The Importability of Performance Tools: The Good, the Bad, and the Ugly of PMUs

Jeanine Cook
Scalable Architectures
SNL ABQ

COE Phoenix April, 2016
Performance Tools

- Mostly talking about non-commercial tools
  - Tools that lots of us hack up to look at performance issues
  - MPI and node data
    - Interfaces like PAPI and mpip
  - Hoping not to have to learn intricacies of complicated performance tool
  - Tool doesn’t quite do what we want it to
PMUs and Performance Counters (1)

- PMUs on CPUs are all different
  - Implement different events
    - “Architected” PMUs provide subset of base events across model lines
  - Names of identical events can differ
  - Different number of physical counters
    - Divided amongst threads
  - Significant errata in PMU behavior
    - But info often not propagated to end user

- Event mapping tables
  - Connects event name and qualifiers to hardware register(s) configuration(s)
  - Some change frequently (Intel’s https://download.01.org/perfmon/)
  - Some hardly get updated or are community supported (papi_presets)
  - Kernel contains some events with compile time mappings (perf list)
    - If hardware requires a change, no way to update map, either can’t access event or may return strange results

COE Phoenix April, 2016
PMUs and Performance Counters (2)

- Documentation is spotty at best, inaccurate at worst
  - Hard to really be sure of precisely what you’re counting without investigation, sometimes microbenchmarking
    - EX: Events commonly counted at issue, so many overcount due to speculation and re-issue (e.g., SNB and IVB FP events)
PAPI Presets vs Native Events

- Presets offer portability... sort of
  - Event names consistent across platforms, but may map to different underlying native events, as best effort
    - Particular event may not actually count same behavior across systems
    - Particular event on single system may not count what event name reflects
      - SNB: PAPI_DP_OPS – Counts DP FP add and multiply, but not divide instructions (separate counter for that)
        » Use papi_decode --a to see what presets count
        » Decode libpfm events using showevtinfo from libpfm
Takeaways

- If using performance counters directly
  - Don’t use presets, use natives
  - Be sure you know what you’re counting
- Performance tools (commercial and home-grown) access PMUs
  - Restricts cross-platform usage

An attempt to make PMU/performance counter use easier and cross-platform…
Perfminer

Property of MinimalMetrics and SNL
Perfminer: Performance Analysis

- Performance Analysis
  - First line support tool for performance investigations
  - Low-overhead, scalable with simple instrumentation
  - “Inside-out” performance metrics from CPU, Memory, Network, Filesystem, Power, with localizing information
    - Data is per-thread and aggregated to all levels
  - Immediate, efficient, visual performance feedback via a web page
    - Drill down methodology with click through to investigate further
      - Supports using other tools
  - Enables continuous performance regression of system and applications
  - Actionable feedback (future development)
    - Suggestions for further testing; automatic generation of test scripts for Perfminer and potentially other tools
Perfminer: Performance Analysis

- Performance Analysis
  - Modular design
    - Collector, front-end, back-end
      - Java web-based, front-end
      - Python back end,
      - Schema-less NoSQL data store.
        - Data is easily accessible for analytics via Excel, R, Matlab, Python, etc
      - Extensible analytics with Python
      - Elegant visualization using D3
  - No recompilation required
  - Easy to install (few dependencies) and trivial to use.
  - Full batch system integration (coming soon)
Perfminer: Performance as a Service

- Continuous monitoring infrastructure
  - Can opt in or out
- Deliver a report through email with actionable feedback
### Job Info

#### Raw Data

<table>
<thead>
<tr>
<th>Job ID</th>
<th>Job Number</th>
<th>Owner</th>
<th>Nodes</th>
<th>Binary</th>
<th>Running Time</th>
<th>Finish Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOJY389439</td>
<td>1025821</td>
<td>pjmucci</td>
<td>4</td>
<td>miniaero.exe</td>
<td>0:00:23</td>
<td>2016-03-20 10:05:25</td>
</tr>
<tr>
<td>XIAT133711</td>
<td>1025820</td>
<td>pjmucci</td>
<td>1</td>
<td>stream-triad-omp-1M</td>
<td>0:00:46</td>
<td>2016-03-20 10:05:48</td>
</tr>
<tr>
<td>CMNG317917</td>
<td>1025818</td>
<td>pjmucci</td>
<td>10</td>
<td>miniaero.exe</td>
<td>0:00:01</td>
<td>2016-03-20 10:09:22</td>
</tr>
<tr>
<td>UQFM259478</td>
<td>1025819</td>
<td>pjmucci</td>
<td>1</td>
<td>stencil-27pt-50</td>
<td>0:00:03</td>
<td>2016-03-20 10:05:05</td>
</tr>
<tr>
<td>OXBX917360</td>
<td>1025817</td>
<td>pjmucci</td>
<td>3</td>
<td>miniaero.exe</td>
<td>0:00:06</td>
<td>2016-03-20 10:04:42</td>
</tr>
<tr>
<td>URGM160184</td>
<td>1025816</td>
<td>pjmucci</td>
<td>1</td>
<td>stream-copy-omp-1M</td>
<td>0:01:05</td>
<td>2016-03-20 10:05:41</td>
</tr>
<tr>
<td>AYKC210492</td>
<td>1025815</td>
<td>pjmucci</td>
<td>1</td>
<td>miniaero.exe</td>
<td>0:00:04</td>
<td>2016-03-20 10:04:40</td>
</tr>
<tr>
<td>NGOJ726228</td>
<td>1025814</td>
<td>pjmucci</td>
<td>1</td>
<td>stream-copy-omp-2G</td>
<td>0:02:39</td>
<td>2016-03-20 10:07:15</td>
</tr>
<tr>
<td>WOPI548517</td>
<td>1025813</td>
<td>pjmucci</td>
<td>1</td>
<td>stream-triad-1M</td>
<td>0:00:46</td>
<td>2016-03-20 10:05:21</td>
</tr>
<tr>
<td>FJGL475723</td>
<td>1025812</td>
<td>pjmucci</td>
<td>8</td>
<td>luash-opt</td>
<td>0:04:27</td>
<td>2016-03-20 10:08:03</td>
</tr>
</tbody>
</table>

**JOB DETAILS**

<table>
<thead>
<tr>
<th>Job Number</th>
<th>Job Owner</th>
<th>Job Script</th>
<th>Nodes Allocated</th>
<th>Nodes Used</th>
<th>MPI Ranks Used</th>
<th>Threads</th>
</tr>
</thead>
<tbody>
<tr>
<td>1025844</td>
<td>pjmucci</td>
<td>miniaero-sbatch-3d_sod_parallel_test_big.sh</td>
<td>16</td>
<td>8</td>
<td>32 (map)</td>
<td>1</td>
</tr>
</tbody>
</table>

**Job Path:** 
...tests/3D_Sod_Parallel_Big/miniaero.exe

**Job Time:** 
0:26:23.075943

**mpiexec Time:** 
0:26:16.289926

**Start Time:** 
2016-03-20 11:03:40

**Finish Time:** 
2016-03-20 11:30:03
Job Info

JOB DETAILS

RAW DATA | CLOCK SCALING | JOB OUTPUT | MODULES

Job Number: 1025844
Job Owner: pjmucci
Job Script: miniaero-sbatch-3d_sod_parallel_test_big.sh
Job Path: ...tests/3D_Sod_Parallel_Big/miniaero.exe
Job Time: 0:26:23.075943
Start Time: 2016-03-20 11:03:40
Finish Time: 2016-03-20 11:30:03

Nodes Allocated: 16
Nodes Used: 8
MPI Ranks Used: 32 (map)
Threads: 1

N=$((N-32))
Nmo=$((N-1))
#slot | hthread | core | socket (default) | numa | board | node
policy=$[policy="--map-by-socket:PE=8"]
echo "Running parallel 3D sod test using $policy and $N processors.
mpirun $policy -np $N ././make/src/miniaero.exe
diff=0
for i in \"seq 0 $Nmo\";
do
data tools/numeric_text_diff results.$i results.$i.gold > diff.$i.txt
diff=$([diff + $i])
done
if [ diff -eq 0 ];

PerfMiner

Running parallel 3D sod test using --map-by-socket:PE=8 and 32 processors.
script: miniaero-sbatch-3d_sod_parallel_test_big.sh
mpip: Found MPIP environment variable [-y -k -o -q -t slurm-1025844.perfminer]
mpip: Set the callstack stack traceback depth to [0]
mpip: Set the output directory to [slurm-1025844.perfminer]
mpip: mpip: mpip V3.4.2 (Build Feb 17 2016/04/46:45)
mpip: Direct questions and errors to mpip-help@lists.sourceforge.net
mpip: ...
  Face creation time: 1.00 seconds ...
...
  Extract BC face and delete ghost time: 0.00 seconds ...
  Start setup communication...
...
  Setup Communication function time: 0.00 seconds ...
  Rest of setup communication time: 1.00 seconds ...
  End setup communication...
...
  Setup time: 3.00 seconds ...

COE Phoenix April, 2016
Job Info

**JOB DETAILS**

- **Job Number:** 1025844
- **Job Owner:** pjmucci
- **Job Script:** miniaero-sbatch-3d_sod_parallel_test_big.sh
- **Job Path:** ...tests/3D_Sod_Parallel_Big/miniaero.exe
- **Job Time:** 0:26:23.075943
- **mpirun Time:** 0:26:16.289926
- **Start Time:** 2016-03-20 11:03:48
- **Finish Time:** 2016-03-20 11:30:03
- **Nodes Allocated:** 16
- **Nodes Used:** 8
- **MPI Ranks Used:** 32 (map)
- **Threads:** 1

**FREQUENCY SCALING**
Metric Correlation

**JOB DETAILS**

- **Job Number**: 1015644
- **Job Owner**: pjmucci
- **Job Script**: miniarc-batch...
  3d_soc_parallel_test_big.sh
- **Job Path**: ...tests/3d_soc_parallel_big/miniaerc.exe
- **Job Time**: 8:16:23.675943
- **Start Time**: 2016-03-20 11:03:40
- **Finish Time**: 2016-03-20 11:30:03
- **Nodes Allocated**: 15
- **Nodes Used**: 8
- **MPI Ranks Used**: 32 (mp)
- **Threads**: 1

**JOB OVERVIEW**

![Bar chart showing CPU and MPI usage over time]

**JOB PERFORMANCE DETAILS**

<table>
<thead>
<tr>
<th>CPU</th>
<th>MPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instructions</td>
<td>Cache</td>
</tr>
</tbody>
</table>

![Pie chart showing instructions breakdown]

- **AVX**: 8.9%
- **Branches**: 15.2%
- **Memory Loads**: 42.7%
- **Memory Stores**: 31.2%
- **Others**: 31.2%
Metric Correlation

JOB PERFORMANCE DETAILS

<table>
<thead>
<tr>
<th>CPU</th>
<th>MPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instructions</td>
<td>Cache</td>
</tr>
</tbody>
</table>

![Bar chart showing job performance details](chart1.png)

JOB PERFORMANCE DETAILS

<table>
<thead>
<tr>
<th>CPU</th>
<th>MPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instructions</td>
<td>Cache</td>
</tr>
</tbody>
</table>

![Bar chart showing job performance details](chart2.png)
# Metric Correlation

## JOB PERFORMANCE DETAILS

<table>
<thead>
<tr>
<th>CPU</th>
<th>NPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instructions</td>
<td>Cache</td>
</tr>
</tbody>
</table>

### Instructions

![Instructions Chart](chart_instructions.png)

### Stalls

![Stalls Chart](chart_stalls.png)
THE END!

jeacook@sandia.gov
phil@minimalmetrics.com
MPI, Outlier Detection, Productivity