Score-P and Scalasca
Portable open-source tools for scalable performance analysis

February 1, 2014 | Alexandre Otto Strube |
Outline

Going Exascale
Scalasca
We’re not alone
Things got messy
Unification
Who uses/develops Score-P
What is ours
Extreme scalability
The future
Going Exascale

Projected Performance Development

February 1, 2014
Alexandre Otto Strube
**TL;DR**

- Single core performance peaking
TL;DR

- Single core performance peaking
- # of cores increasing
TL;DR

- Single core performance peaking
- # of cores increasing
- Hybrid environments
TL;DR

- Single core performance peaking
- # of cores increasing
- Hybrid environments
- That affects YOU - TODAY - RIGHT NOW
TL;DR

- Single core performance peaking
- # of cores increasing
- Hybrid environments
- That affects YOU - TODAY - RIGHT NOW
- HPC is just the spearhead
TL;DR

- Single core performance peaking
- # of cores increasing
- Hybrid environments
- That affects YOU - TODAY - RIGHT NOW
- HPC is just the spearhead
- We only find the problems before the others
TL;DR

- Single core performance peaking
- # of cores increasing
- Hybrid environments
- That affects YOU - TODAY - RIGHT NOW
- HPC is just the spearhead
- We only find the problems before the others
- Supercomputers of today → notebooks of tomorrow
It doesn’t get easier

- Increasing machine complexity (gpu, accelerators, etc)
It doesn’t get easier

- Increasing machine complexity (gpu, accelerators, etc)
- Every doubling of scale reveals a new bottleneck
It doesn’t get easier

- Increasing machine complexity (gpu, accelerators, etc)
- Every doubling of scale reveals a new bottleneck
- Perturbation and data volume
It doesn’t get easier

- Increasing machine complexity (gpu, accelerators, etc)
- Every doubling of scale reveals a new bottleneck
- Perturbation and data volume
- Drawing insight from measurements
Example: Sweep3d Wait States on BG/P (2010)
This is an old song

- Several performance tools exist, for many years
This is an old song

- Several performance tools exist, for many years
- Most cease to work in huge processor/core counts
This is an old song

- Several performance tools exist, for many years
- Most cease to work in huge processor/core counts
- KOJAK performance tool was created 16 years ago.
Scalasca

- Started in 2006 (following KOJAK from ’98)
Scalasca

- Started in 2006 (following KOJAK from ’98)
- Goals:
Scalasca

- Started in 2006 (following KOJAK from ’98)
- Goals:
  - *Scalable* performance analysis toolset
Scalasca

- Started in 2006 (following KOJAK from ’98)
- Goals:
  - *Scalable* performance analysis toolset
  - Specifically targeting large-scale parallel applications such as those running on IBM Blue Gene or Cray XT with 10,000s or 100,000s of processes
Scalasca: Features

- Open source (New BSD license)
Scalasca: Features

- Open source (New BSD license)
- Portable
Scalasca: Features

- Open source (New BSD license)
- Portable
- IBM Blue Gene, Cray XT, SGI Altix, IBM SP, blade clusters, Solaris, Linux clusters, NEC SX, K Computer, Fujitsu FX10
Scalasca: Features

- Open source (New BSD license)
- Portable
- IBM Blue Gene, Cray XT, SGI Altix, IBM SP, blade clusters, Solaris, Linux clusters, NEC SX, K Computer, Fujitsu FX10
- Supports common parallel programming paradigms & languages
Scalasca: Features

- Open source (New BSD license)
- Portable
- IBM Blue Gene, Cray XT, SGI Altix, IBM SP, blade clusters, Solaris, Linux clusters, NEC SX, K Computer, Fujitsu FX10
- Supports common parallel programming paradigms & languages
  - Fortran, C, C++
Scalasca: Features

- Open source (New BSD license)
- Portable
- IBM Blue Gene, Cray XT, SGI Altix, IBM SP, blade clusters, Solaris, Linux clusters, NEC SX, K Computer, Fujitsu FX10
- Supports common parallel programming paradigms & languages
  - Fortran, C, C++
  - MPI 2.2, basic OpenMP & hybrid MPI+OpenMP
Scalasca: Features

- Open source (New BSD license)
- Portable
- IBM Blue Gene, Cray XT, SGI Altix, IBM SP, blade clusters, Solaris, Linux clusters, NEC SX, K Computer, Fujitsu FX10
- Supports common parallel programming paradigms & languages
  - Fortran, C, C++
  - MPI 2.2, basic OpenMP & hybrid MPI+OpenMP
- Unique:
Scalasca: Features

- Open source (New BSD license)
- Portable
- IBM Blue Gene, Cray XT, SGI Altix, IBM SP, blade clusters, Solaris, Linux clusters, NEC SX, K Computer, Fujitsu FX10
- Supports common parallel programming paradigms & languages
  - Fortran, C, C++
  - MPI 2.2, basic OpenMP & hybrid MPI+OpenMP
- Unique:
  - scalable trace analysis
Scalasca: Features

- Open source (New BSD license)
- Portable
- IBM Blue Gene, Cray XT, SGI Altix, IBM SP, blade clusters, Solaris, Linux clusters, NEC SX, K Computer, Fujitsu FX10
- Supports common parallel programming paradigms & languages
  - Fortran, C, C++
  - MPI 2.2, basic OpenMP & hybrid MPI+OpenMP
- Unique:
  - scalable trace analysis
  - Automatic wait-state search
Scalasca: Features

- Open source (New BSD license)
- Portable
- IBM Blue Gene, Cray XT, SGI Altix, IBM SP, blade clusters, Solaris, Linux clusters, NEC SX, K Computer, Fujitsu FX10
- Supports common parallel programming paradigms & languages
  - Fortran, C, C++
  - MPI 2.2, basic OpenMP & hybrid MPI+OpenMP
- Unique:
  - scalable trace analysis
  - Automatic wait-state search
  - Parallel replay exploits memory & processors to deliver scalability
Scalasca: Features

- Open source (New BSD license)
- Portable
- IBM Blue Gene, Cray XT, SGI Altix, IBM SP, blade clusters, Solaris, Linux clusters, NEC SX, K Computer, Fujitsu FX10
- Supports common parallel programming paradigms & languages
  - Fortran, C, C++
  - MPI 2.2, basic OpenMP & hybrid MPI+OpenMP
- Unique:
  - scalable trace analysis
  - Automatic wait-state search
  - Parallel replay exploits memory & processors to deliver scalability
    - *INSIGHTFUL*
This looks understandable...
... but this is a real code.
and this.
... it can get really confusing.
Scalasca
Scalasca
Scalasca

- Measurement library
- Instr. target application
- Instrumented executable
- Instrumenter compiler / linker
- Source modules

- Summary report
- Report manipulation

Which problem?  Where in the program?  Which process?

February 1, 2014  Alexandre Otto Strube

Slide 18
Scalasca

Optimized measurement configuration

- Measurement library
  - HWC
- Instr. target application
- Instrumented executable
- Source modules
- Instrumenter compiler / linker

Which problem?

Where in the program?

Which process?

Summary report

Report manipulation

February 1, 2014

Alexandre Otto Strube
We’re not alone

- Several tools exist
We’re not alone

- Several tools exist
- Different goals, similar needs
We’re not alone

- Several tools exist
- Different goals, similar needs
- Separate measurement systems and output formats
We’re not alone

- Several tools exist
- Different goals, similar needs
- Separate measurement systems and output formats
- Complementary features and overlapping functionality
We’re not alone

- Several tools exist
- Different goals, similar needs
- Separate measurement systems and output formats
- Complementary features and overlapping functionality
- Redundant effort for development and maintenance
We’re not alone

- Several tools exist
- Different goals, similar needs
- Separate measurement systems and output formats
- Complementary features and overlapping functionality
- Redundant effort for development and maintenance
- Limited or expensive interoperability
We’re not alone

- Several tools exist
- Different goals, similar needs
- Separate measurement systems and output formats
- Complementary features and overlapping functionality
- Redundant effort for development and maintenance
- Limited or expensive interoperability
- Complications for user experience, support, training
Things got messy
Unification
Score-P project idea

- Community project with common infrastructure

So, Score-P is the base instrumentation/measurement for several projects
Score-P project idea

- Community project with common infrastructure
- What we share:

So, Score-P is the base instrumentation/measurement for several projects
Score-P project idea

- Community project with common infrastructure
- What we share:
  - Single instrumentation and measurement system

So, Score-P is the base instrumentation/measurement for several projects
Score-P project idea

- Community project with common infrastructure
- What we share:
  - Single instrumentation and measurement system
  - Common data formats: Open Trace Format 2 (OTF2) for traces

So, Score-P is the base instrumentation/measurement for several projects
Score-P project idea

- Community project with common infrastructure
- What we share:
  - Single instrumentation and measurement system
  - Common data formats: Open Trace Format 2 (OTF2) for traces
  - Performance report: Cube4

So, Score-P is the base instrumentation/measurement for several projects
Score-P project idea

- Community project with common infrastructure
- What we share:
  - Single instrumentation and measurement system
  - Common data formats: Open Trace Format 2 (OTF2) for traces
  - Performance report: Cube4
- Single development effort, testing, support

So, Score-P is the base instrumentation/measurement for several projects
Score-P project idea

- Community project with common infrastructure
- What we share:
  - Single instrumentation and measurement system
  - Common data formats: Open Trace Format 2 (OTF2) for traces
  - Performance report: Cube4
- Single development effort, testing, support
- Single installation, interoperability, etc

So, Score-P is the base instrumentation/measurement for several projects
Who uses/develops Score-P?

- Scalasca (Fz-Juelich, RTWH Aachen)
Who uses/develops Score-P?

- Scalasca (Fz-Juelich, RTWH Aachen)
- Vampir (TU Dresden)
Who uses/develops Score-P?

- Scalasca (Fz-Juelich, RTWH Aachen)
- Vampir (TU Dresden)
- Periscope (Tu Munich)
Who uses/develops Score-P?

- Scalasca (Fz-Juelich, RTWH Aachen)
- Vampir (TU Dresden)
- Periscope (Tu Munich)
- Tau (U. Oregon)
And why we did it?
Cleaning the house
What do we measure?

- Measurement of MPI, OpenMP, User-level functions

Call-path profiles

Needs recompilation

Some overhead - might need filtering

Trace analysis

Identifies inefficiency patterns in communication and synchronization

Traces can quickly get huge - better filter that
What do we measure?

- Measurement of MPI, OpenMP, User-level functions
- Generation of flat MPI profiles

- Only relinking
- Minimum overhead
- Times each function was called
- Time spent in each function
- Amount of data transferred

- Call-path profiles
  - Needs recompilation
  - Some overhead - might need filtering

- Trace analysis
  - Identifies inefficiency patterns in communication and synchronization
  - Traces can quickly get huge - better filter that
What do we measure?

- Measurement of MPI, OpenMP, User-level functions
- Generation of flat MPI profiles
  - Only relinking

Minimum overhead
Times each function was called
Time spent in each function
Amount of data transferred
Call-path profiles

Needs recompilation
Some overhead - might need filtering

Trace analysis
Identifies inefficiency patterns in communication and synchronization
Traces can quickly get huge - better filter that
What do we measure?

- Measurement of MPI, OpenMP, User-level functions
- Generation of flat MPI profiles
  - Only relinking
  - Minimum overhead
What do we measure?

- Measurement of MPI, OpenMP, User-level functions
- Generation of flat MPI profiles
  - Only relinking
  - Minimum overhead
  - Times each function was called
What do we measure?

- Measurement of MPI, OpenMP, User-level functions
- Generation of flat MPI profiles
  - Only relinking
  - Minimum overhead
  - Times each function was called
  - Time spent in each function

Amount of data transferred

Call-path profiles

Needs recompilation

Some overhead - might need filtering

Trace analysis

Identifies inefficiency patterns in communication and synchronization

Traces can quickly get huge - better filter that
What do we measure?

- Measurement of MPI, OpenMP, User-level functions
- Generation of flat MPI profiles
  - Only relinking
  - Minimum overhead
  - Times each function was called
  - Time spent in each function
  - Amount of data transferred

Trace analysis
Identifies inefficiency patterns in communication and synchronization
Traces can quickly get huge - better filter
What do we measure?

- Measurement of MPI, OpenMP, User-level functions
- Generation of flat MPI profiles
  - Only relinking
  - Minimum overhead
  - Times each function was called
  - Time spent in each function
  - Amount of data transferred
- Call-path profiles
What do we measure?

- Measurement of MPI, OpenMP, User-level functions
- Generation of flat MPI profiles
  - Only relinking
  - Minimum overhead
  - Times each function was called
  - Time spent in each function
  - Amount of data transferred
- Call-path profiles
  - Needs recompilation
What do we measure?

- Measurement of MPI, OpenMP, User-level functions
- Generation of flat MPI profiles
  - Only relinking
  - Minimum overhead
  - Times each function was called
  - Time spent in each function
  - Amount of data transferred
- Call-path profiles
  - Needs recompilation
  - Some overhead - might need filtering
What do we measure?

- Measurement of MPI, OpenMP, User-level functions
- Generation of flat MPI profiles
  - Only relinking
  - Minimum overhead
  - Times each function was called
  - Time spent in each function
  - Amount of data transferred
- Call-path profiles
  - Needs recompilation
  - Some overhead - might need filtering
- Trace analysis
What do we measure?

- Measurement of MPI, OpenMP, User-level functions
- Generation of flat MPI profiles
  - Only relinking
  - Minimum overhead
  - Times each function was called
  - Time spent in each function
  - Amount of data transferred
- Call-path profiles
  - Needs recompilation
  - Some overhead - might need filtering
- Trace analysis
  - Identifies inefficiency patterns in communication and synchronization
What do we measure?

- Measurement of MPI, OpenMP, User-level functions
- Generation of flat MPI profiles
  - Only relinking
  - Minimum overhead
  - Times each function was called
  - Time spent in each function
  - Amount of data transferred
- Call-path profiles
  - Needs recompilation
  - Some overhead - might need filtering
- Trace analysis
  - Identifies inefficiency patterns in communication and synchronization
  - Traces can quickly get huge - better filter that
Extreme scalability

All parallel:

- Data collection/reduction
Extreme scalability

All parallel:

- Data collection/reduction
- Analysis:
Extreme scalability

All parallel:

- Data collection/reduction
- Analysis:
  - Pattern search
Extreme scalability

All parallel:

- Data collection/reduction
- Analysis:
  - Pattern search
  - Delay analysis
Extreme scalability

All parallel:

- Data collection/reduction
- Analysis:
  - Pattern search
  - Delay analysis
  - Critical-path analysis
Extreme scalability

All parallel:

- Data collection/reduction
- Analysis:
  - Pattern search
  - Delay analysis
  - Critical-path analysis
- Visualization
Some MPI patterns

(a) Late Sender

(b) Late Receiver

(c) Late Sender / Wrong Order

(d) Wait at N x N
Late sender
Late sender and application topology
Direct wait time analysis
Indirect wait time analysis
Direct wait time
Indirect wait time analysis
Root cause analysis
6D Hardware topology
The Future

WELCOME

TO THE WOLRD OF TOMORROW
The Future

- Energy awareness
The Future

- Energy awareness
- Bring performance analysis to YOU!
The Future

- Energy awareness
- Bring performance analysis to YOU!
- There’s a bunch of experts craving for users and parallel application developers!
The Future

- Energy awareness
- Bring performance analysis to YOU!
- There’s a bunch of experts craving for users and parallel application developers!
- support@score-p.org
The Future

- Energy awareness
- Bring performance analysis to YOU!
- There’s a bunch of experts craving for users and parallel application developers!
- support@score-p.org
- http://www.scalasca.org
Merrie Melodies

"That's all Folks!"

A WARNER BROS. CARTOON

A VITAPHONE® RELEASE