Virtual Institute – High Productivity Supercomputing

1 February 2012

Brian Wylie
Jülich Supercomputing Centre
b.wylie@fz-juelich.de
Wednesday 1 February

- 09:00
  - Introduction to VI-HPS
  - Large-scale parallel performance analysis with Scalasca
- 10:00-10:15 (break)
- 10:00-10:15 (break)
- 12:00-14:00 (lunch)
  - Using Scalasca: hands-on tutorial with BT-MZ on gofree
- 12:00-14:00 (lunch)
- 15:30-15:45 (break)
  - Using Scalasca: coaching with your own code(s)
- 15:30-15:45 (break)
- 17:00 (adjourn)
We'd like to know a little about you, your application(s), and your expectations and desires from this tutorial

- What programming paradigms do you use in your app(s)?
  - only MPI, only OpenMP, mixed-mode/hybrid OpenMP/MPI, ...
  - Fortran, C, C++, multi-language, ...

- What platforms/systems must your app(s) run well on?
  - Cray XT/XE/XK, IBM BlueGene, SGI Altix, Linux cluster™, ...

- Who's already familiar with serial performance analysis?
  - Which tools have you used?
    - time, print/printf, prof/gprof, ...

- Who's already familiar with parallel performance analysis?
  - Which tools have you used?
    - time, print/printf, prof/gprof, Scalasca, TAU, Vampir, ...
Prepare to analyse your own application codes

- Ensure your application codes build and run to completion with appropriate datasets
  - initial configuration should ideally run in less than 15 minutes with 1-4 compute nodes (up to 48 processes/threads)
    - to facilitate rapid turnaround and quick experimentation
  - larger/larger scalability configurations are also interesting
    - turnaround may be limited due to busyness of batch queues

- Compare your application performance on other systems
  - VI-HPS tools already installed on a number of HPC systems
    - if not, ask your system administrator to install them
      (or install a personal copy yourself)
Goal: Improve the quality and accelerate the development process of complex simulation codes running on highly-parallel computer systems

- Start-up funding (2006-2011) by Helmholtz Association of German Research Centres

- Activities
  - Development and integration of HPC programming tools
    - Correctness checking & performance analysis
  - Training workshops
  - Service
    - Support email lists
    - Application engagement
  - Academic workshops

www.vi-hps.org
VI-HPS partners & associates

Forschungszentrum Jülich
  - Jülich Supercomputing Centre

RWTH Aachen University
  - Centre for Computing & Communication

Technical University of Dresden
  - Centre for Information Services & HPC

University of Tennessee (Knoxville)
  - Innovative Computing Laboratory

German Research School
  - Laboratory of Parallel Programming

Technical University of Munich
  - Chair for Computer Architecture

University of Oregon
  - Performance Research Laboratory

University of Stuttgart
  - HPC Centre
VI-HPS productivity tools

- Marmot/MUST
  - MPI correctness checking
- PAPI
  - Interfacing to hardware performance counters
- Periscope
  - Automatic analysis via an on-line distributed search
- Scalasca
  - Large-scale parallel performance analysis
- TAU
  - Integrated parallel performance system
- Vampir/VampirTrace
  - Event tracing and graphical trace visualization & analysis
VI-HPS productivity tools (cont'd)

- **KCacheGrind**
  - Callgraph-based cache analysis [x86 only]
- **MAQAO**
  - Assembly instrumentation & optimization [x86 only]
- **ompP**
  - OpenMP profiling tool
- **OpenMPI**
  - Memory checking
- **Paraver/Extrae**
  - Event tracing and graphical trace visualization & analysis
- **Score-P**
  - Common instrumentation & measurement infrastructure
- **SIONlib**
  - Optimized native parallel file I/O
Technologies and their integration

- Technologies:
  - SCALASCA
  - VAMPIR / PARAVER
  - PAPI
  - MARMOT / MUST
  - KCACHEGRIND
  - PERISCOPE
  - TAU
  - MAQAO
  - SIONLIB / OPENMPI

- Integration Stages:
  - Error detection
  - Hardware monitoring
  - Automatic profile & trace analysis
  - Visual trace analysis
  - Execution
  - Optimization
- Goals
  - Give an overview of the programming tools suite
  - Explain the functionality of individual tools
  - Teach how to use the tools effectively
  - Offer hands-on experience and expert assistance using tools
  - Receive feedback from users to guide future development
- For best results, bring & analyse/tune your own code(s)!

- VI-HPS Tutorial series
  - SC'08, ICCS'09, SC'09, Cluster'10, SC'10, SC'11
- VI-HPS Tuning Workshop series
  - 2008 (Aachen & Dresden), 2009 (Jülich & Bremen), 2010 (Garching & Amsterdam), 2011 (Stuttgart & Aachen)
  - 2012/04/23-27 (Versailles), 2012/10/15-19 (Garching)
Upcoming VI-HPS training events

- 9th VI-HPS Tuning Workshop (23-27 Apr 2012)
  - hosted by UVSQ, Versailles, France
  - using PRACE Tier-0 Curie system at CEA / TGCC
  - Scalasca, Vampir, TAU, Periscope, KCachegrind, MAQAO, ...

- Further events to be determined
  - (one-day) tutorials
    - with guided exercises using Live DVD
  - (multi-day) training workshops
    - with your own applications on real HPC systems

Check [www.vi-hps.org/training](http://www.vi-hps.org/training) for announced events

- Contact us if you might be interested in hosting an event
• Bootable Linux installation ISO (on DVD or USB stick)
• Includes everything needed to try out our parallel tools on an x86-architecture notebook computer
  ▪ VI-HPS tools: KCachegrind, Marmot, PAPI, Periscope, Scalasca, TAU, VT/Vampir*
  ▪ Also: Eclipse/PTP, TotalView*, etc.
    ▷ * time/capability-limited evaluation licences provided for commercial products
▪ GCC (w/ OpenMP), OpenMPI
▪ Manuals/User Guides
▪ Tutorial exercises & examples
• Produced by U. Oregon PRL
  ▪ Sameer Shende
Cachegrind: cache analysis by simple cache simulation
- Captures dynamic callgraph
- Based on valgrind dynamic binary instrumentation
- Runs on x86/PowerPC/ARM unmodified binaries
  - No root access required
- ASCII reports produced

[KQ] Cachegrind GUI
- Visualization of cachegrind output

Developed by TU Munich
- Released as GPL open-source
- http://kcachegrind.sf.net/

Diagram:
- Binary → Valgrind
  - Memory Accesses
  - Event Counters
  - Debug Info
- 2-level $ Simulator
- Profile
KCachegrind GUI

- Event cost tree map
- Source code view
- Call graph view
- Machine code annotation
Tool to check for correct MPI usage at runtime

- Checks conformance to MPI standard
  - Supports Fortran & C bindings of MPI-1.2
- Checks parameters passed to MPI
- Monitors MPI resource usage

Implementation

- C++ library gets linked to the application
- Does not require source code modifications
- Additional process used as DebugServer
- Results written in a log file (ASCII/HTML/CUBE)

Developed by HLRS & TU Dresden

- Released as open-source
- http://www.hlrs.de/organization/av/amt/projects/marmot
Next generation MPI runtime error detection tool
- Successor of the Marmot and Umpire tools
- Initial merge of Marmot's many local checks with Umpire's non-local checks
- Improved scalability expected in future

Developed by TU Dresden, LLNL & LANL
- to be released as open-source (BSD license)
- currently in beta-testing for first release in November 2011
- http://tu-dresden.de/.../must
Portable performance counter library & utilities

- Configures and accesses hardware/system counters
- Predefined events derived from available native counters
- Core component for CPU/processor counters
  - instructions, floating point operations, branches predicted/taken, cache accesses/misses, TLB misses, cycles, stall cycles, …
  - performs transparent multiplexing when required
- Extensible components for off-processor counters
  - InfiniBand network, Lustre filesystem, system hardware health, …
- Used by multi-platform performance measurement tools
  - Periscope, Scalasca, TAU, VampirTrace, ...

Developed by UTK-ICL
- Available as open-source for most modern processors
  http://icl.cs.utk.edu/papi/
Available events and hardware information.

-----------------------------------------------
PAPI Version             : 4.1.0.0
Vendor string and code   : GenuineIntel (1)
Model string and code    : Intel(R) Xeon(R) CPU X5570 @ 2.93GHz (26)
CPU Revision             : 5.000000
CPUID Info               : Family: 6 Model: 26 Stepping: 5
CPU Megahertz            : 1600.000000
CPU Clock Megahertz      : 1600
Hdw Threads per core     : 2
Cores per Socket         : 4
NUMA Nodes               : 2
CPU's per Node           : 8
Total CPU's              : 16
Number Hardware Counters : 16
Max Multiplex Counters   : 512
-----------------------------------------------

Name        Code    Avail Deriv Description
PAPI_L1_DCM 0x80000000  Yes No Level 1 data cache misses
 PAPI_L1_ICM 0x80000001  Yes No Level 1 instruction cache misses

Of 107 possible events, 35 are available, of which 9 are derived.
### PAPI native counters (and qualifiers)

```
juropa$ papi_native_avail
Available native events and hardware information.

<table>
<thead>
<tr>
<th>Event Code</th>
<th>Symbol</th>
<th>Long Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x40000000</td>
<td>UNHALTED_CORE_CYCLES</td>
<td>count core clock cycles whenever the clock signal on the specific core is running (not halted). Alias to event CPU_CLK_UNHALTED:THREAD</td>
</tr>
<tr>
<td>0x40000001</td>
<td>INSTRUCTION_RETIRED</td>
<td>count the number of instructions at retirement. Alias to event INST_RETIRED:ANY_P</td>
</tr>
</tbody>
</table>

```

Total events reported: 135
Automated profile-based performance analysis

- Iterative on-line performance analysis
  - Multiple distributed hierarchical agents
- Automatic search for bottlenecks based on properties formalizing expert knowledge
  - MPI wait states
  - Processor utilization hardware counters
- Clustering of processes/threads with similar properties
- Eclipse-based integrated environment

Supports

- SGI Altix Itanium2, IBM Power and x86-based architectures

Developed by TU Munich

- Released as open-source
- http://www.lrr.in.tum.de/periscope
MPI

- Excessive MPI communication time
- Excessive MPI time due to many small messages
- Excessive MPI time in receive due to late sender
- ...

Hardware performance counters (platform-specific)

- Cycles lost due to cache misses
  - High L1/L2/L3 demand load miss rate
- Cycles lost due to store instructions
- Cycles lost due to address translation misses
- Cycles lost due to no instruction to dispatch
- ...

Periscope properties & strategies (examples)
Periscope plug-in to Eclipse environment

Source code view

SIR outline view

Project view

Properties view
Automatic performance analysis toolset

- Scalable performance analysis of large-scale applications
  - particularly focused on MPI & OpenMP paradigms
  - analysis of communication & synchronization overheads
- Automatic and manual instrumentation capabilities
- Runtime summarization and/or event trace analyses
- Automatic search of event traces for patterns of inefficiency
  - Scalable trace analysis based on parallel replay
- Interactive exploration GUI and algebra utilities for XML callpath profile analysis reports

Developed by JSC & GRS

- Released as open-source
- http://www.scalasca.org/
Late Sender Time

Description:
Refers to the time lost waiting caused by a blocking receive operation (e.g., MPI_Recv()) or MPI_Wait()) that is posted earlier than the corresponding send operation.
Integrated performance toolkit

- Instrumentation, measurement, analysis & visualization
  - Highly customizable installation, API, envvars & GUI
  - Supports multiple profiling & tracing capabilities
- Performance data management & data mining
- Targets all parallel programming/execution paradigms
  - Ported to a wide range of computer systems
- Performance problem solving framework for HPC
- Extensive bridges to/from other performance tools
  - PerfSuite, Scalasca, Vampir, ...

Developed by U. Oregon/PRL

- Broadly deployed open-source software
- http://tau.uoregon.edu/
TAU ParaProf GUI displays (selected)
TAU PerfExplorer data mining
Interactive event trace analysis

- Alternative & supplement to automatic trace analysis
- Visual presentation of dynamic runtime behaviour
  - event timeline chart for states & interactions of processes/threads
  - communication statistics, summaries & more
- Interactive browsing, zooming, selecting
  - linked displays & statistics adapt to selected time interval (zoom)
  - scalable server runs in parallel to handle larger traces

Developed by TU Dresden ZIH

- Open-source VampirTrace library bundled with OpenMPI 1.3
- http://www.tu-dresden.de/zh/vampirtrace/
- Vampir Server & GUI have a commercial license
- http://www.vampir.eu/
Vampir interactive trace analysis GUI
Vampir interactive trace analysis GUI
Vampir interactive trace analysis GUI (zoom)
Interactive event trace analysis
- Visual presentation of dynamic runtime behaviour
  - event timeline chart for states & interactions of processes
  - Interactive browsing, zooming, selecting
- Large variety of highly configurable analyses & displays

Developed by Barcelona Supercomputing Center
- Paraver trace analyser and Extrae measurement library
- Open source available from http://www.bsc.es/paraver/
### Paraver Interactive Trace Analysis GUI

#### Process and Thread Overview

<table>
<thead>
<tr>
<th>Process</th>
<th>Idle (%)</th>
<th>Running (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>8.26 %</td>
<td>91.74 %</td>
</tr>
<tr>
<td>1</td>
<td>7.33 %</td>
<td>92.67 %</td>
</tr>
<tr>
<td>2</td>
<td>30.30 %</td>
<td>69.70 %</td>
</tr>
<tr>
<td>3</td>
<td>8.34 %</td>
<td>91.66 %</td>
</tr>
<tr>
<td>4</td>
<td>8.43 %</td>
<td>91.57 %</td>
</tr>
<tr>
<td>5</td>
<td>8.31 %</td>
<td>91.69 %</td>
</tr>
<tr>
<td>6</td>
<td>31.77 %</td>
<td>68.23 %</td>
</tr>
<tr>
<td>7</td>
<td>8.64 %</td>
<td>91.36 %</td>
</tr>
<tr>
<td>8</td>
<td>8.64 %</td>
<td>91.36 %</td>
</tr>
<tr>
<td>9</td>
<td>8.59 %</td>
<td>91.41 %</td>
</tr>
<tr>
<td>10</td>
<td>32.52 %</td>
<td>67.48 %</td>
</tr>
</tbody>
</table>

#### Total Statistics

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Idle</td>
<td>224.89 %</td>
</tr>
<tr>
<td>Total Running</td>
<td>1,375.11 %</td>
</tr>
<tr>
<td>Average Idle</td>
<td>14.06 %</td>
</tr>
<tr>
<td>Average Running</td>
<td>85.94 %</td>
</tr>
<tr>
<td>Maximum Idle</td>
<td>32.52 %</td>
</tr>
<tr>
<td>Maximum Running</td>
<td>92.98 %</td>
</tr>
<tr>
<td>Minimum Idle</td>
<td>7.02 %</td>
</tr>
<tr>
<td>Minimum Running</td>
<td>67.48 %</td>
</tr>
<tr>
<td>StdDev Idle</td>
<td>10.18 %</td>
</tr>
<tr>
<td>StdDev Running</td>
<td>10.18 %</td>
</tr>
<tr>
<td>Avg/Max</td>
<td>0.43/0.52</td>
</tr>
</tbody>
</table>
Key tool components also provided as open-source

- **Program/library instrumentation**
  - COBI, OPARI, PDToolkit

- **Runtime measurement systems**
  - Score-P, UniMCI

- **Scalable I/O**
  - SIONlib

- **Libraries & tools for handling (and converting) traces**
  - EPILOG, PEARL, OTF

- **Analysis algebra & hierarchical/topological presentation**
  - CUBE
Portable native parallel I/O library & utilities

- Scalable massively-parallel I/O to task-local files
- Manages single or multiple physical files on disk
  - optimizes bandwidth available from I/O servers by matching blocksizes/alignment, reduces metadata-server contention
- POSIX-I/O-compatible sequential & parallel API
  - adoption requires minimal source-code changes
- Tuned for common parallel filesystems
  - GPFS (BlueGene), Lustre (Cray), ...
- Convenient for application I/O, checkpointing,
  - Used by Scalasca tracing (when configured)

Developed by JSC

- Available as open-source from http://www.fz-juelich.de/jsc/sionlib/