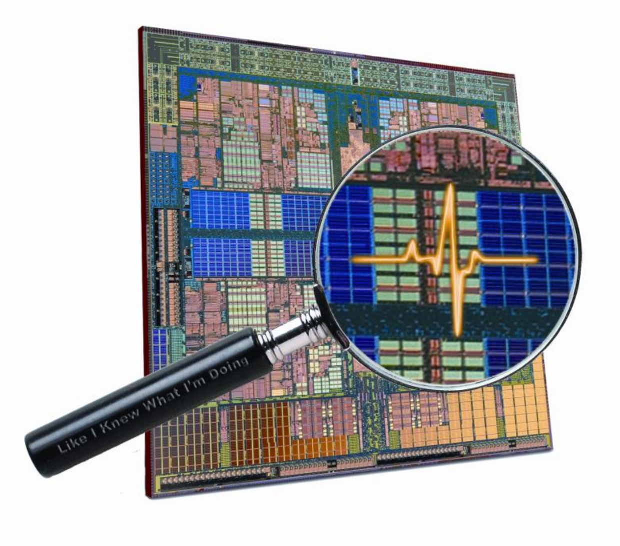


Neues von LIKWID

G. Hager, M. Meier, J. Treibig
RRZE

ZKI AK Supercomputing
15.03.2013

LIKWID – Lightweight Multicore Tools

- **Lightweight command line tools for Linux**
- **Help to face the challenges without getting in the way**
- **Focus on x86 architecture**
- **Philosophy:**
  - Simple
  - Efficient
  - Portable
  - Extensible

Open source project (GPL v2):
http://code.google.com/p/likwid/

LIKWID tools

`likwid-topology`	probe node topology
`likwid-pin`	enforce thread/core affinity
`likwid-mpirun`	startup wrapper script for MPI+X
`likwid-perfctr`	count performance events
`likwid-powermeter`	measure energy consumption
`likwid-memsweeper`	clear FS buffer cache on ccNUMA systems
`likwid-bench`	microbenchmarking framework

likwid-perfctr

- **Reads out processor performance counters**
  - similar to “perfex” on IRIX, “hpmcount” on AIX, “lipfpm” on Linux/Altix
- **A coarse overview of hardware performance monitoring data is often sufficient**
- **Implemented completely in user space (uses msr kernel module)**
  - Exception: Uncore events on Sandy Bridge
- **A small proxy application managing a controlled access to the MSR device files is available**
- **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 ./a.out
```

- **Preconfigured and extensible metric groups, list with `likwid-perfctr -a`** →

```
BRANCH: Branch prediction miss rate/ratio
CACHE: Data cache miss rate/ratio
CLOCK: Clock of cores
DATA: Load to store ratio
FLOPS_DP: Double Precision MFlops/s
FLOPS_SP: Single Precision MFlops/s
FLOPS_X87: X87 MFlops/s
L2: L2 cache bandwidth in MBytes/s
L2CACHE: L2 cache miss rate/ratio
L3: L3 cache bandwidth in MBytes/s
L3CACHE: L3 cache miss rate/ratio
MEM: Main memory bandwidth in MBytes/s
TLB: TLB miss rate/ratio
```

- **Supported processors:**
  - Intel Core 2
  - Intel Nehalem / Westmere (all variants) supporting Uncore events
  - Intel NehalemEX/WestmereEX (with Uncore)
  - Intel Sandy Bridge
  - AMD K8/K10
  - AMD Interlagos
- **Most popular usage: Wrapper mode – provides Simple end-to-end measurement of hardware performance metrics**

likwid-perfctr

Example usage for Wrapper mode

```
$ env OMP_NUM_THREADS=4 likwid-perfctr -C N:0-3 -g FLOPS_DP  ./stream.exe
-------------------------------------------------------------
CPU type:       Intel Core Lynnfield processor
CPU clock:      2.93 GHz
-------------------------------------------------------------
Measuring group FLOPS_DP
-------------------------------------------------------------
YOUR PROGRAM OUTPUT
+--------------------------------------+-------------+-------------+-------------+-------------+
|                 Event                |   core 0    |   core 1    |   core 2    |   core 3    |
+--------------------------------------+-------------+-------------+-------------+-------------+
|          INSTR_RETIRED_ANY           | 1.97463e+08 | 2.31001e+08 | 2.30963e+08 | 2.31885e+08 |
|        CPU_CLK_UNHALTED_CORE         | 9.56999e+08 | 9.58401e+08 | 9.58637e+08 | 9.57338e+08 |
|      FP_COMP_OPS_EXE_SSE_FP_PACKED   | 4.00294e+07 | 3.08927e+07 | 3.08866e+07 | 3.08904e+07 |
|      FP_COMP_OPS_EXE_SSE_FP_SCALAR   |     882     |      0      |      0      |      0      |
| FP_COMP_OPS_EXE_SSE_SINGLE_PRECISION |      0      |      0      |      0      |      0      |
| FP_COMP_OPS_EXE_SSE_DOUBLE_PRECISION | 4.00303e+07 | 3.08927e+07 | 3.08866e+07 | 3.08904e+07 |
+--------------------------------------+-------------+-------------+-------------+-------------+
+---------------------------+------------+----------+----------+----------+
|          Metric           |   core 0   |  core 1  |  core 2  |  core 3  |
+---------------------------+------------+----------+----------+----------+
|        Runtime [s]        |  0.326242  | 0.32672  | 0.326801 | 0.326358 |
|            CPI            |  4.84647   | 4.14891  | 4.15061  | 4.12849  |
| DP MFlops/s (DP assumed)  |  245.399   | 189.108  | 189.024  | 189.304  |
|      Packed MUOPS/s       |  122.698   | 94.554   | 94.5121  | 94.6519  |
|      Scalar MUOPS/s       | 0.00270351 |    0     |    0     |    0     |
|        SP MUOPS/s         |     0      |    0     |    0     |    0     |
|        DP MUOPS/s         |  122.701   | 94.554   | 94.5121  | 94.6519  |
+---------------------------+------------+----------+----------+----------+
```

Always measured

Configured metrics (this group)

Derived metrics

- **Restrict counting to parts of an application**
- **The API only turns counters on/off. The configuration of the counters is still done by likwid-perfctr**
- **Multiple named regions can be measured**
- **Results on multiple calls are accumulated**
- **Inclusive and overlapping Regions are allowed**

```
likwid_markerInit();  // must be called from serial region

likwid_markerStartRegion("Compute");
. . .
likwid_markerStopRegion("Compute");


likwid_markerStartRegion("postprocess");
. . .
likwid_markerStopRegion("postprocess");


likwid_markerClose();  // must be called from serial region
```

- **likwid-perfctr measures on a core basis and has no notion what runs on the cores**

  This enables to listen on what currently happens on the machine:

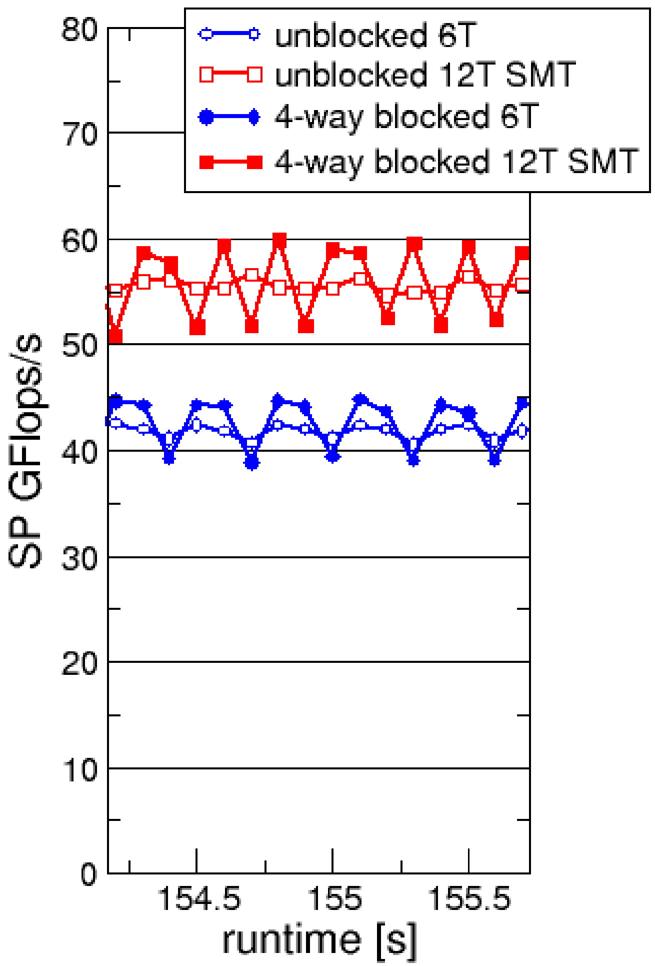
  ```
  $ likwid-perfctr -c N:0-11 -g FLOPS_DP  -s 10
  ```

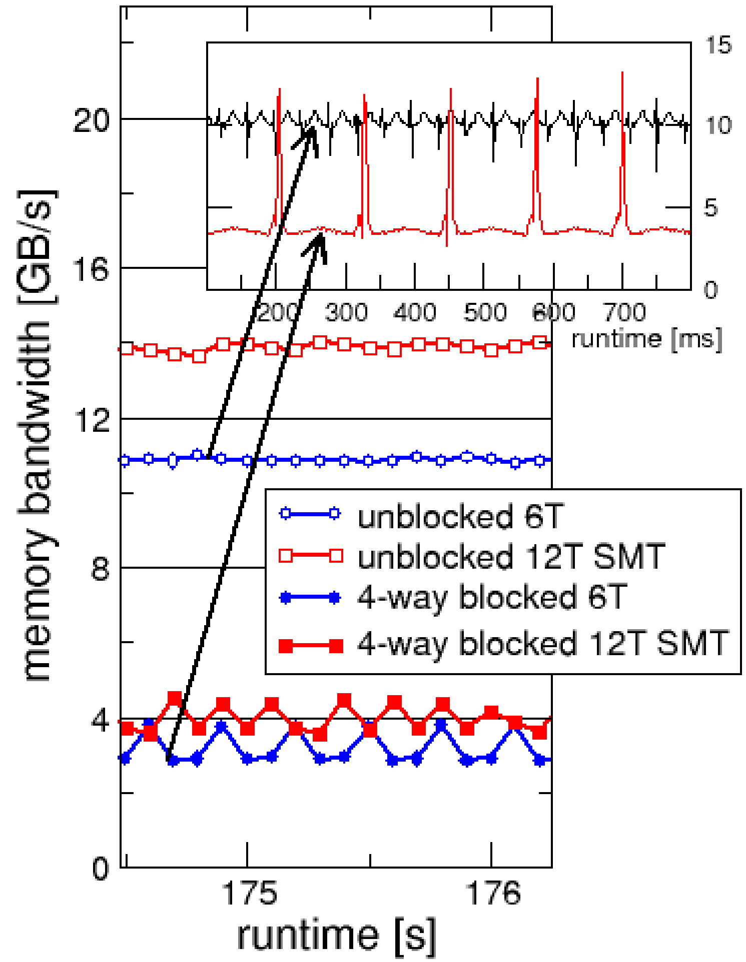
- **Can be used to measure a certain part of a long running parallel application from outside**

- **Can also be used as cluster/server monitoring tool**
  - Yay, nice graphs!
  - Can spot some obvious problems such as wrong pinning

- **Time-resolved measurements of full node:**

```
likwid-perfctr -c N:0-11 -g MEM -d 50ms  > out.txt
```

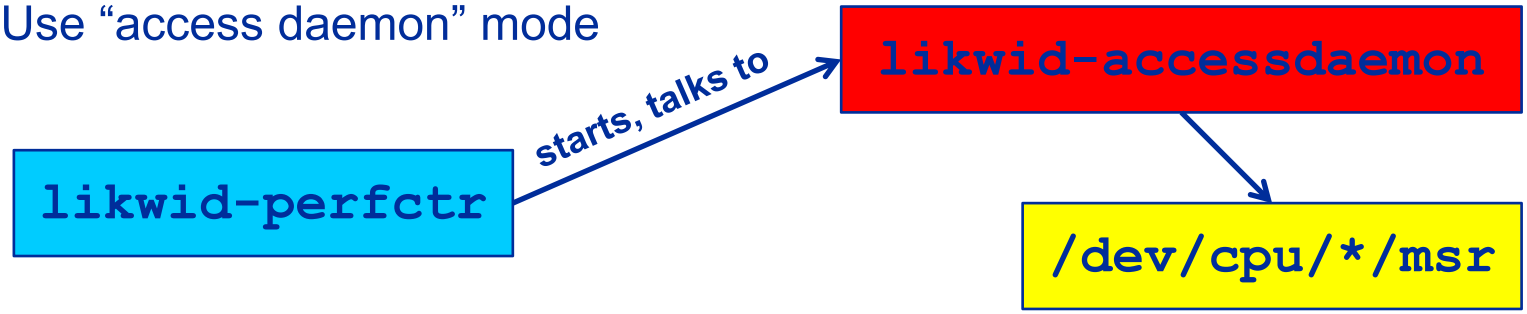




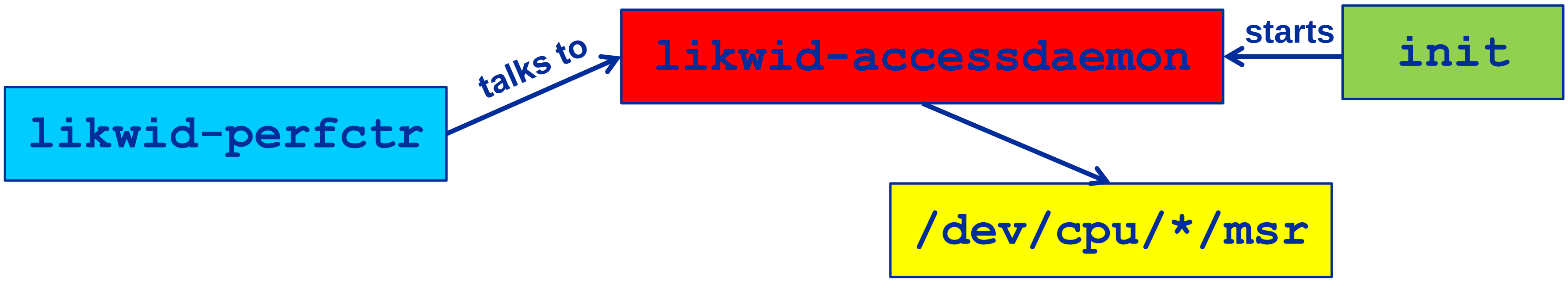
Access rights for likwid-perfctr

- **Options for access to /dev/cpu/*/msr**
  1. Grant direct access (UNIX permissions)

  2. Use “access daemon” mode



  3. Use “system daemon” mode



Concurrent use

- **Only one "user" can use processor performance counters at a time**
- **Collisions between user's measurements and cluster monitoring!**

- **Solution: system daemon mode**
  - Originally developed to improve security by filtering MSR accesses from users – running set(u|g)id.
    - Clients can mark themselves as "low priority"
    - "low priority" clients get dropped when a new client connects, normal clients don't.
    - Cluster monitoring runs "low priority", thus leaving the MSRs to the user if he wishes to use them.

LIKWID in monitoring

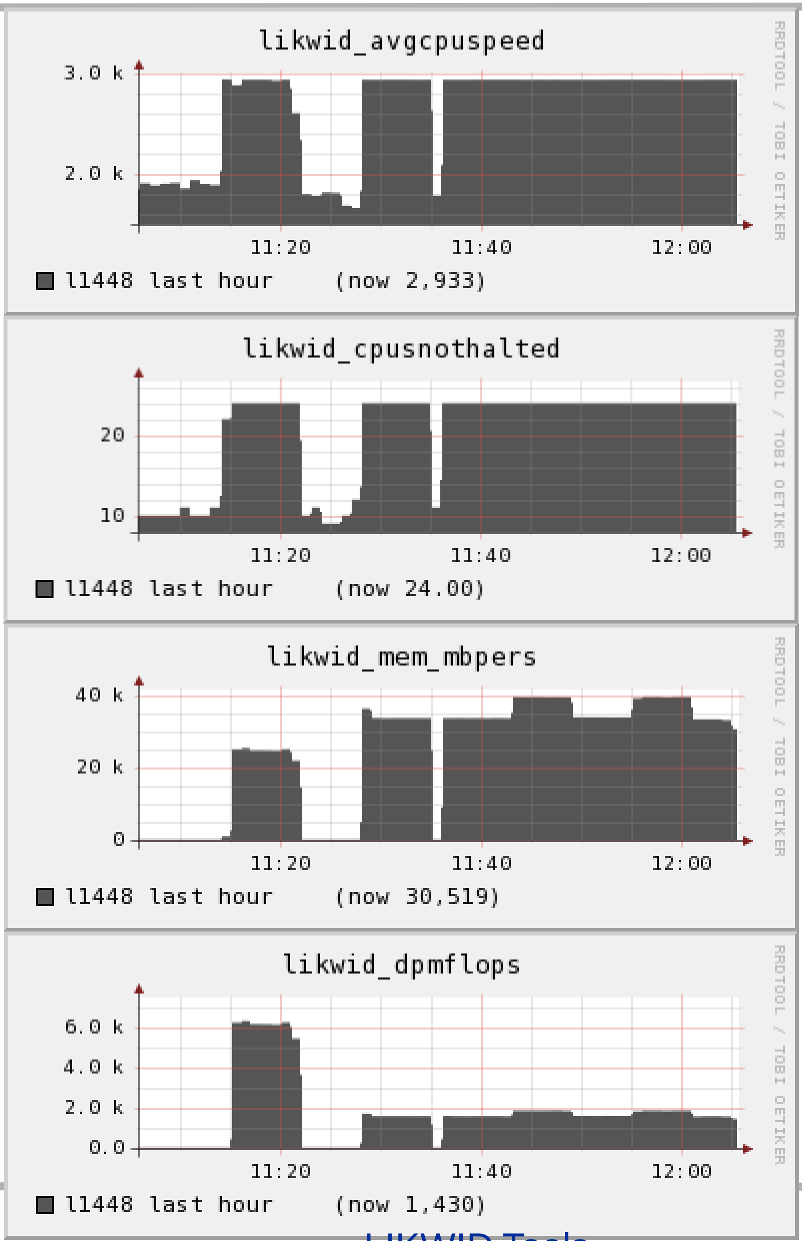
Nice graphs for our webpage

Explain the fluctuations?

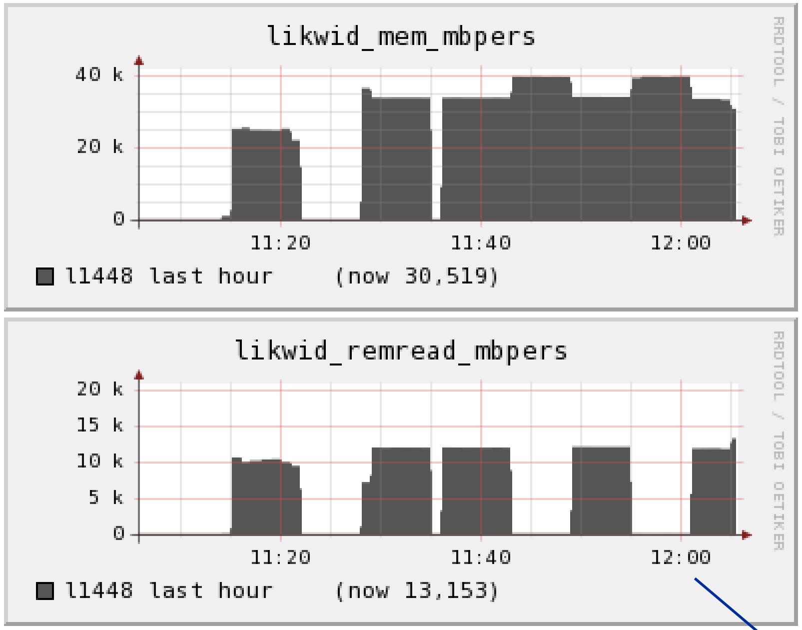
- **Stream compiled with NTIMES=2000 (i.e. a run takes >5 minutes), run in an endless loop.**
- **Performance varies:**

```
Function      Rate (MB/s)   Avg time     Min time     Max time
Copy:        27162.9823     0.0473       0.0471       0.0586
Scale:       40862.1095     0.0315       0.0313       0.0457
Add:         39729.3692     0.0485       0.0483       0.0622
Triad:       40515.7055     0.0476       0.0474       0.0554

Function      Rate (MB/s)   Avg time     Min time     Max time
Copy:        24374.0852     0.0527       0.0525       0.0604
Scale:       30375.6230     0.0422       0.0421       0.0565
Add:         35139.6704     0.0548       0.0546       0.0654
Triad:       35491.0609     0.0543       0.0541       0.2115

Function      Rate (MB/s)   Avg time     Min time     Max time
Copy:        27183.1997     0.0472       0.0471       0.0574
Scale:       40871.1307     0.0315       0.0313       0.0393
Add:         39773.7153     0.0484       0.0483       0.0647
Triad:       40575.5182     0.0475       0.0473       0.0580
```

Explain the fluctuations?



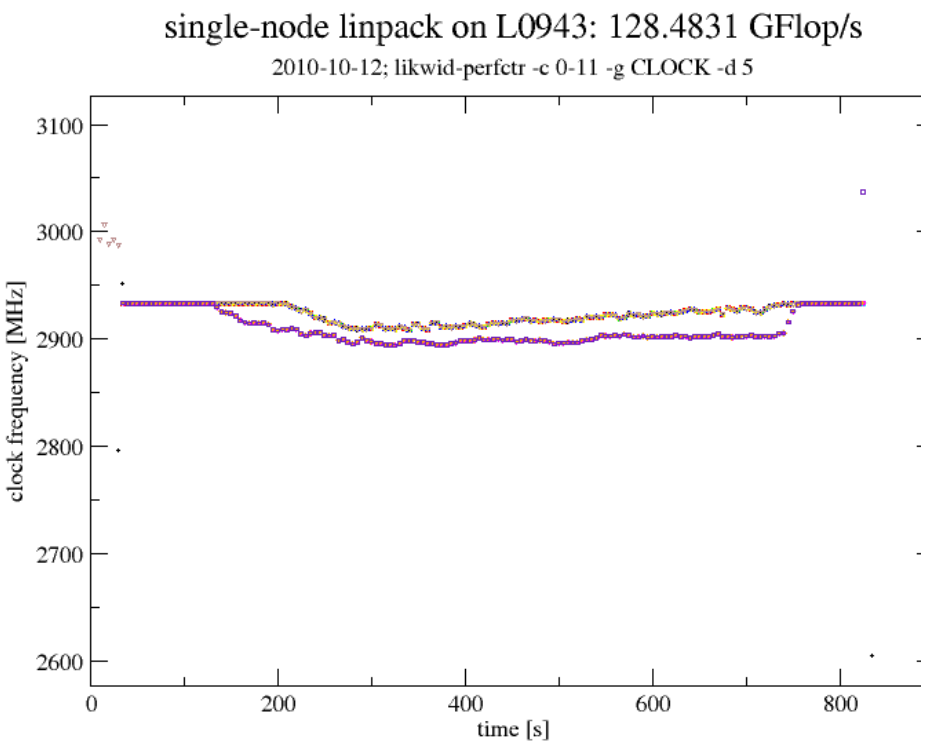
The reason

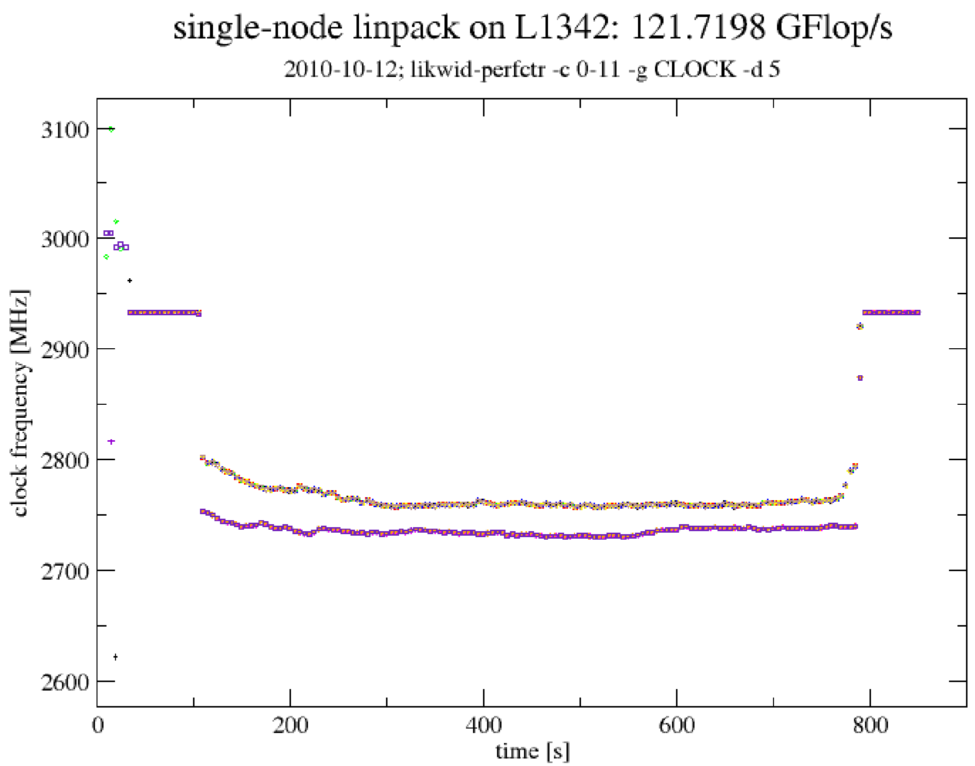


QuickPath remote reads → NUMA problem!

Turbo mode on Westmere

Turbo mode causes volatile performance values





Measuring energy consumption

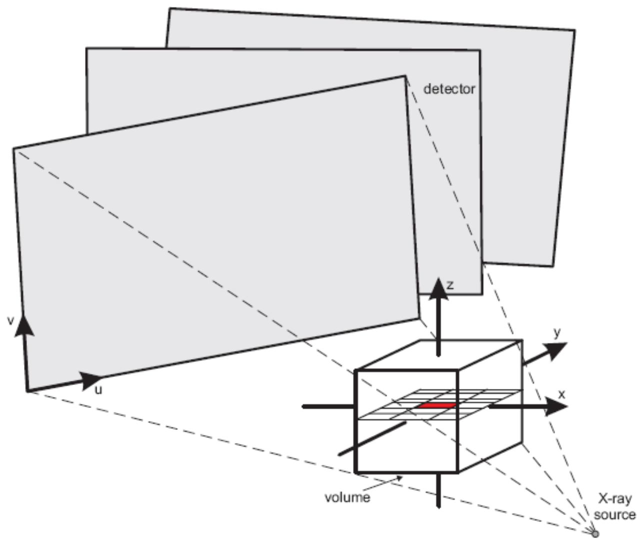
likwid-powermeter

- **Implements Intel RAPL interface (Sandy Bridge)**
- **RAPL = “Running Average Power Limit”**

```
-------------------------------------------------------------
CPU name:        Intel Core SandyBridge processor
CPU clock:       3.49 GHz
-------------------------------------------------------------
Base clock:      3500.00 MHz
Minimal clock:   1600.00 MHz
Turbo Boost Steps:
C1 3900.00 MHz
C2 3800.00 MHz
C3 3700.00 MHz
C4 3600.00 MHz
-------------------------------------------------------------
Thermal Spec Power: 95 Watts
Minimum  Power: 20 Watts
Maximum  Power: 95 Watts
Maximum  Time Window: 0.15625 micro sec
-------------------------------------------------------------
```

Example:

A medical image reconstruction code on Sandy Bridge



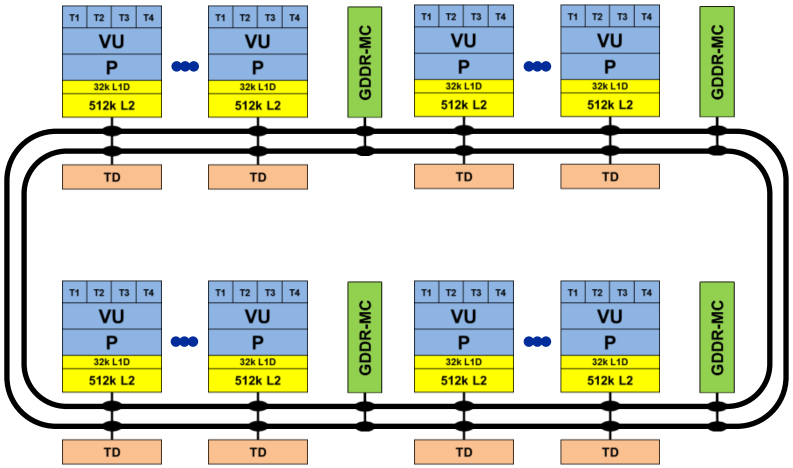
Sandy Bridge EP (8 cores, 2.7 GHz base freq.)

Test case	Runtime [s]	Power [W]	Energy [J]
8 cores, plain C	**90.43**	90	8110
8 cores, SSE	29.63	93	2750
8 cores (SMT), SSE	22.61	102	2300
8 cores (SMT), AVX	**18.42**	111	2040

Faster code ➜ less energy

LIKWID current state, problems and outlook

- **Intel Sandy Bridge**
  - Uncore supported, but no remote memory access metrics (?)
  - Very complex PCIe-based Uncore counter access
  - Unreliable FLOP counting
- **Intel Ivy Bridge**
  - Works, but same limitations as for Sandy Bridge
- **Intel Xeon Phi**
  - Basic support
  - Metric groups under development
  - Uncore still unsupported



- **Under developent**
  - Extended topology support (scatter, compact, strided)
  - Marker API + system daemon mode

Thank you.

http://code.google.com/p/likwid/