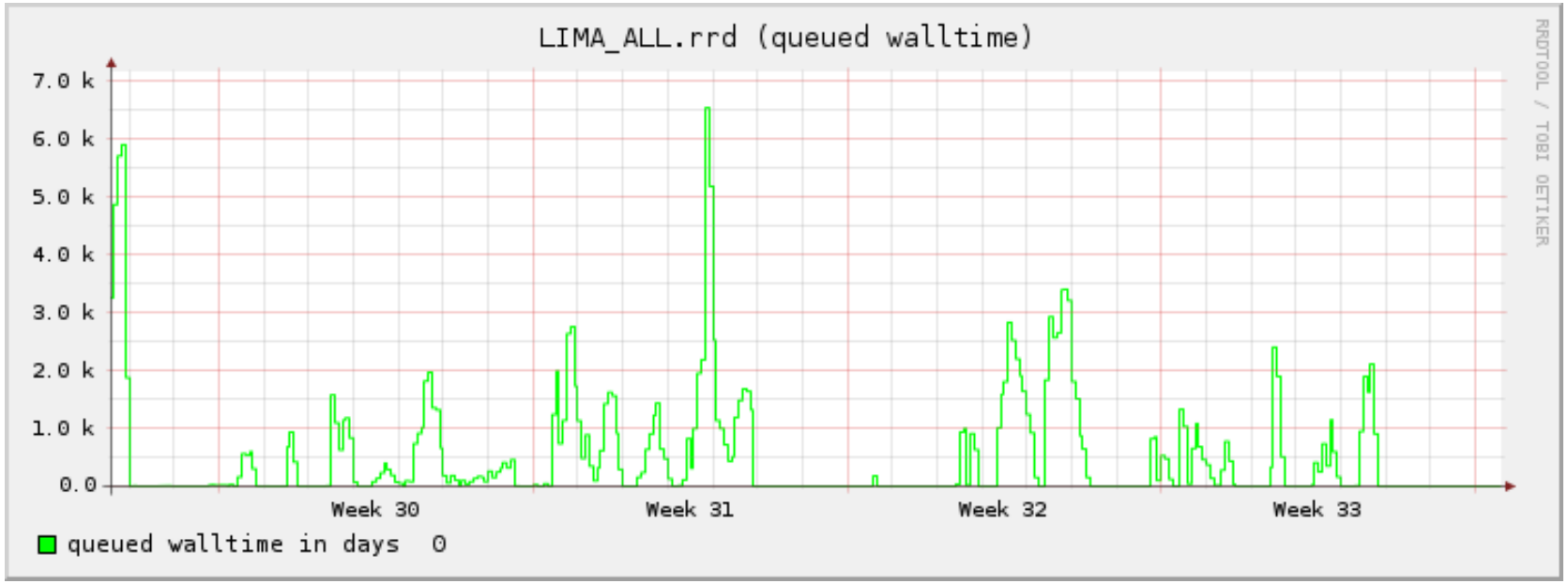
User-specific data for all group members

Load monitoring

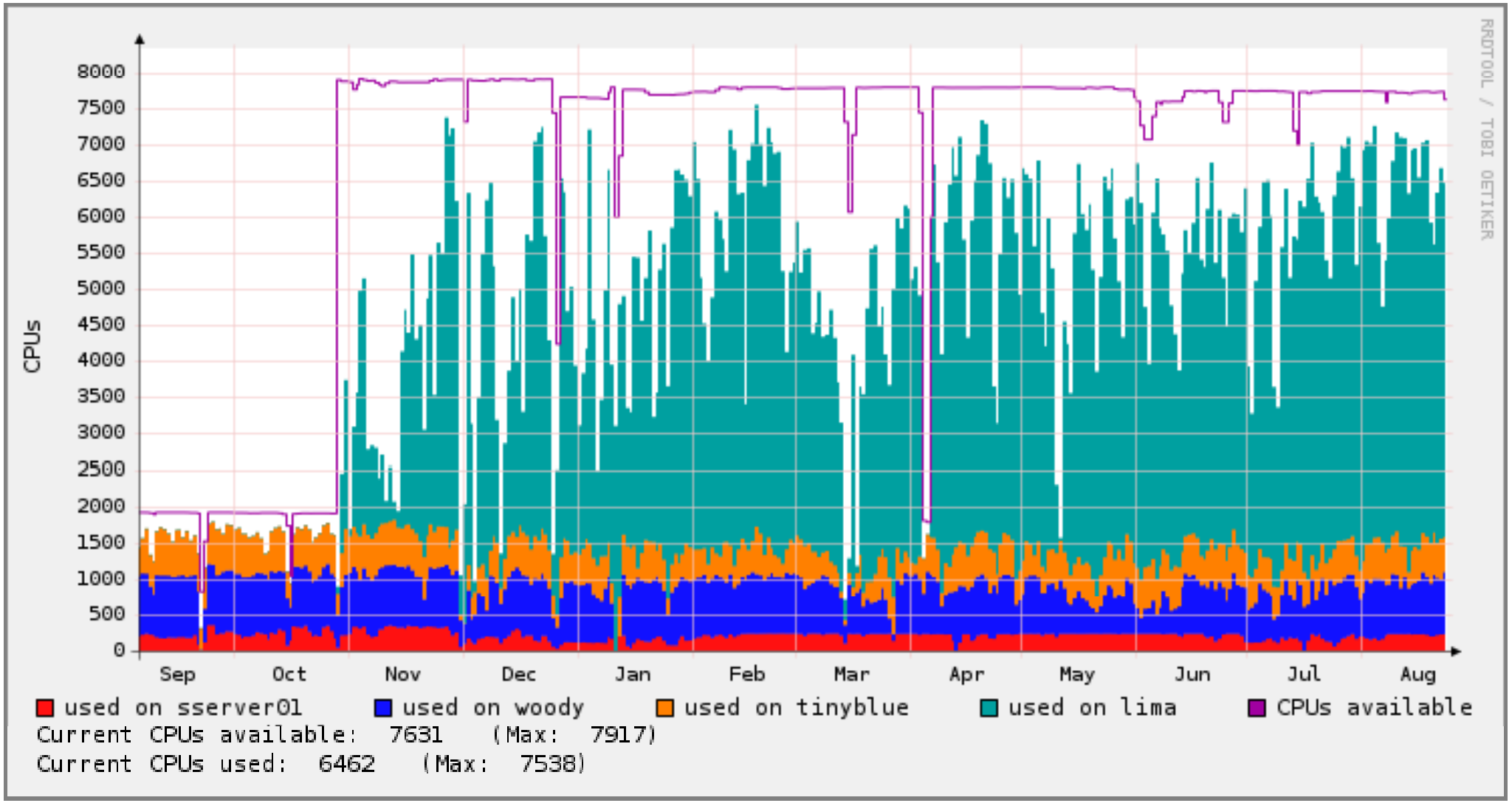
- **Continuous monitoring of**
  - “Load” (cores used as given by batch system) per cluster and per queue
  - Available cores per cluster
- **Direct queries to Torque servers via Python interface**
- **Data fed into RRD database**
  - day/week/month/year/sevenyears accumulation
  - Web interface with graphs or CLI

Examples: Overall cluster load & queued walltime





Examples: Center-wide HPC usage



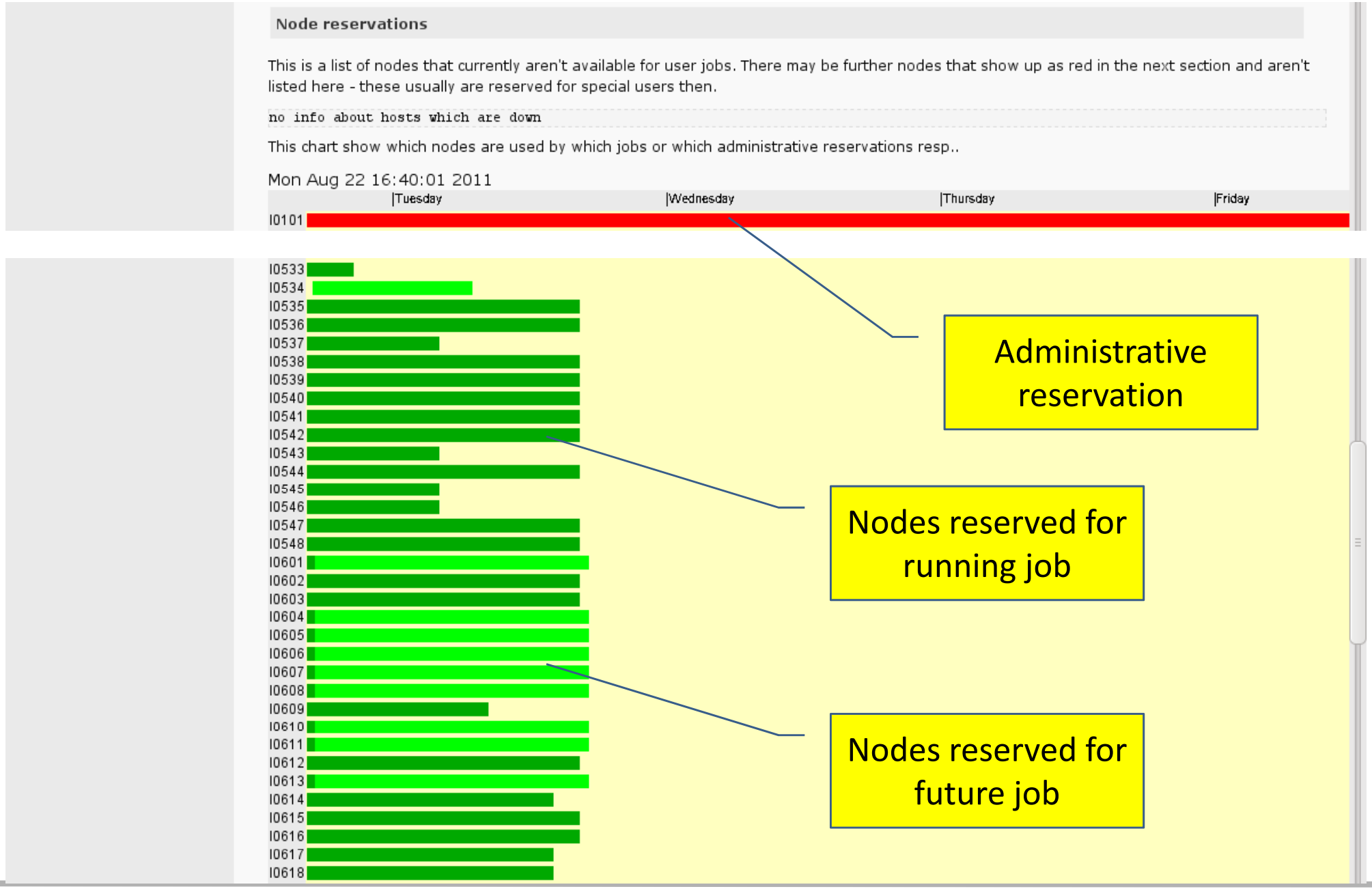
User-visible queue status

- **Users cannot see all jobs in the queue**
  - "qstat" only displays jobs from other users in the same group (RRZE hack)
  - This reduces complaints about "my job not starting"
- **Web site displays information about queue status (per cluster):**

```
===EMPTY RESOURCES CURRENTLY AVAILABLE FOR JOBS IN devel QUEUE====
 14 nodes available for 00:29:48
 13 nodes available for 00:41:42
 12 nodes available with no timelimit
===EMPTY RESOURCES CURRENTLY AVAILABLE FOR JOBS IN work QUEUE===
  2 nodes available for 00:29:48
  1 nodes available for 00:41:42
===IDLE JOBS====
JobID      Priority  XFactor  Q    Procs     WCLimit     Class      SystemQueueTime
112933*     -26963      1.1   -       96    13:59:00      work   Mon Aug 22 15:44:04
112935*     -26970      1.1   -       96    13:59:00      work   Mon Aug 22 15:51:37
112937*     -26990      1.0   -       96    11:59:00      work   Mon Aug 22 16:11:00
112939*     -35607      1.0   -      384  1:00:00:00      work   Mon Aug 22 16:15:16
===BLOCKED JOBS===
 JobID    Reason
  112895  'Hold'
  112897  'Hold'
  112898  'Hold'
==================
Generated at Mon Aug 22 16:40:01 CEST 2011
```

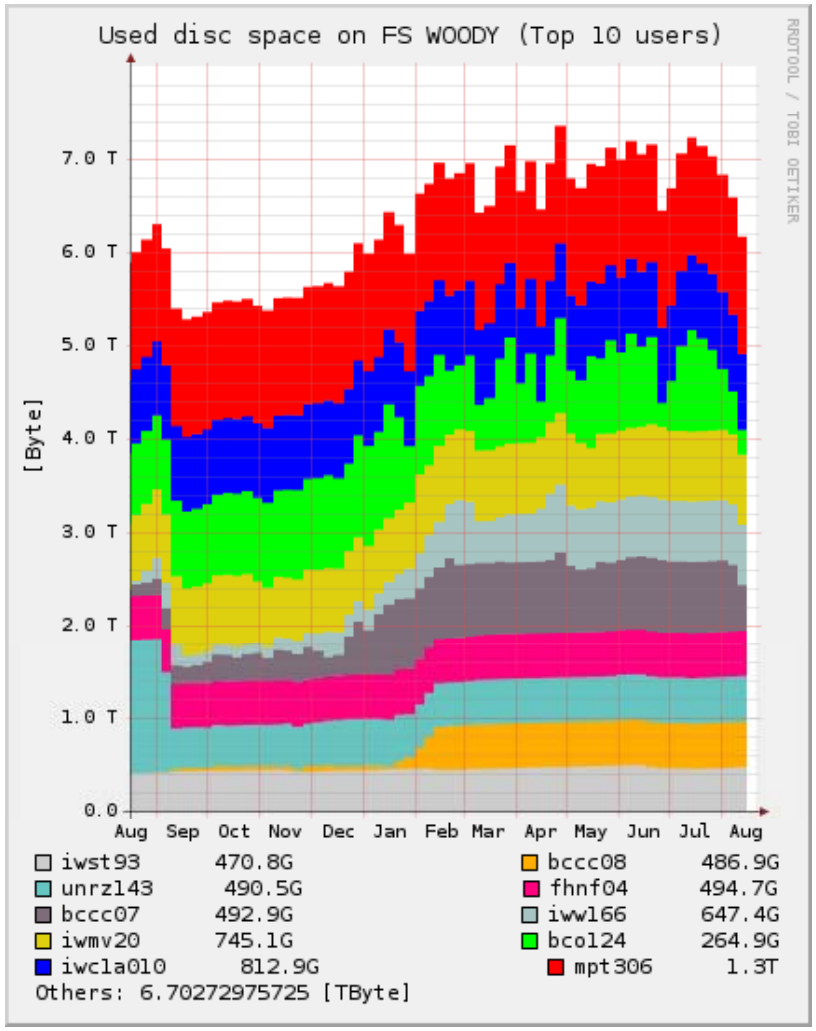
Job priority according to Maui scheduler

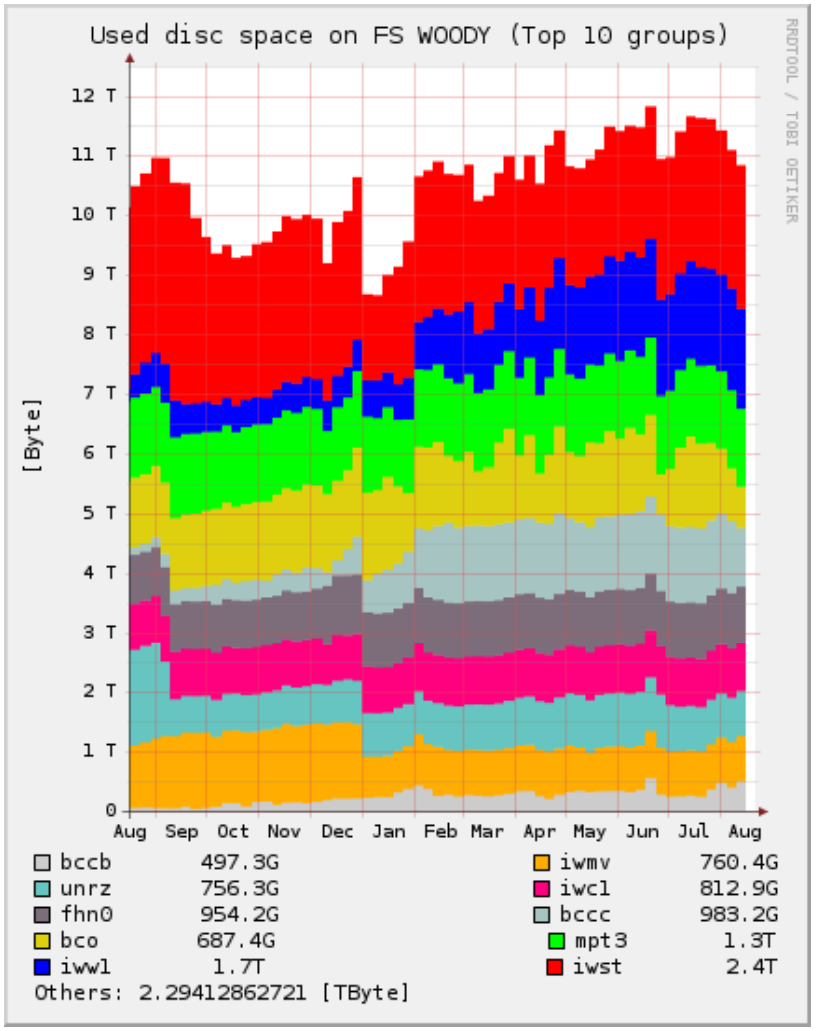
User-visible node reservations via maureswww

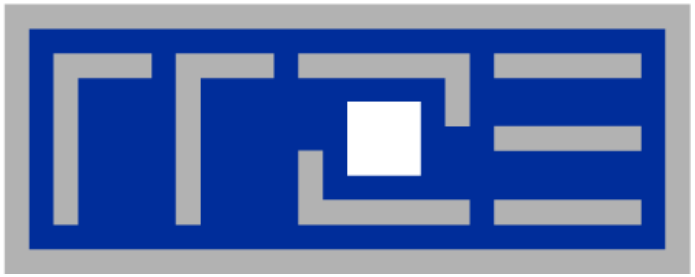


Disk space monitoring

- Regular repquota runs on all file servers → RRD
- Accounting server collects data and provides “top 10” graphs for users and groups



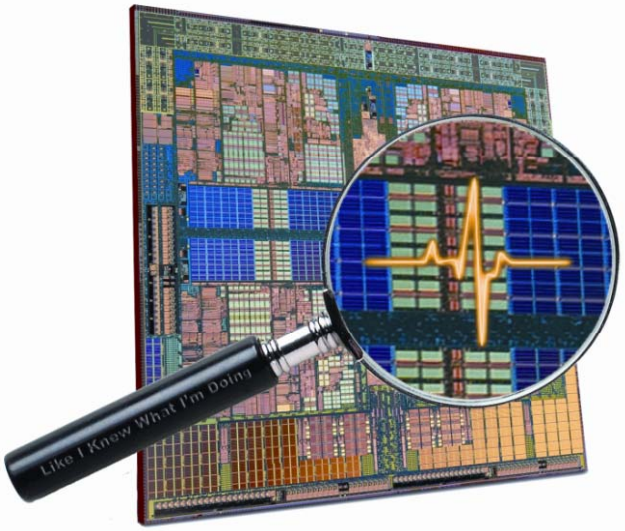




Performance monitoring and analysis

Likwid Tool Suite (developed by Jan Treibig, RRZE)

- **Command line tools for Linux**
  - easy to install
  - works with standard linux 2.6 kernel
  - simple and clear to use
  - supports Intel and AMD CPUs
  - Open source project (GPL v2):
    `http://code.google.com/p/likwid/`



- **Current tools**
  - **likwid-topology**: Print thread and cache topology
  - **likwid-pin**: Pin threaded application without touching code
  - **likwid-perfctr:** Measure performance counters
  - likwid-perfscope: Performance oscilloscope w/ real-time display
  - likwid-powermeter: Current power consumption of chip (alpha stage)
  - likwid-features: View and enable/disable hardware prefetchers
  - likwid-bench: Low-level bandwidth benchmark generator tool
  - likwid-mpirun: mpirun wrapper script for easy LIKWID integration

Output of likwid-topology

```
CPU name:       Intel Core i7 processor
CPU clock:      2666683826 Hz
*************************************************************
Hardware Thread Topology
*************************************************************
Sockets:                2
Cores per socket:       4
Threads per core:       2
-------------------------------------------------------------
HWThread        Thread          Core            Socket
0               0               0               0
1               1               0               0
2               0               1               0
3               1               1               0
4               0               2               0
5               1               2               0
6               0               3               0
7               1               3               0
8               0               0               1
9               1               0               1
10              0               1               1
11              1               1               1
12              0               2               1
13              1               2               1
14              0               3               1
15              1               3               1
-------------------------------------------------------------
```

Output of likwid-topology continued

```
Socket 0: ( 0 1 2 3 4 5 6 7 )
Socket 1: ( 8 9 10 11 12 13 14 15 )
--------------------------------------------------------------

**************************************************************
Cache Topology
**************************************************************
Level:   1
Size:    32 kB
Cache groups:   ( 0 1 ) ( 2 3 ) ( 4 5 ) ( 6 7 ) ( 8 9 ) ( 10 11 ) ( 12 13 ) ( 14 15 )
--------------------------------------------------------------
Level:   2
Size:    256 kB
Cache groups:   ( 0 1 ) ( 2 3 ) ( 4 5 ) ( 6 7 ) ( 8 9 ) ( 10 11 ) ( 12 13 ) ( 14 15 )
--------------------------------------------------------------
Level:   3
Size:    8 MB
Cache groups:   ( 0 1 2 3 4 5 6 7 ) ( 8 9 10 11 12 13 14 15 )
--------------------------------------------------------------
**************************************************************
NUMA Topology
**************************************************************
NUMA domains: 2
--------------------------------------------------------------
Domain 0:
 Processors:  0 1 2 3 4 5 6 7
Memory: 5182.37 MB free of total 6132.83 MB
--------------------------------------------------------------
Domain 1:
 Processors:  8 9 10 11 12 13 14 15
Memory: 5568.5 MB free of total 6144 MB
--------------------------------------------------------------
```

Output of likwid-topology

- **ASCII art output with the `-g` option!**

```
Socket 0:
+-------------------------------------+
| +------+ +------+ +------+ +------+ |
| |  0  1| |  2  3| |  4  5| |  6  7| |
| +------+ +------+ +------+ +------+ |
| +------+ +------+ +------+ +------+ |
| |  32kB| |  32kB| |  32kB| |  32kB| |
| +------+ +------+ +------+ +------+ |
| +------+ +------+ +------+ +------+ |
| | 256kB| | 256kB| | 256kB| | 256kB| |
| +------+ +------+ +------+ +------+ |
| +---------------------------------+ |
| |                8MB              | |
| +---------------------------------+ |
+-------------------------------------+
Socket 1:
+-------------------------------------+
| +------+ +------+ +------+ +------+ |
| |  8  9| |10  11| |12  13| |14  15| |
| +------+ +------+ +------+ +------+ |
| +------+ +------+ +------+ +------+ |
| |  32kB| |  32kB| |  32kB| |  32kB| |
| +------+ +------+ +------+ +------+ |
| +------+ +------+ +------+ +------+ |
| | 256kB| | 256kB| | 256kB| | 256kB| |
| +------+ +------+ +------+ +------+ |
| +---------------------------------+ |
| |                8MB              | |
| +---------------------------------+ |
+-------------------------------------+
```

- **Inspired by and based on `ptoverride` (Michael Meier, RRZE) and `taskset`**
- **Pins processes and threads to specific cores without touching code**
- **Directly supports pthreads, gcc OpenMP, Intel OpenMP**
- **Allows user to specify skip mask (shepherd threads should not be pinned)**
- **Based on combination of wrapper tool together with overloaded pthread library**
- **Can also be used as a superior replacement for taskset**
- **Supports logical core numbering within a node and within an existing CPU set**
  - Useful for running inside CPU sets defined by someone else, e.g., the MPI start mechanism or a batch system
- **Configurable colored output**

- **Usage examples:**
  - `likwid-pin -t intel -c 0,2,4-6  ./myApp parameters`
  - `mpirun likwid-pin -s 0x3 -c 0,3,5,6  ./myApp parameters`

- **Running the STREAM benchmark with likwid-pin:**

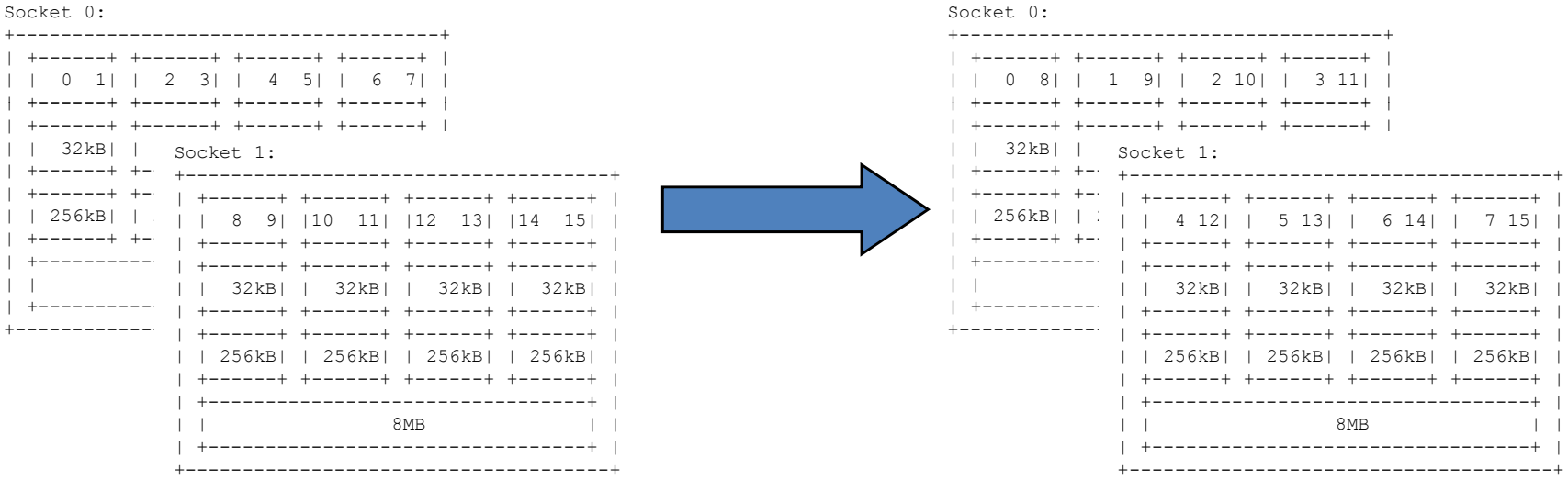
```
$ export OMP_NUM_THREADS=4
$ likwid-pin -s 0x1 -c 0,1,4,5 ./stream
[likwid-pin] Main PID -> core 0 - OK
-------------------------------------------------
 Double precision appears to have 16 digits of accuracy
 Assuming 8 bytes per DOUBLE PRECISION word
-------------------------------------------------
[... some STREAM output omitted ...]
 The *best* time for each test is used
 *EXCLUDING* the first and last iterations
[pthread wrapper] PIN_MASK: 0->1  1->4  2->5
[pthread wrapper] SKIP MASK: 0x1
[pthread wrapper 0] Notice: Using libpthread.so.0
        threadid 1073809728 -> SKIP
[pthread wrapper 1] Notice: Using libpthread.so.0
        threadid 1078008128 -> core 1 - OK
[pthread wrapper 2] Notice: Using libpthread.so.0
        threadid 1082206528 -> core 4 - OK
[pthread wrapper 3] Notice: Using libpthread.so.0
        threadid 1086404928 -> core 5 - OK
[... rest of STREAM output omitted ...]
```

Main PID always pinned

Skip shepherd thread

Pin all spawned threads in turn

- **Core numbering may vary from system to system even with identical hardware**
  - Likwid-topology delivers this information, which can then be fed into likwid-pin
- **Alternatively, likwid-pin can abstract this variation and provide a purely logical numbering (physical cores first)**



  - Across all cores in the node:
    `likwid-pin -c N:0-7 ./a.out`
  - Across the cores in each socket and across sockets in each node:
    `likwid-pin -c S0:0-3@S1:0-3 ./a.out`

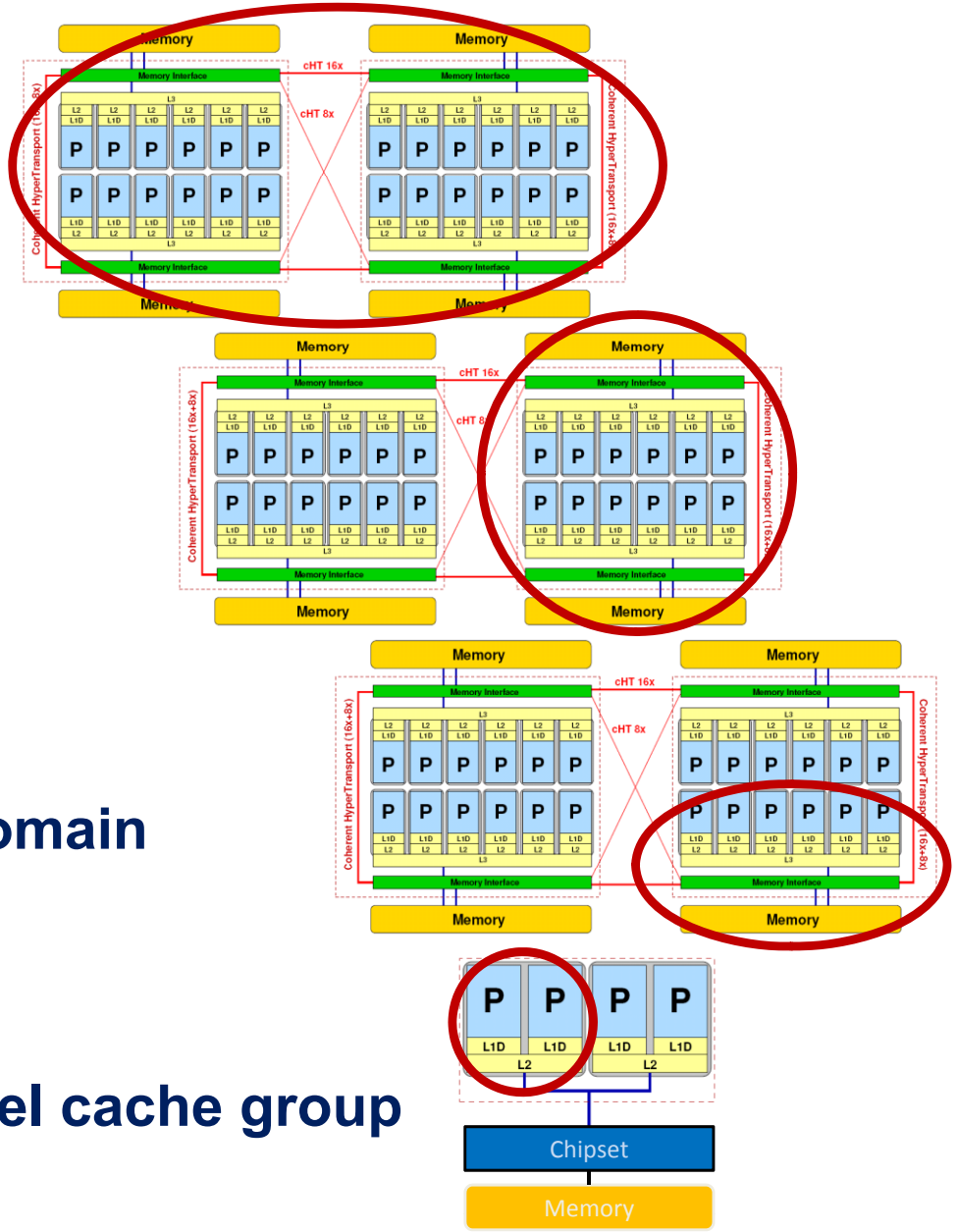
- **Possible unit prefixes**

N **node**

S **socket**

M **NUMA domain**

C **outer level cache group**



L: Logical numbering inside CPUset (just like the good old dplace)

Some facts about likwid-perfctr:

- **Implemented completely in user space (uses `msr` kernel module)**
  - Direct access to msr files OR daemon mode
- **Allows multithreaded measurements**
- **No overhead during measurements**
- **Preconfigured event groups with sensible event sets and useful derived metrics**
- **Event groups can be changed or extended by user**
- **Supported processors:**
  - Intel Core 2
  - Intel Nehalem (all variants) supporting Uncore events
  - Intel Nehalem EX (without Uncore – work in progress)
  - Sandy Bridge
  - Intel Atom
  - AMD K8/K10

Likwid-perfctr: Ways to use it

```
$ likwid-perfctr -c 0-1,6-7  -g PSTI  ./a.out
-------------------------------------------------------------
CPU type:   Intel Core Westmere processor
CPU clock:  2.93 GHz
-------------------------------------------------------------
Measuring group PSTI
-------------------------------------------------------------
+-------------------------------------+-------------+-------------+-------------+-------------+
|                Event                |   core 0    |   core 1    |   core 6    |   core 7    |
+-------------------------------------+-------------+-------------+-------------+-------------+
|          INSTR_RETIRED_ANY          | 6.32619e+09 | 6.3182e+09  | 4.96786e+09 | 4.91894e+09 |
|        CPU_CLK_UNHALTED_CORE        | 3.25469e+10 | 3.26195e+10 | 3.39702e+10 | 3.39457e+10 |
|         CPU_CLK_UNHALTED_REF        | 2.98326e+10 | 2.98997e+10 | 2.99041e+10 | 2.98821e+10 |
|    FP_COMP_OPS_EXE_SSE_FP_PACKED    | 2.80542e+09 | 2.81081e+09 | 2.88353e+09 | 2.88135e+09 |
| FP_COMP_OPS_EXE_SSE_DOUBLE_PRECISION | 2.80542e+09 | 2.81081e+09 | 2.88353e+09 | 2.88135e+09 |
|       UNC_QMC_NORMAL_READS_ANY      | 1.64096e+09 |      0      | 1.64028e+09 |      0      |
|       UNC_QMC_WRITES_FULL_ANY       | 5.47979e+08 |      0      | 5.47968e+08 |      0      |
|    UNC_QHL_REQUESTS_REMOTE_READS    |   511461    |      0      |   422731    |      0      |
+-------------------------------------+-------------+-------------+-------------+-------------+
+-----------------------------+---------+---------+-------------+---------+
|           Metric            | core 0  | core 1  |   core 6    | core 7  |
+-----------------------------+---------+---------+-------------+---------+
|             CPI             | 5.14479 | 5.16278 |    6.838    | 6.90102 |
|         Clock [MHz]         | 3200.28 | 3200.21 |   3332.25   | 3332.29 |
|  DP MFlops/s (DP assumed)   | 386.557 |  387.3  |   397.32    | 397.02  |
|       Packed MUOPS/s        | 193.279 | 193.65  |   198.66    | 198.51  |
| Memory bandwidth [MBytes/s] | 9651.61 |    0    |   9648.56   |    0    |
|  Remote Read BW [MBytes/s]  | 2.25517 |    0    |   1.86393   |    0    |
+-----------------------------+---------+---------+-------------+---------+
```

Likwid-perfctr: Group files

```
SHORT PSTI
EVENTSET
FIXC0 INSTR_RETIRED_ANY
FIXC1 CPU_CLK_UNHALTED_CORE
FIXC2 CPU_CLK_UNHALTED_REF
PMC0  FP_COMP_OPS_EXE_SSE_FP_PACKED
PMC1  FP_COMP_OPS_EXE_SSE_FP_SCALAR
PMC2  FP_COMP_OPS_EXE_SSE_SINGLE_PRECISION
PMC3  FP_COMP_OPS_EXE_SSE_DOUBLE_PRECISION
UPMC0  UNC_QMC_NORMAL_READS_ANY
UPMC1  UNC_QMC_WRITES_FULL_ANY
UPMC2 UNC_QHL_REQUESTS_REMOTE_READS
UPMC3 UNC_QHL_REQUESTS_LOCAL_READS
METRICS
Runtime [s] FIXC1*inverseClock
CPI  FIXC1/FIXC0
Clock [MHz]  1.E-06*(FIXC1/FIXC2)/inverseClock
DP MFlops/s (DP assumed) 1.0E-06*(PMC0*2.0+PMC1)/time
Packed MUOPS/s   1.0E-06*PMC0/time
Scalar MUOPS/s 1.0E-06*PMC1/time
SP MUOPS/s 1.0E-06*PMC2/time
DP MUOPS/s 1.0E-06*PMC3/time
Memory bandwidth [MBytes/s] 1.0E-06*(UPMC0+UPMC1)*64/time;
Remote Read BW [MBytes/s] 1.0E-06*(UPMC2)*64/time;
LONG
Formula:
DP MFlops/s =  (FP_COMP_OPS_EXE_SSE_FP_PACKED*2 +  FP_COMP_OPS_EXE_SSE_FP_SCALAR)/ runtime.
```

- Groups are architecture-specific
- Defined in simple text files
- **Code generation @ compile time**
- likwid-perfctr can output a list of available groups
- **An extensive documentation is configurable for every group**

Likwid-perfctr: Marker API

- **Marker API allows measurements in parts of the code (manual instrumentation)**
- **API only turns counters on/off**
  - The configuration of the counters is still done by likwid-perfctr
- **Multiple named regions are allowed**
  - Results on multiple calls are accumulated.

```
int coreID = likwid_processGetProcessorId();
likwid_markerInit(numberOfThreads,numberOfRegions);
RegionId = likwid_markerRegisterRegion("Main");
RegionId2 = likwid_markerRegisterRegion("Accum");

likwid_markerStartRegion(threadId, coreID);
. . .  // region "Main"
likwid_markerStopRegion(threadId, coreID, RegionId);


likwid_markerStartRegion(threadId, coreID);
. . .  // region "Accum"
likwid_markerStopRegion(threadId, coreID, RegionId2);


likwid_markerClose();
```

Likwid-perfctr: Other usage models

- **likwid-perfctr takes counter data on specified cores**
  - No matter what runs on the cores
    → Listening to what currently happens without any overhead (stethoscope mode):

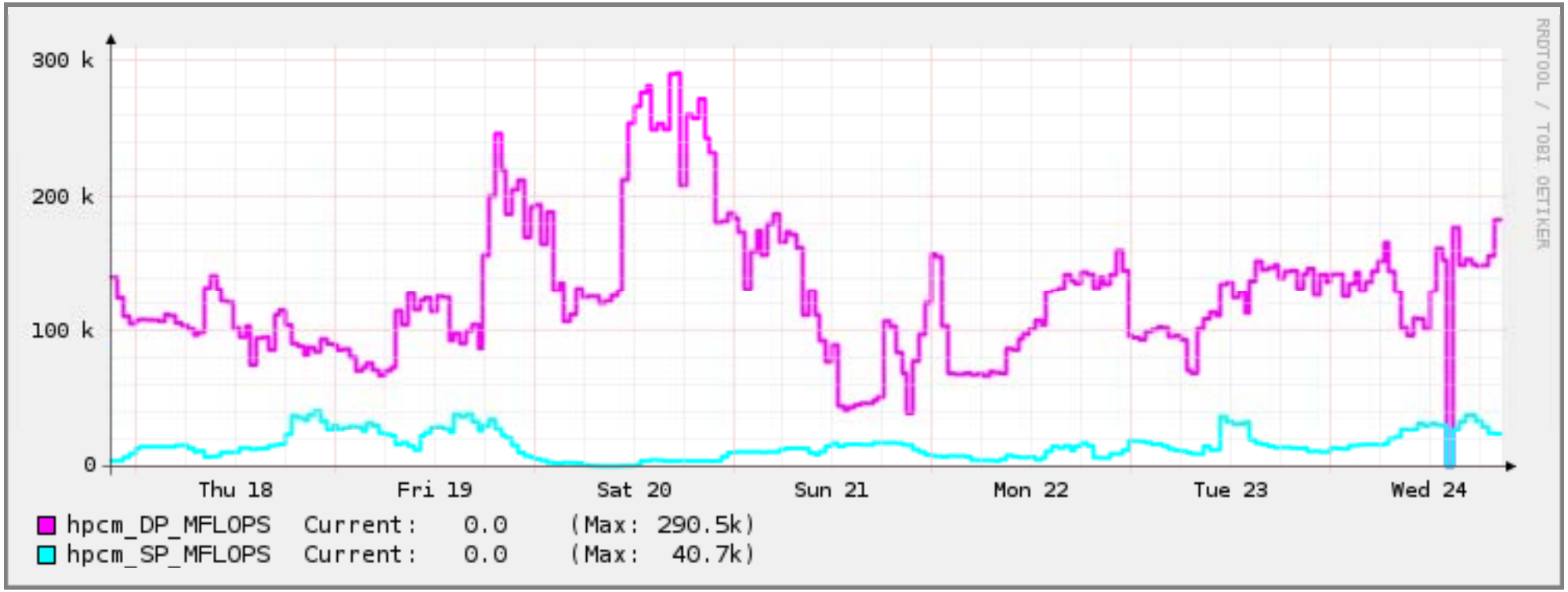
```
$ likwid-perfctr -c 0-11 -g FLOPS_DP  sleep 10
```

- **A frequent use is to measure a certain part of a long running application from outside**
- **likwid-perfctr also allows to specify arbitrary event sets on the command line:**

```
$ likwid-perfctr -c 0-12 -g \
INSTR_RETIRED_ANY:FIXC0,CPU_CLK_UNHALTED_CORE:FIXC1,\
FP_COMP_OPS_EXE_SSE_FP_PACKED:PMC0,\
UNC_L3_LINES_IN_ANY:UPMC0  sleep 10
```

- **It can also be used as cluster/server monitoring tool**

Cluster monitoring: Mflop/s protocol

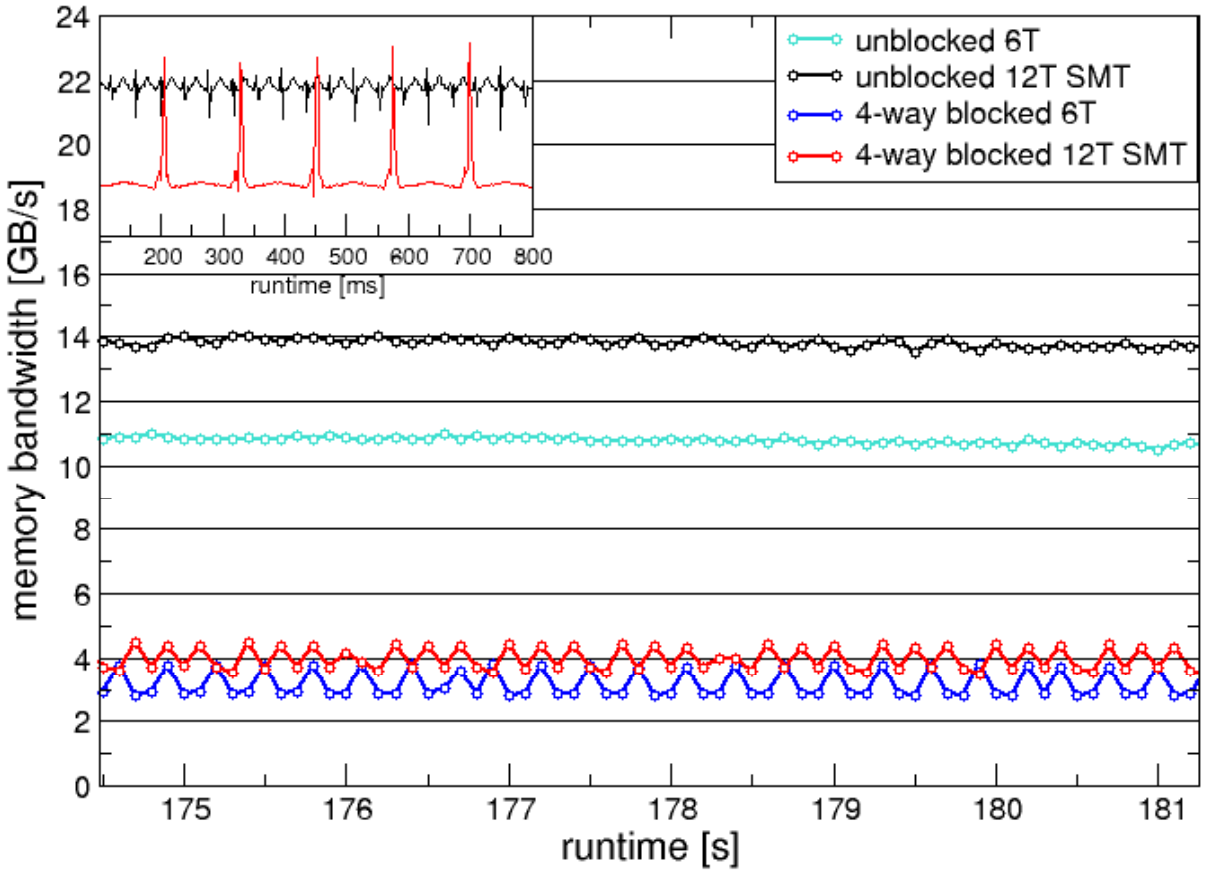


- Functionality of pin is now fully integrated in perfctr:

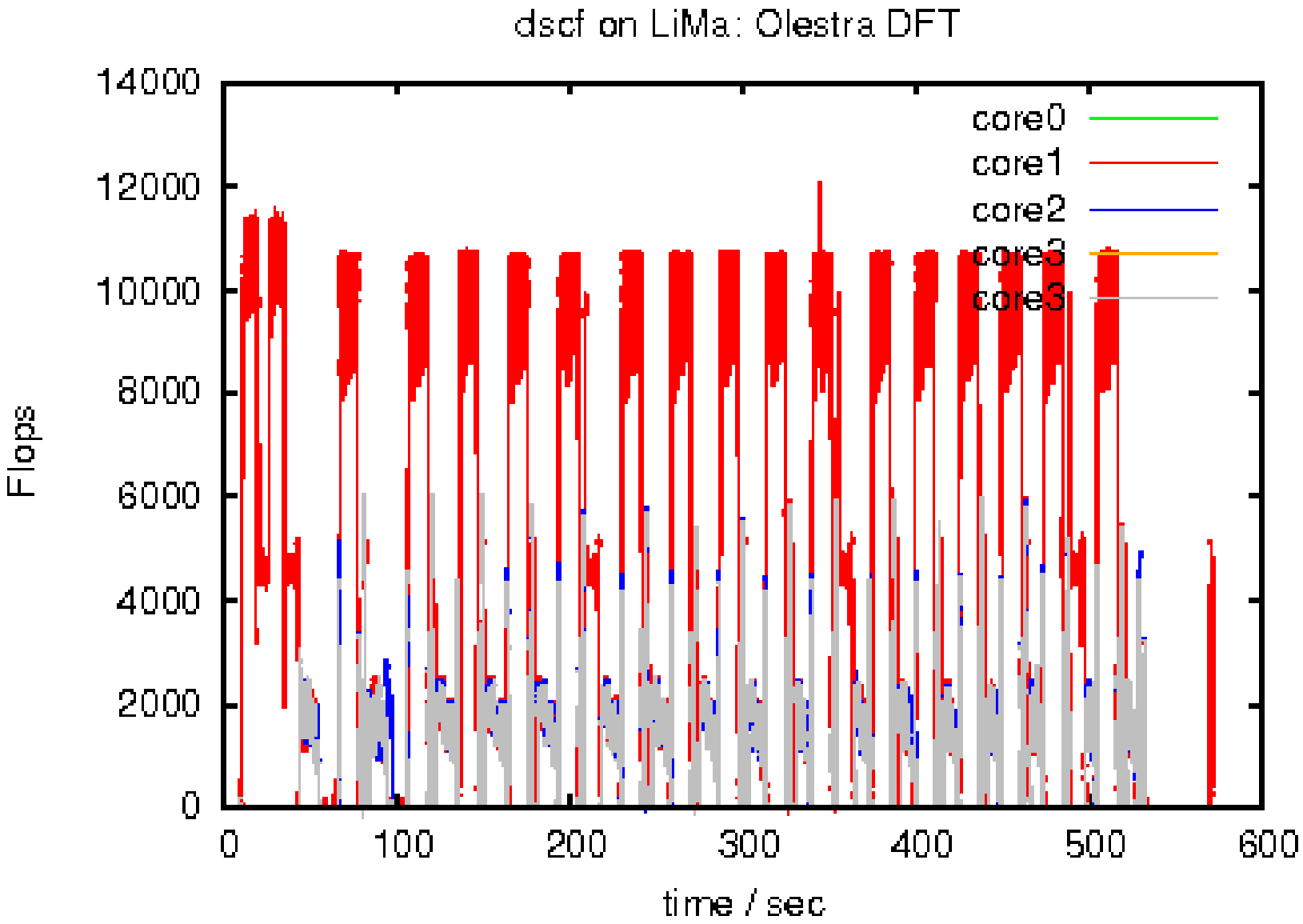
```
likwid-perfctr -C 0-12 -t intel -g FLOPS_DP  ./a.out
```

- Timeline mode:

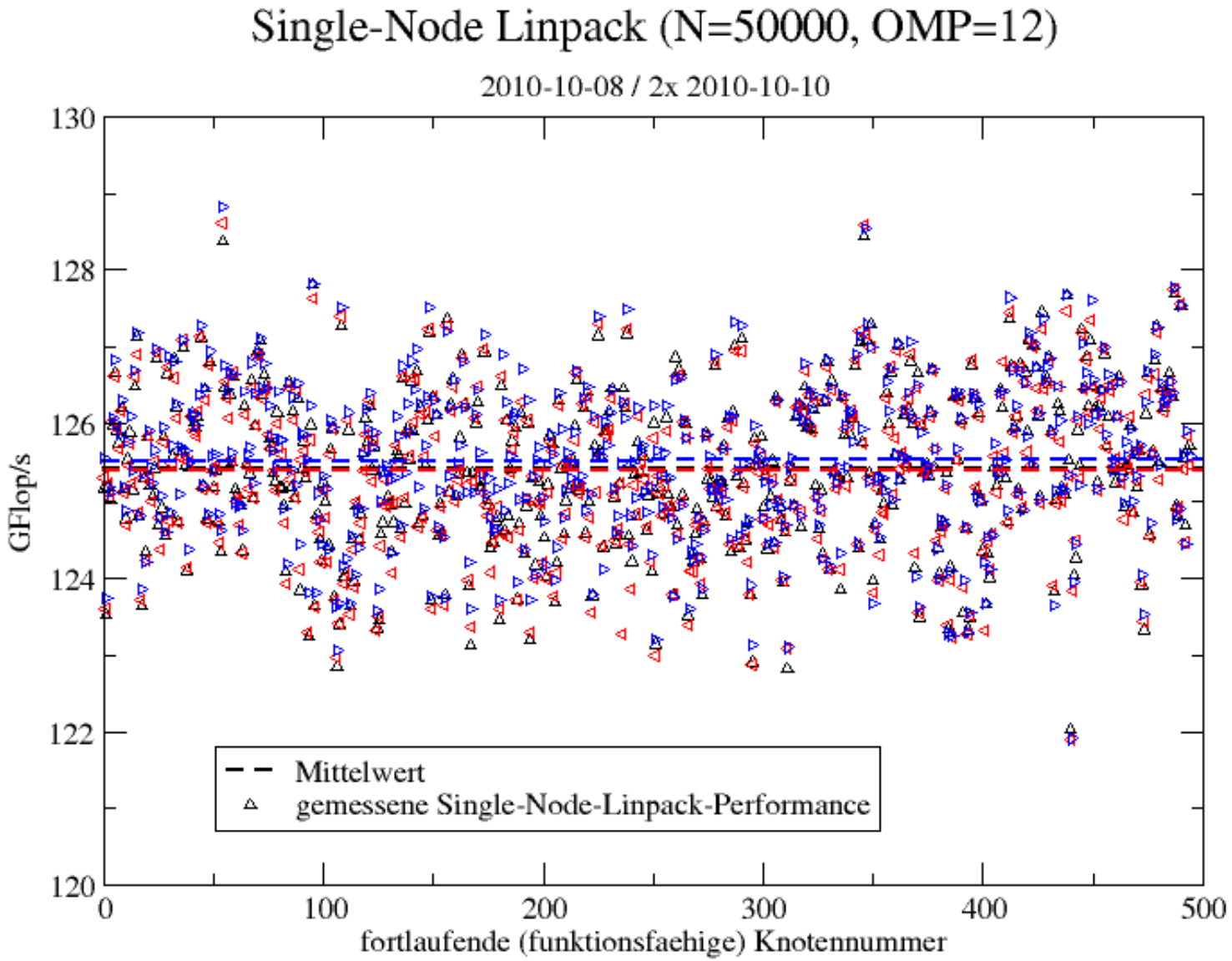
```
likwid-perfctr -c 0-12  -g MEM -d 50ms  > out.txt
```



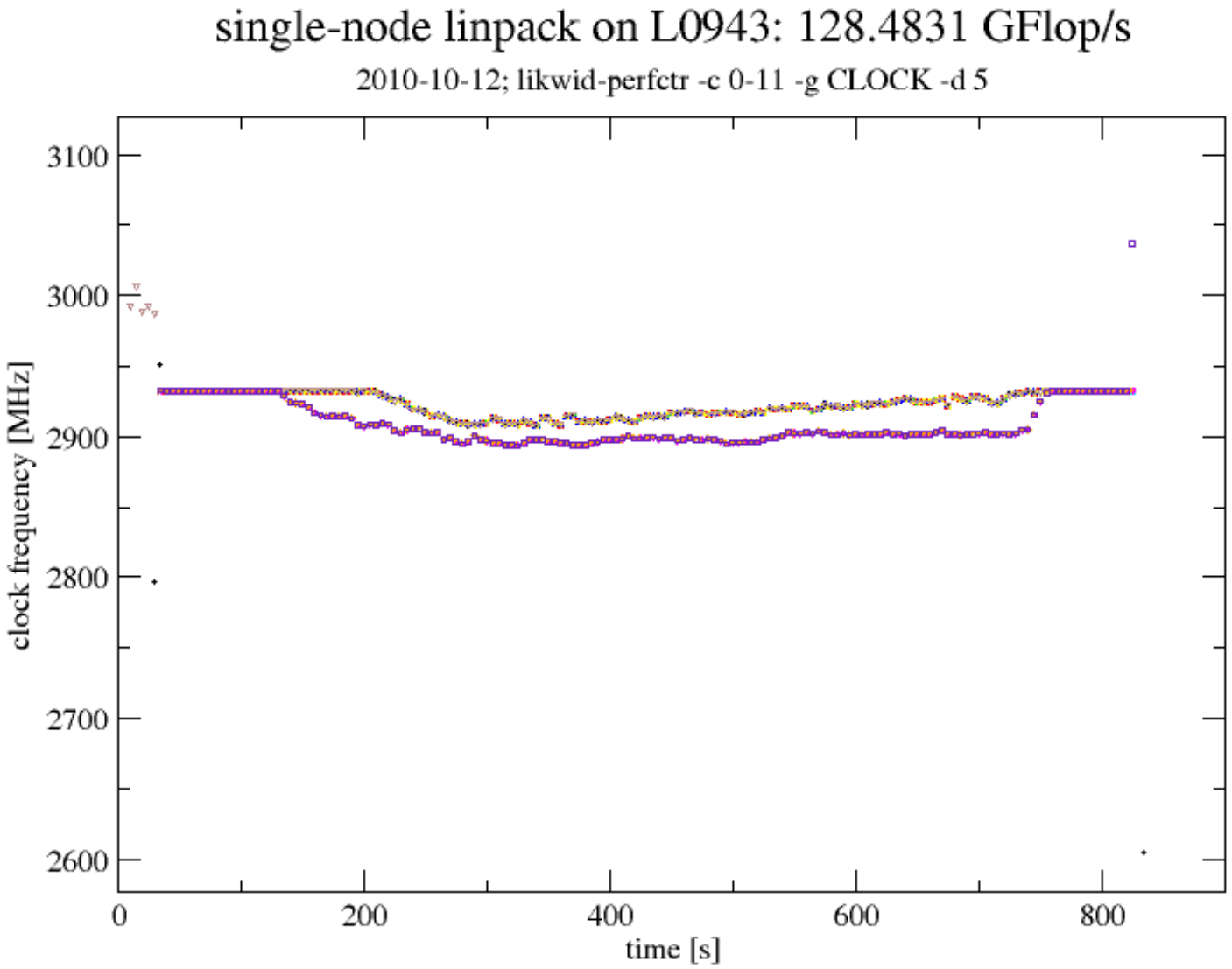
Likwid-perfctr and turbomole (MPI)



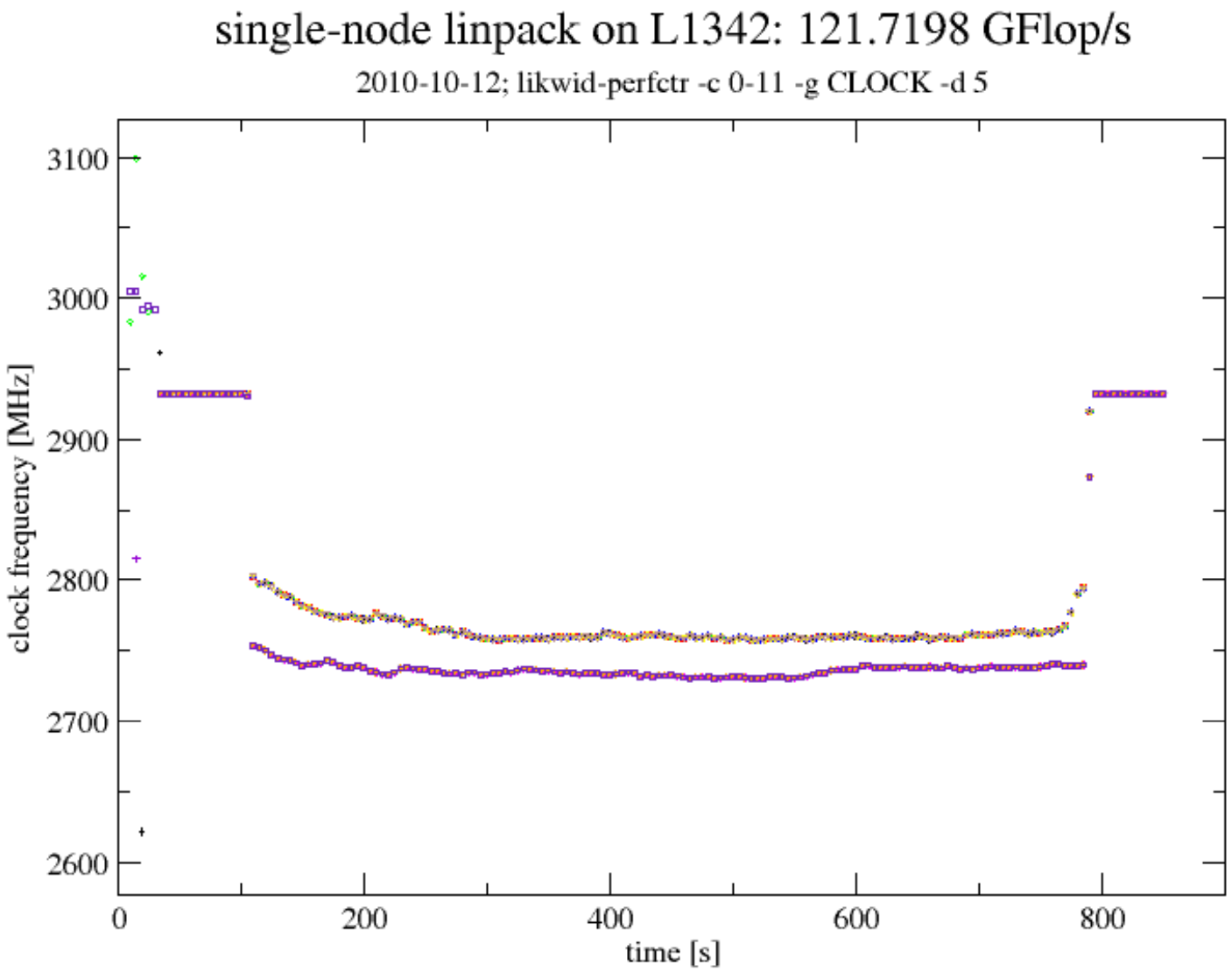
The dangers of Overclocking …

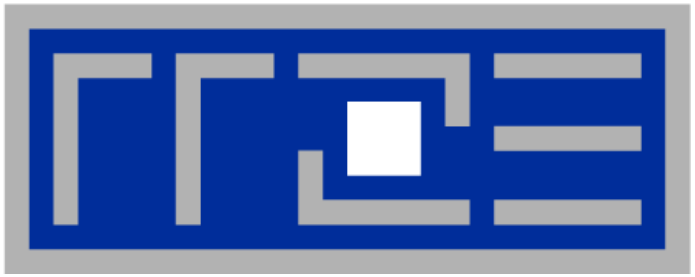


Turbo mode on Nehalem – fast node



Turbo mode on Nehalem – slow node





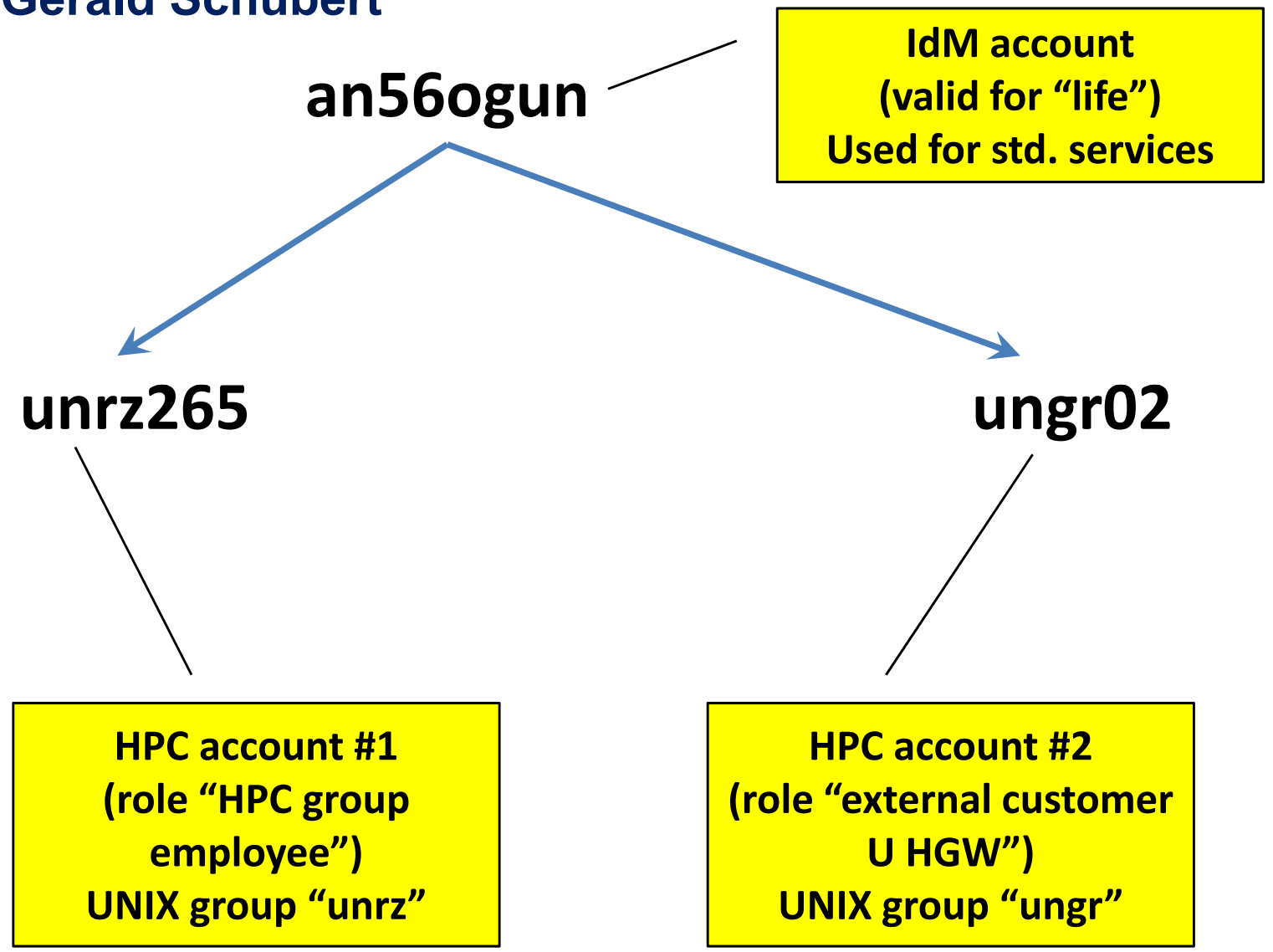
User management HPC@RRZE

HPC user management: Boundary conditions

- **RRZE systems IdM provides LDAP DB with all users and groups**
  - HPC users are members of a specific group
- **Direct access of every HPC node to central NIS/LDAP server is out of the question**
  - Severe performance and stability concerns
- **Every person at FAU has a unique "identity"**
  - Given by their "IdM" account
  - This account stays the same over the "lifetime" of this person at FAU
- **HPC accounting and resource management are incompatible with the "IdM account" concept**
- **Solution: HPC accounts are "associated" with IdM accounts and can be changed easily**
  - "Telling" names, consistent UNIX group memberships
  - More than one HPC account possible for "Servants of Two Masters"
  - This part of the identity could easily be pulled out of the IdM system and made its own subsystem, if required ;-)

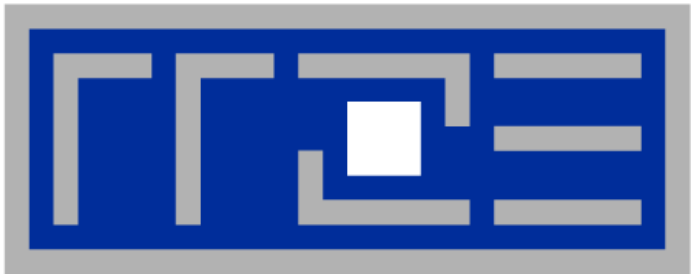
Example

- User: Dr. Gerald Schubert



How about credentials?

- **Cluster nodes**
  - `/etc/passwd` updated 2x per day on each cluster node from central (HPC) NIS server (`getent group infohpc` etc…)
  - NIS server updated regularly from IdM LDAP
- **File systems**
  - Home directories on all HPC file systems generated along with changes in passwd DB
  - Data from “old” users deleted after some waiting period
  - All file systems for HPC accounts are under the control of the HPC group
- **CPU time accounting**
  - As of now independent of all IdM functions
  - Future: Possible integration of user accounting data into IdM portal



Vielen Dank