Scalasca: A Scalable Portable Performance Measurement and Analysis Toolset

1. July 2011 | Bernd Mohr
Institute for Advanced Simulation (IAS)
Jülich Supercomputing Centre (JSC)

Scalasca Overview

- **Goal**
  - Measure and analyze performance of parallel programs

- **Context**
  - “Real-world” HPC applications
  - Batch processing, space sharing
  - Large distributed memory machines

- **Current focus**
  - Automation
  - Portability
  - Extreme scalability w.r.t. number of cores
Increasing Importance of Scaling

• Number of Processors share for TOP 500 Jun 2011

<table>
<thead>
<tr>
<th>NProc</th>
<th>Count</th>
<th>Share</th>
<th>(\Sigma R_{\text{max}})</th>
<th>Share</th>
<th>(\Sigma N_{\text{Proc}})</th>
</tr>
</thead>
<tbody>
<tr>
<td>1025-2048</td>
<td>2</td>
<td>0.4%</td>
<td>168 TF</td>
<td>0.3%</td>
<td>2,632</td>
</tr>
<tr>
<td>2049-4096</td>
<td>20</td>
<td>4.0%</td>
<td>1,177 TF</td>
<td>2.0%</td>
<td>71,734</td>
</tr>
<tr>
<td>4097-8192</td>
<td>195</td>
<td>39.0%</td>
<td>9,759 TF</td>
<td>16.6%</td>
<td>1,262,738</td>
</tr>
<tr>
<td>8193-16384</td>
<td>224</td>
<td>44.8%</td>
<td>15,216 TF</td>
<td>25.8%</td>
<td>2,337,998</td>
</tr>
<tr>
<td>&gt; 16384</td>
<td>59</td>
<td>11.8%</td>
<td>32,556 TF</td>
<td>55.3%</td>
<td>4,100,234</td>
</tr>
<tr>
<td>Total</td>
<td>500</td>
<td>100%</td>
<td>58,876 TF</td>
<td>100%</td>
<td>7,775,336</td>
</tr>
</tbody>
</table>

• Average system size: 15,551 cores
• Median system size: 8,556 cores

Increasing Importance of Scaling II

• Number of Processors share for TOP 500 Jun 2001 – Jun 2011
Portability: Supported Platforms

- Instrumentation and measurement only (visual analysis on front-end or workstation)
  - Cray XT3/XT4/XT5 and XE6
  - IBM BlueGene/L, BlueGene/P, BlueGene/Q
  - NEC SX8 and SX9

- Full support (instrumentation, measurement, and automatic analysis)
  - Linux IA32, IA64, x86_64, and PPC based clusters
  - IBM AIX Power3/4/5/6 based clusters
  - SGI Irix MIPS based clusters (Origin 2K, Origin 3K)
  - SGI Linux IA64 and x86_64 based clusters (Altix, …)
  - SUN Solaris Sparc and x86/x86_64 based clusters

---

Known Installations of Scalasca

- Companies
  - Bull (France)
  - Dassault Aviation (France)
  - GNS (Germany)
  - MAGMA (Germany)
  - RECOM (Germany)
  - Shell (Netherlands)
  - Sun Microsystems (USA, Singapore, India)
  - Qontix (UK)

- Research and Supercomputing Centres
  - ANL (USA)
  - BSC (Spain)
  - CEA (France)
  - CERFACS (France)
  - CINECA (Italy)
  - CSC (Finland)
  - CSCS (Switzerland)
  - DLR (Germany)
  - DKRZ (Germany)
  - EPCC (UK)
  - HLRN (Germany)
  - HLRS (Germany)
  - ICHEC (Ireland)
  - IDRIS (France)

- Research and Supercomputing Centres (cont.)
  - LNL (USA)
  - LRZ (Germany)
  - NCAR (USA)
  - NCSA (USA)
  - ORNL (USA)
  - PSC (USA)
  - RZG (Germany)
  - SARA (Netherlands)
  - SAITC (Bulgaria)
  - TACC (USA)

- Universities
  - RPI (USA)
  - RWTH (Germany)
  - TUD (Germany)
  - UOregon (USA)
  - UTK (USA)

- DoD Computing Centers (all USA)
  - AFRL DSRC
  - ARL DSRC
  - ARSC DSRC
  - ERDC DSRC
  - Navy DSRC
  - MHPCc DSRC
  - SSC-Pacifica
“A picture is worth 1000 words…”

- MPI ring program
- “Real world” example

“What about 100’s of pictures?”
(with 100’s of menu options)
Example Automatic Analysis: Late Sender

Example Patterns

(a) Late Sender

(b) Late Receiver

(c) Late Sender / Wrong Order

(d) Wait at N x N

© 2011 Bernd Mohr
The Scalasca Project

- **Scalable Analysis of Large Scale Applications**
- Follow-up project to KOJAK

http://www.scalasca.org/

**Approach**
- **Instrument** C, C++, and Fortran parallel applications
  - Based on MPI, OpenMP, SHMEM, or hybrid
- **Option 1: scalable call-path profiling**
- **Option 2: scalable event trace analysis**
  - **Collect** event traces
  - **Search** trace for event patterns representing inefficiencies
  - **Categorize and rank** inefficiencies found

- Supports MPI 2.2 and basic OpenMP

Analysis Process: Profiling

- **measurement library**
- **PAPI**
- **Instrumented process**
- **Instrumented executable**
- **Multi-level instrumenter**
- **Source modules**

**Summary report**

- **TAU paraprof**
- **CUBE Report explorer**

Report manipulation

Third-party component
CUBE Result Browser

- Representation of results (3D severity matrix) along three hierarchical axes
  - Metric
  - Call tree path
  - System location
- Three coupled tree browsers
- Each node displays severity
  - As colour: for easy identification of bottlenecks
  - As value: for precise comparison

CUBE Result Browser (II)

- Each node displays severity
  - as colour
  - as value
- Dependent on state
  - **Collapsed**
    - Inclusive time
    - Entire time spent in the function
  - **Expanded**
    - Exclusive time
    - Time spent in the function without taking calls to children into account

```
int main()
{
  int a;
  a = a + 1;
  foo();
  bar();
  a = a + 1;
  return a;
}
```
**CUBE Result Browser (III)**

Value boxes colored according to scale

**Metric Dimension**

What kind of performance problem?

Right-click metric context menu for info or description
Call Tree Dimension

Where is it in the source code? In what context?

Right-click function context menu to go to source location or to manipulate tree

System Tree Dimension

How is it distributed across the system
Alternative: Topology View

Summary analysis sweep3D@294,912
### Trace analysis sweep3D@294,912

- 10 min sweep3D runtime
- 11 sec replay
- 4 min trace data write/read (576 files)
- 7.6 TB buffered trace data
- 510 billion events
Scalasca Unified Command: scalasca

- Run without action argument for basic usage info
  
  % scalasca
  usage: scalasca [-v][-n] {action}

  1. prepare application objects and executable for measurement:
     scalasca -instrument <compile-or-link-command> # skin
  2. run application under control of measurement system:
     scalasca -analyze <application-launch-command> # scan
  3. interactively explore measurement analysis report:
     scalasca -examine <experiment-archive|report> # square

  [-h] show quick reference guide (only)

- Simply a convenience wrapper for action commands
- For full details see Scalasca Quick Reference Guide

Scalasca Instrumenter: skin

- Usage: scalasca -instrument [opts] <compile_cmd>
  - scalasca -instrument -user mpicc -fast -c bar.c
  - skin mpif90 -openmp -O3 -o foobar foo.c bar.o -lm

- Processes source modules during compile and augments link with measurement library
  - Configures automatic function instrumentation capability of native compiler (if available)
    - All functions in source module(s) are instrumented
  - [--pomp] option enables processing of POMP directives
    - Optional manual source annotation of functions + regions
    - Replaces automatic function instrumentation
  - [--user] option activates Scalasca user-annotation API
  - [--pdt] option uses TAU PDT instrumenter
Scalasca Collector + Analyzer: scan

- **Usage:** `scalasca -analyze [opts] <launch_cmd>`
  - `scan mpirun -np 2 ./myprog infile`
- Prepares and runs measurement collection, with follow-on trace analysis (if appropriate)
  - `[-n]` preview without executing launches
  - `[-s]` enables runtime summarization [default]
  - `[-q -t]` enables trace collection & pattern analysis
  - determines NP and/or NT (number of processes + threads) and MODE=vn|co|dual|smp (where appropriate)
  - names default measurement experiment archive `epik_${TARGET}_${MODE}${NP}x${NT}_[sum|trace]`
  - `[-f filter]` specifies file listing functions not to be measured
  - `[-m metric1:metric2:...]` includes HW ctr metrics

Scalasca Analysis Report Explorer: square

- **Usage:** `scalasca -examine [opts] <epik_archive>`
  - `scalasca -examine epik_sweep3d_co32_trace`
  - `square epik_sweep3d_co32_trace`
- Prepares and presents final analysis report
  - Checks EPIK archive directory for CUBE files
  - Remaps primitive initial analysis report(s) into refined formal report(s) with enriched metrics + metric hierarchies
    - `epitome.cube ⇔ summary.cube`
    - `scout.cube ⇔ trace.cube`
  - Presents refined report in CUBE3 browser
    - Trace analysis shown in preference to summary analysis
    - Additional reports can be loaded via File/Open menu
  - `[-s]` skip display and output textual score report
EPIK Experiment Archive

- Directory created by measurement library
  - Measurement aborts if archive already exists!

- Contains all files related to measurement
  - Measurement & analysis logs (epik.log, scout.log, etc.)
  - Primitive analysis reports (epitome.cube, scout.cube)
  - Refined analysis reports (summary.cube, trace.cube)
  - Process trace datafiles (ELG/*)
  - Unified definitions & map data (epik.esd, epik.map)
  - Miscellaneous (epik.conf, epik.filt, epik.path)

CUBE Algebra Tools

- CUBE files can be compared/combined with some useful command line tools
- Note that these work directly on CUBE files and not on archive directories
  - Reads CUBE2 & CUBE3 files, but only writes CUBE3 files

- General usage:
  - `cube3_TOOLS [-o <output file>] <input file>`

- If no output file is specified, `tool.cube` is generated
CUBE Algebra Tools (II)

- **cube3_mean**
  - Can eliminate “measurement noise” by averaging the results of several experiments

- **cube3_cut [-p prune] [-r root]**
  - Creates a new CUBE file without pruned subtrees and/or containing only the specified call tree node as new root(s)

- **cube3_diff**
  - Calculates the difference of two experiments
  - Useful to measure improvement/degradation due to a modification

Diff Example

- Negative value [Sunken color box] ⇒ Degradation
- Positive value [Raised color box] ⇒ Improvement
Scalasca Limitations

- **MPI 2.2 apart from dynamic process creation**
  - C++ interface deprecated with MPI 2.2
- **OpenMP 2.5 apart from nested thread teams**
  - partial support for dynamically-sized and/or conditional thread teams (*)
    - automatic trace analysis currently not supported
    - no support for OpenMP used in macros or included files
- **No OpenMP 3.0 tasks yet**
- Hybrid OpenMP+MPI
  - partial support for non-uniform thread teams (*)
  - no support for MPI_THREAD_MULTIPLE

(*) Summary & trace measurements are possible, and traces may be analyzed with Vampir or other trace visualizers

Scalasca Interoperability

- **Instrumentation**
  - Scalasca source instrumentation via TAU/PDToolkit
  - Adapter for VT manual instrumentation macros
  - TAU instrumentation with Scalasca measurement
- **Profile experiments**
  - Alternative presentation with TAU Paraprof/PerfExplorer
- **Trace experiments**
  - Vampir 7.3 and VampirServer 2.3 can read and display Scalasca traces
  - Trace converters for
    - paraver (elg2prv)
    - Vampir (elg2otf)
3D Visualization

- Visualizing CUBE data using the TAU ParaProf 3D visualizer

Vampir Trace Visualization

- Scalasca trace of hybrid MPI + OpenMP code
Vampir Trace Visualization

- Scalasca trace of hybrid MPI + OpenMP code

The Scalasca Team

JSC
- Michael Knobloch
- Bernd Mohr
- Peter Philipp

Helmholtz Group
- Dominic Eschweiler
- Markus Geimer
- Daniel Lorenz
- Christian Rössler
- Daniel Becker
- David Böhme
- Marc-André Hermans

GRS
- Pavel Savvinkau
- Alexandre Stube
- Brian Wylie
- Christian Siebert
- Zoltán Szabenyi
- Felix Wolff

Sponsors

Deutsche Forschungsgemeinschaft

Bundesministerium für Bildung und Forschung

ITEA2

SEVENTY-FRAMEWORK PROGRAMME
Thank you!

WRF-NMM weather prediction code on MareNostrum @ 1600 CPUs

http://www.scalasca.org
scalasca@fz-juelich.de