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/
<table>
<thead>
<tr>
<th>LIKWID tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>likwid-topology</td>
</tr>
<tr>
<td>likwid-pin</td>
</tr>
<tr>
<td>likwid-mpirun</td>
</tr>
<tr>
<td>likwid-perfctr</td>
</tr>
<tr>
<td>likwid-powermeter</td>
</tr>
<tr>
<td>likwid-memsweeper</td>
</tr>
<tr>
<td>on ccNUMA systems</td>
</tr>
<tr>
<td>likwid-bench</td>
</tr>
</tbody>
</table>
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
  ```
### likwid-perfctr

More information

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

- **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

<table>
<thead>
<tr>
<th>Metric Group</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BRANCH</td>
<td>Branch prediction miss rate/ratio</td>
</tr>
<tr>
<td>CACHE</td>
<td>Data cache miss rate/ratio</td>
</tr>
<tr>
<td>CLOCK</td>
<td>Clock of cores</td>
</tr>
<tr>
<td>DATA</td>
<td>Load to store ratio</td>
</tr>
<tr>
<td>FLOPS_DP</td>
<td>Double Precision MFlops/s</td>
</tr>
<tr>
<td>FLOPS_SP</td>
<td>Single Precision MFlops/s</td>
</tr>
<tr>
<td>FLOPS_X87</td>
<td>X87 MFlops/s</td>
</tr>
<tr>
<td>L2</td>
<td>L2 cache bandwidth in MBytes/s</td>
</tr>
<tr>
<td>L2CACHE</td>
<td>L2 cache miss rate/ratio</td>
</tr>
<tr>
<td>L3</td>
<td>L3 cache bandwidth in MBytes/s</td>
</tr>
<tr>
<td>L3CACHE</td>
<td>L3 cache miss rate/ratio</td>
</tr>
<tr>
<td>MEM</td>
<td>Main memory bandwidth in MBytes/s</td>
</tr>
<tr>
<td>TLB</td>
<td>TLB miss rate/ratio</td>
</tr>
</tbody>
</table>
**Example usage for Wrapper mode**

```bash
$ 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**

<table>
<thead>
<tr>
<th>Event</th>
<th>core 0</th>
<th>core 1</th>
<th>core 2</th>
<th>core 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>INSTR_RETIRED_ANY</td>
<td>1.97463e+08</td>
<td>2.31001e+08</td>
<td>2.30963e+08</td>
<td>2.31885e+08</td>
</tr>
<tr>
<td>CPU_CLK_UNHALTED_CORE</td>
<td>9.56999e+08</td>
<td>9.58401e+08</td>
<td>9.58637e+08</td>
<td>9.57338e+08</td>
</tr>
<tr>
<td>FP_COMP_OPS_EXE_SSE_FP_PACKED</td>
<td>4.00294e+07</td>
<td>3.08927e+07</td>
<td>3.08866e+07</td>
<td>3.08904e+07</td>
</tr>
<tr>
<td>FP_COMP_OPS_EXE_SSE_FP_SCALAR</td>
<td>882</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>FP_COMP_OPS_EXE_SSE_SINGLE_PRECISION</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>FP_COMP_OPS_EXE_SSE_DOUBLE_PRECISION</td>
<td>4.00303e+07</td>
<td>3.08927e+07</td>
<td>3.08866e+07</td>
<td>3.08904e+07</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Metric</th>
<th>core 0</th>
<th>core 1</th>
<th>core 2</th>
<th>core 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Runtime [s]</td>
<td>0.326242</td>
<td>0.32672</td>
<td>0.326801</td>
<td>0.326358</td>
</tr>
<tr>
<td>CPI</td>
<td>4.84647</td>
<td>4.14891</td>
<td>4.15061</td>
<td>4.12849</td>
</tr>
<tr>
<td>DP MFlops/s (DP assumed)</td>
<td>245.399</td>
<td>189.108</td>
<td>189.024</td>
<td>189.304</td>
</tr>
<tr>
<td>Packed MUOPS/s</td>
<td>122.698</td>
<td>94.554</td>
<td>94.5121</td>
<td>94.6519</td>
</tr>
<tr>
<td>Scalar MUOPS/s</td>
<td>0.00270351</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SP MUOPS/s</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>DP MUOPS/s</td>
<td>122.701</td>
<td>94.554</td>
<td>94.5121</td>
<td>94.6519</td>
</tr>
</tbody>
</table>

---

**Derived metrics**

**Always measured**

**Configured metrics (this group)**
likwid-perfctr

Marker API

- 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

```c
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

Stethoscope mode

- 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
**likwid-perfctr**

*Timeline mode*

- **Time-resolved measurements of full node:**
  
  ```
  likwid-perfctr -c N:0-11 -g MEM -d 50ms > out.txt
  ```

---

**Timeline graphs**

- **CPU GFlops/s**
  - unblocked 6T
  - unblocked 12T SMT
  - 4-way blocked 6T
  - 4-way blocked 12T SMT

- **Memory bandwidth [GB/s]**
  - unblocked 6T
  - unblocked 12T SMT
  - 4-way blocked 6T
  - 4-way blocked 12T SMT
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

[Diagram showing the relationships between likwid-perfctr, likwid-accessdaemon, /dev/cpu/*/msr, and init.]
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.

15.03.2013  LIKWID Tools
LIKWID in monitoring
Nice graphs for our webpage

15.03.2013

LIKWID Tools
Explain the fluctuations?

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Rate (MB/s)</th>
<th>Avg time</th>
<th>Min time</th>
<th>Max time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy:</td>
<td>27162.9823</td>
<td>0.0473</td>
<td>0.0471</td>
<td>0.0586</td>
</tr>
<tr>
<td>Scale:</td>
<td>40862.1095</td>
<td>0.0315</td>
<td>0.0313</td>
<td>0.0457</td>
</tr>
<tr>
<td>Add:</td>
<td>39729.3692</td>
<td>0.0485</td>
<td>0.0483</td>
<td>0.0622</td>
</tr>
<tr>
<td>Triad:</td>
<td>40515.7055</td>
<td>0.0476</td>
<td>0.0474</td>
<td>0.0554</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function</th>
<th>Rate (MB/s)</th>
<th>Avg time</th>
<th>Min time</th>
<th>Max time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy:</td>
<td>24374.0852</td>
<td>0.0527</td>
<td>0.0525</td>
<td>0.0604</td>
</tr>
<tr>
<td>Scale:</td>
<td>30375.6230</td>
<td>0.0422</td>
<td>0.0421</td>
<td>0.0565</td>
</tr>
<tr>
<td>Add:</td>
<td>35139.6704</td>
<td>0.0548</td>
<td>0.0546</td>
<td>0.0647</td>
</tr>
<tr>
<td>Triad:</td>
<td>35491.0609</td>
<td>0.0543</td>
<td>0.0541</td>
<td>0.2115</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function</th>
<th>Rate (MB/s)</th>
<th>Avg time</th>
<th>Min time</th>
<th>Max time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy:</td>
<td>27183.1997</td>
<td>0.0472</td>
<td>0.0471</td>
<td>0.0574</td>
</tr>
<tr>
<td>Scale:</td>
<td>40871.1307</td>
<td>0.0315</td>
<td>0.0313</td>
<td>0.0393</td>
</tr>
<tr>
<td>Add:</td>
<td>39773.7153</td>
<td>0.0484</td>
<td>0.0483</td>
<td>0.0647</td>
</tr>
<tr>
<td>Triad:</td>
<td>40575.5182</td>
<td>0.0475</td>
<td>0.0473</td>
<td>0.0580</td>
</tr>
</tbody>
</table>

15.03.2013 LIKWID Tools
Explain the fluctuations?
The reason

QuickPath remote reads \( \rightarrow \) NUMA problem!
Turbo mode on Westmere

Turbo mode causes volatile performance values

Single-node Linpack on L0943: 128.4831 GFlop/s
2010-10-12; likwid-perfctr -c 0-11 -g CLOCK -d 5

Single-node Linpack on L1342: 121.7198 GFlop/s
2010-10-12; likwid-perfctr -c 0-11 -g CLOCK -d 5
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

---

![Image of medical image reconstruction](image.jpg)

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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>8 cores, plain C</td>
<td>90.43</td>
<td>90</td>
<td>8110</td>
</tr>
<tr>
<td>8 cores, SSE</td>
<td>29.63</td>
<td>93</td>
<td>2750</td>
</tr>
<tr>
<td>8 cores (SMT), SSE</td>
<td>22.61</td>
<td>102</td>
<td>2300</td>
</tr>
<tr>
<td>8 cores (SMT), AVX</td>
<td>18.42</td>
<td>111</td>
<td>2040</td>
</tr>
</tbody>
</table>

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

15.03.2013 LIKWID Tools
Thank you.

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