JÜLICH
FORSCHUNGSZENTRUM

Score-P and Scalasca

Portable open-source tools for scalable performance analysis

February 1, 2014 | Alexandre Otto Strube |

Member of the Helmholtz-Association

Outline

Going Exascale

Scalasca

We’re not alone

Things got messy

Unification

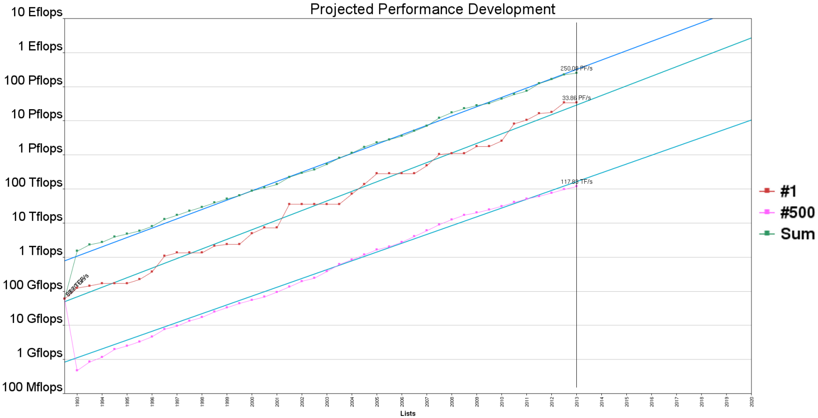
Who uses/develops Score-P

What is ours

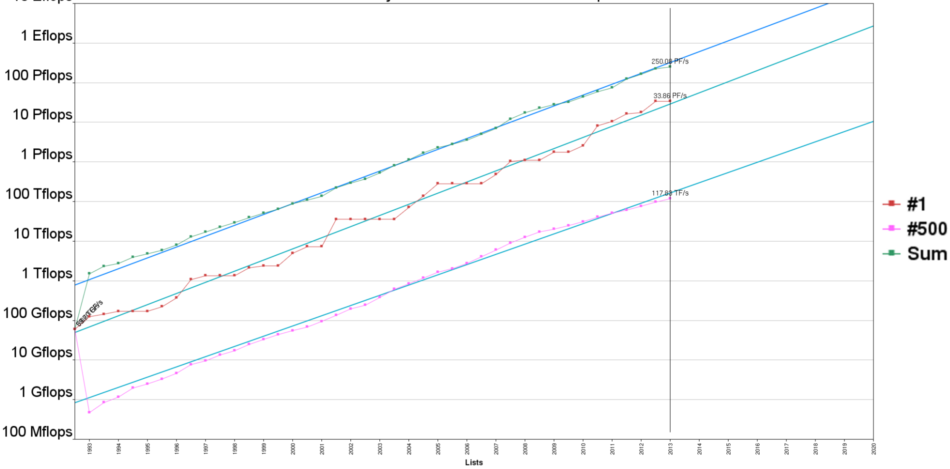
Extreme scalability

The future

Going Exascale



Projected Performance Development



TL;DR

- Single core perfomance peaking

TL;DR

- Single core perfomance peaking
- # of cores increasing

TL;DR

- Single core perfomance peaking
- # of cores increasing
- Hybrid environments

TL;DR

- Single core perfomance peaking
- # of cores increasing
- Hybrid environments
- That affects YOU - TODAY - RIGHT NOW

TL;DR

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

TL;DR

- Single core perfomance 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 perfomance 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

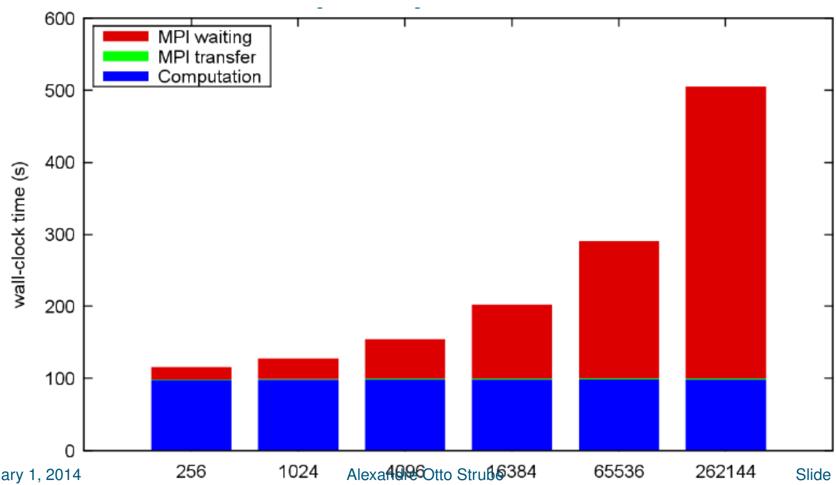
It doesn’t get easier

- 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

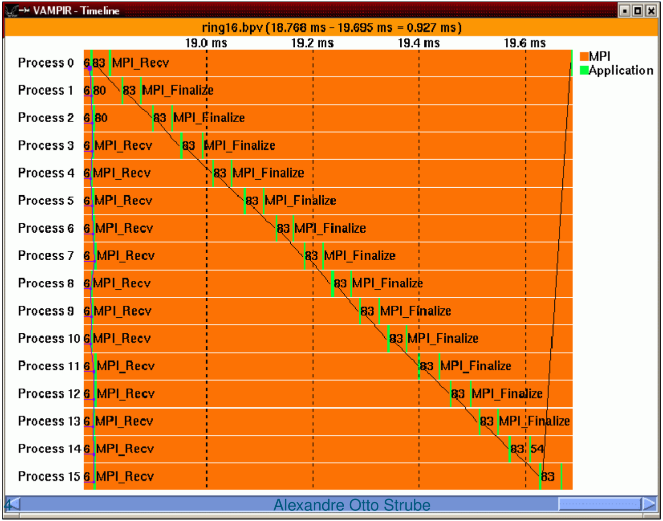
Scalasca: Features

- 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

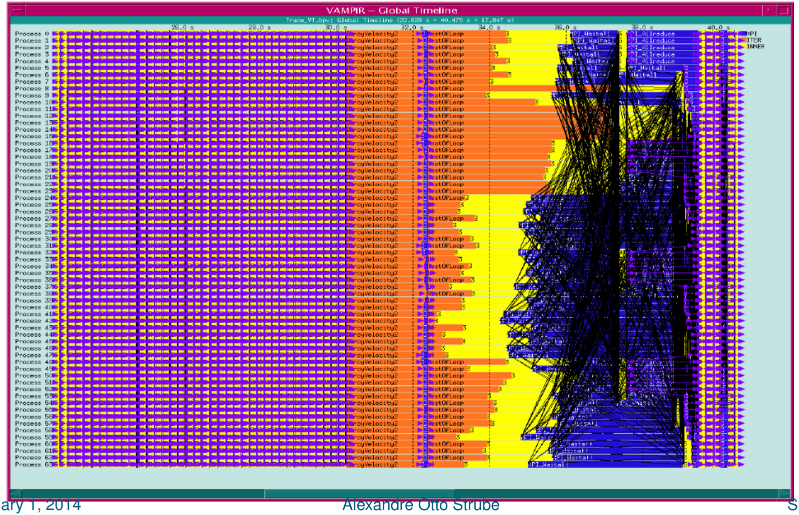
Scalasca: Features

- 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...


VAMPIR - Timeline
ring16.bpv (18.768 ms - 19.695 ms = 0.927 ms)
19.0 ms
19.2 ms
19.4 ms
19.6 ms
MPI
Application
Process 0 6 83 MPI_Recv
Process 1 6 80 83 MPI_Finalize
Process 2 6 80 83 MPI_Finalize
Process 3 6 MPI_Recv 83 MPI_Finalize
Process 4 6 MPI_Recv 83 MPI_Finalize
Process 5 6 MPI_Recv 83 MPI_Finalize
Process 6 6 MPI_Recv 83 MPI_Finalize
Process 7 6 MPI_Recv 83 MPI_Finalize
Process 8 6 MPI_Recv 83 MPI_Finalize
Process 9 6 MPI_Recv 83 MPI_Finalize
Process 10 6 MPI_Recv 83 MPI_Finalize
Process 11 6 MPI_Recv 83 MPI_Finalize
Process 12 6 MPI_Recv 83 MPI_Finalize
Process 13 6 MPI_Recv 83 MPI_Finalize
Process 14 6 MPI_Recv 83 54
Process 15 6 MPI_Recv 83

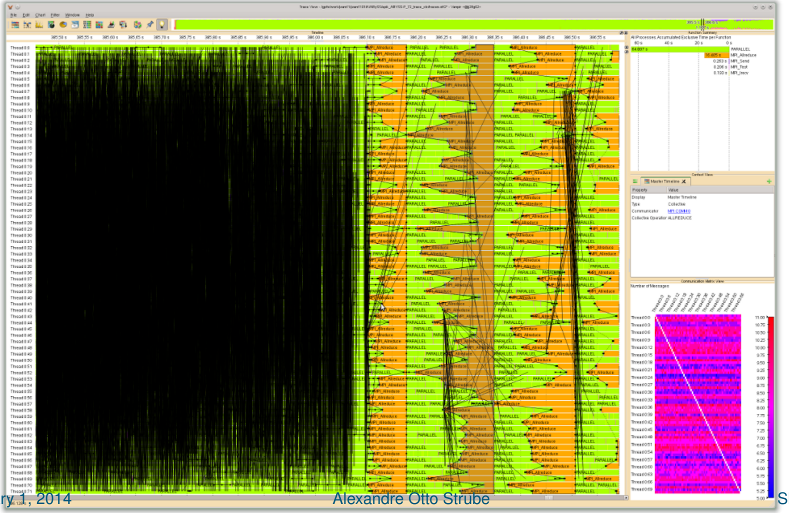
... but this is a real code.



VAMPIR – Global Timeline

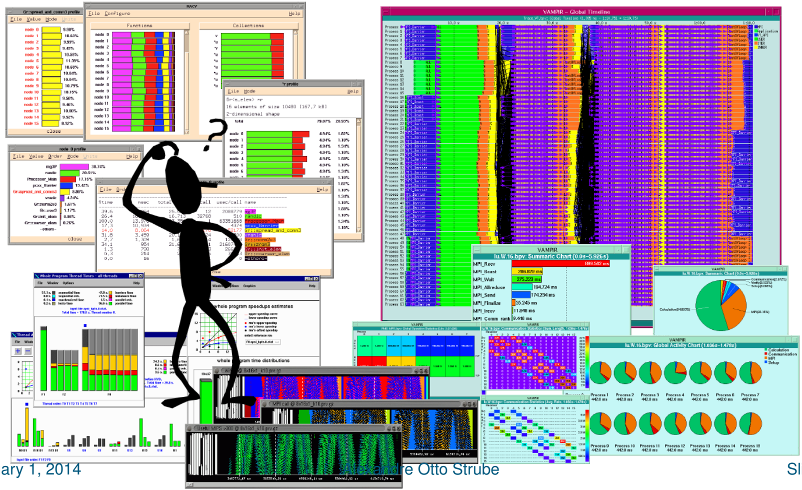
Trace_VT.bpv: Global Timeline (22.628 s - 40.475 s = 17.847 s)

and this.

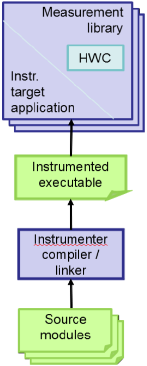


Property	Value
Display	Master Timeline
Type	Collective
Communicator	MPI COMM 0
Collective Operation	ALLREDUCE

... it can get really confusing.

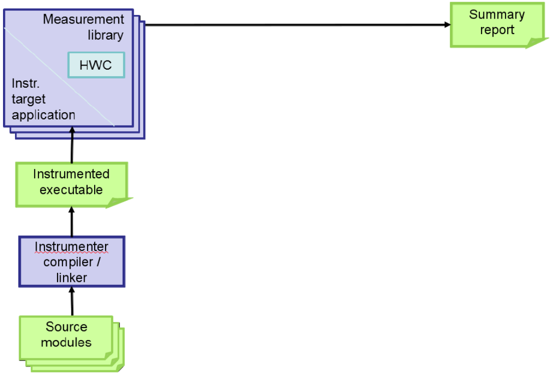

RACY
File Configure Help
Functions
Collections
Gr::spread_and_comm3 profile
node 0 profile
Whole Program Thread Times - all threads
whole program speedups estimates
whole program time distributions
VAMPIR – Global Timeline
VAMPIR
lu.W.16.bpv: Summary Chart (0.0s–5.926s)
MPI_Recv 899.582 ms
MPI_Bcast 286.829 ms
MPI_Wait 275.223 ms
MPI_Allreduce 194.724 ms
MPI_Send 174.234 ms
MPI_Finalize 35.245 ms
MPI_Irecv 11.848 ms
MPI_Comm_rank 9.446 ms
lu.W.16.bpv: Global Activity Chart (1.036s–1.478s)
Calculation
Communication
MPI
Setup
MPI call @ 8x16x1_k10.prv.gz
Useful MIPS >300 @ 8x16x1_k10.prv.gz

Scalasca

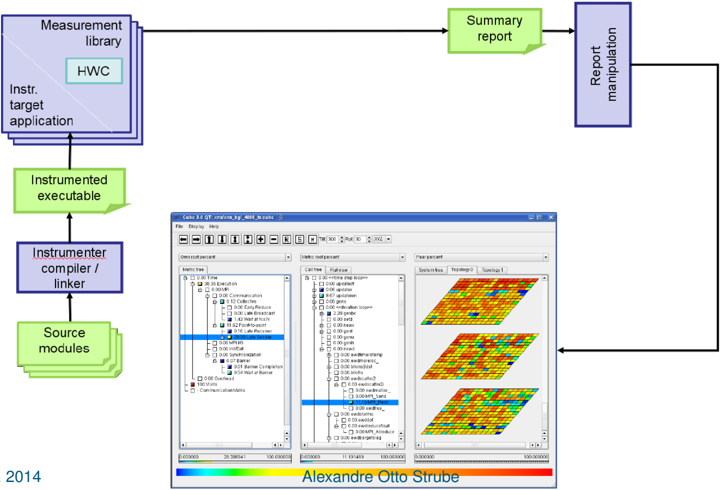


Measurement library
HWC
Instr. target application
Instrumented executable
Instrumenter compiler / linker
Source modules

Scalasca


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

Scalasca

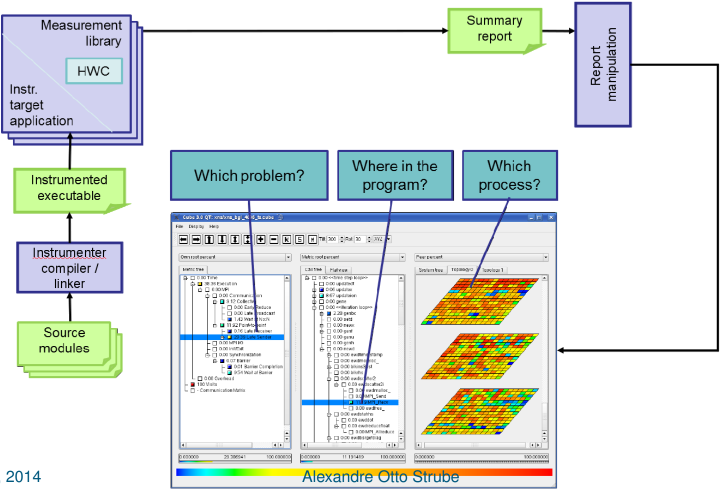


```
0.00 Time
  38.36 Execution
    0.00 MPI
      0.00 Communication
        9.12 Collective
          0.00 Early Reduce
          0.00 Late Broadcast
          1.43 Wait at NxN
        11.92 Point-to-point
          0.16 Late Receiver
          29.39 Late Sender
      0.00 MPI I/O
      0.00 Init/Exit
      0.00 Synchronization
        0.07 Barrier
          0.01 Barrier Completion
          9.54 Wait at Barrier
  0.00 Overhead
100 Visits
- Communication Matrix

0.000000    29.386941    100.000000
```

```
0.00 <<time step loop>>
  0.00 updatedt
  0.06 updatex
  8.67 updateien
  0.00 gene
  0.00 <<iteration loop>>
    2.28 genbc
    0.00 setd
    0.00 newx
    0.00 genf
    0.00 genu
    0.00 genh
    0.00 newd
      0.00 ewdtimerstamp
      0.00 ewdmoreloc_
      0.00 blkins3dst
      0.00 blkrhs
      0.00 ewdscatter2
        0.00 ewdscatter2i
          0.00 ewdmalloc_
          0.00 MPI_Send
          11.19 MPI_Recv
          0.00 ewdfree_
      0.00 ewdstatrhs
        0.00 ewddot
        0.00 ewdreducefloat
          0.00 MPI_Allreduce
      0.00 ewdbsrgetdiag

0.000000    11.191419    100.000000
```

```
0.000000    100.000000
```

Scalasca



Cube 3.0 QT: xns/xns_bgl_4096_ts.cube

File Display Help

Tilt 300 Rot 30 XYZ

Own root percent | Metric root percent | Peer percent

Metric tree

```
□ 0.00 Time
  □ 38.36 Execution
    □ 0.00 MPI
      □ 0.00 Communication
        □ 9.12 Collective
          □ 0.00 Early Reduce
          □ 0.00 Late Broadcast
          □ 1.43 Wait at NxN
        □ 11.92 Point-to-point
          □ 0.16 Late Receiver
          □ 29.39 Late Sender
      □ 0.00 MPI I/O
      □ 0.00 Init/Exit
      □ 0.00 Synchronization
        □ 0.07 Barrier
          □ 0.01 Barrier Completion
          □ 9.54 Wait at Barrier
    □ 0.00 Overhead
□ 100 Visits
□ - Communication Matrix
```

0.000000 29.386941 100.000000

Call tree | Flat view

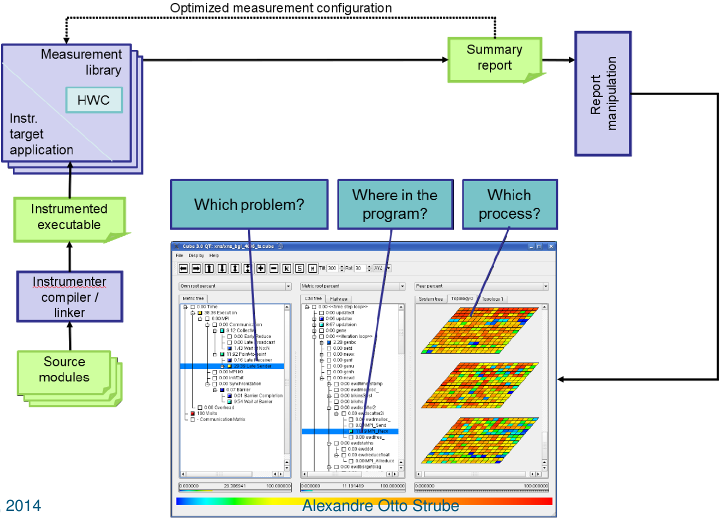
```
□ 0.00 <<time step loop>>
  □ 0.00 updatedt
  □ 0.06 updatex
  □ 8.67 updateien
  □ 0.00 gene
  □ 0.00 <<iteration loop>>
    □ 2.28 genbc
    □ 0.00 setd
    □ 0.00 newx
    □ 0.00 genf
    □ 0.00 genu
    □ 0.00 genh
    □ 0.00 newd
      □ 0.00 ewdtimestamp
      □ 0.00 ewdmalloc_
      □ 0.00 blkins3dst
      □ 0.00 blkrhs
      □ 0.00 ewdscatter2
        □ 0.00 ewdscatter2i
          □ 0.00 ewdmalloc_
          □ 0.00 MPI_Send
          □ 11.19 MPI_Recv
          □ 0.00 ewdfree_
      □ 0.00 ewdstatrhs
        □ 0.00 ewddot
        □ 0.00 ewdreducefloat
          □ 0.00 MPI_Allreduce
      □ 0.00 ewdbsrgetdiag
```

0.000000 11.191419 100.000000

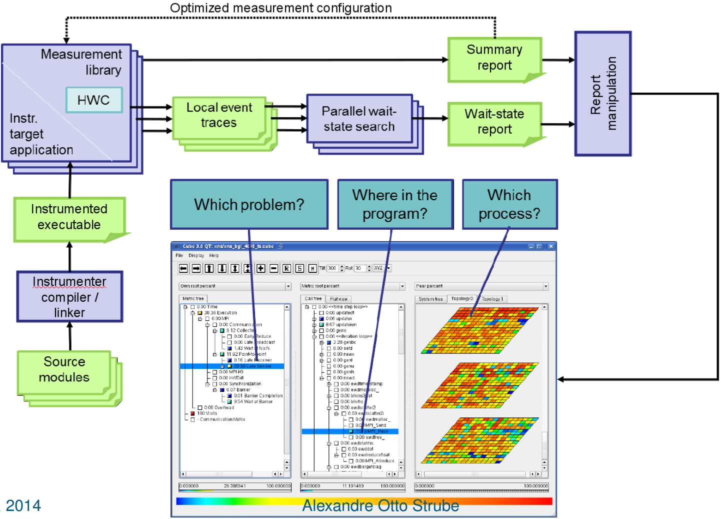
System tree | Topology 0 | Topology 1

0.000000 100.000000

Scalasca


Optimized measurement configuration
Measurement library
HWC
Instr. target application
Summary report
Report manipulation
Instrumented executable
Instrumenter compiler / linker
Source modules
Which problem?
Where in the program?
Which process?
Cube 3.0 QT: xns/xns_bgl_4096_ts.cube
File Display Help
Own root percent
Metric root percent
Peer percent
Metric tree
Call tree
Flat view
System tree
Topology 0
Topology 1

Scalasca



Optimized measurement configuration
Measurement library
HWC
Instr. target application
Summary report
Local event traces
Parallel wait-state search
Wait-state report
Report manipulation
Instrumented executable
Instrumenter compiler / linker
Source modules
Which problem?
Where in the program?
Which process?

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

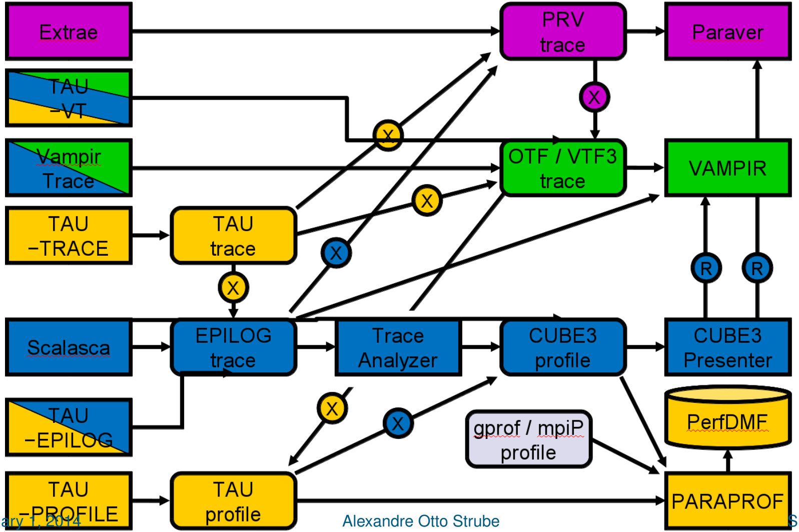
We’re not alone

- 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

- Communnity project with common infrastructure

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

Score-P project idea

- Communnity project with common infrastructure
- What we share:

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

Score-P project idea

- Communnity 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

- Communnity 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

- Communnity 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

- Communnity 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

- Communnity 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)

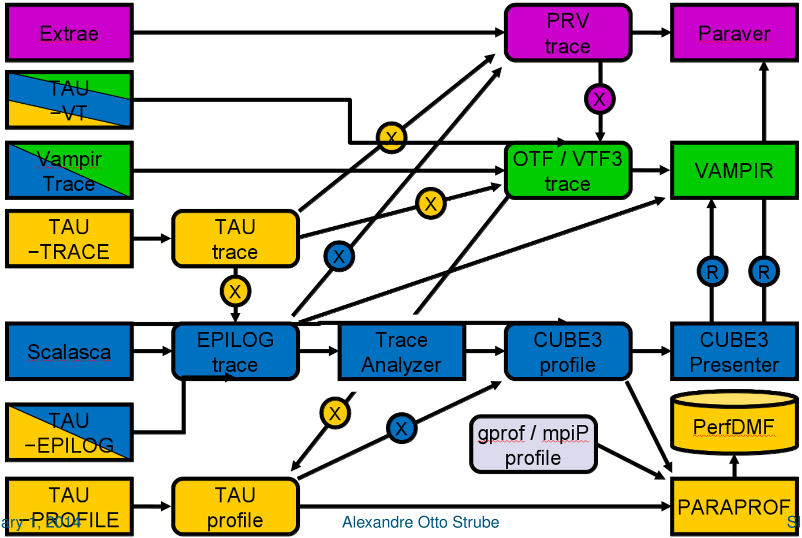
Who uses/develops Score-P?

- Scalasca (Fz-Juelich, RTWH Aachen)
- Vampir (TU Dresden)
- Periscope (Tu Munich)

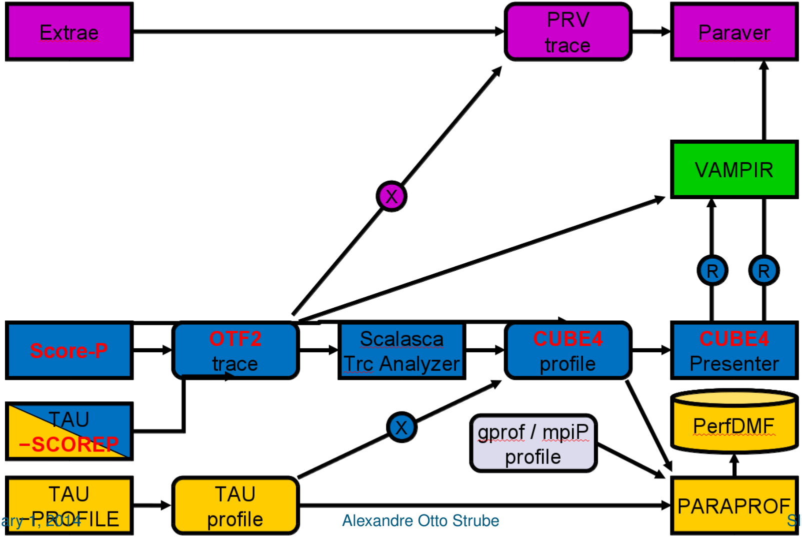
Who uses/develops Score-P?

- 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

What do we measure?

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

What do we measure?

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

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

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

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 inneficiency 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 ineficiency 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:

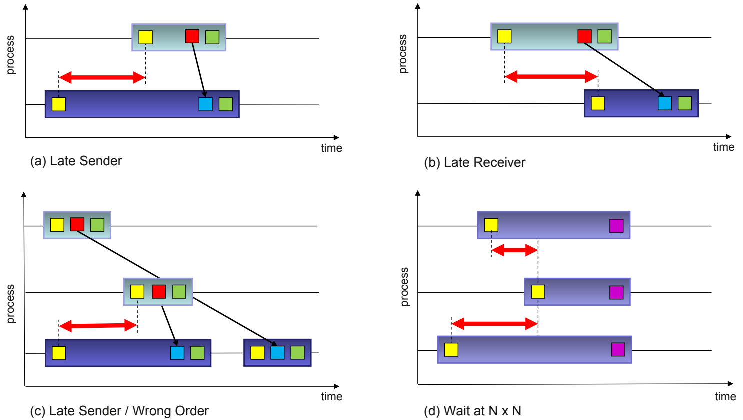
- Data collection/reduction
- Analysis:
  - Pattern search
  - Delay analysis
  - Critical-path analysis

Extreme scalability

All parallel:

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

Some MPI patterns



(a) Late Sender

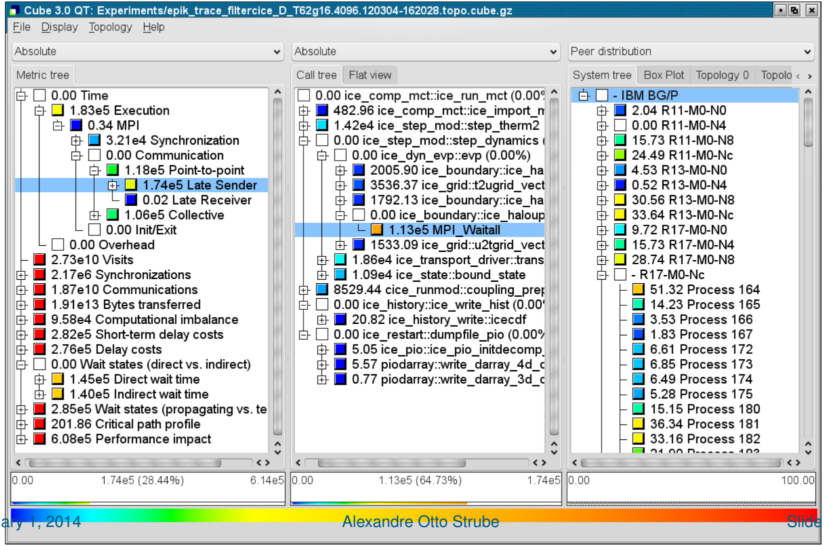
(b) Late Receiver

(c) Late Sender / Wrong Order

(d) Wait at N x N

🟨 ENTER 🟩 EXIT 🟥 SEND 🟦 RECV 🟪 COLLEXIT

Late sender

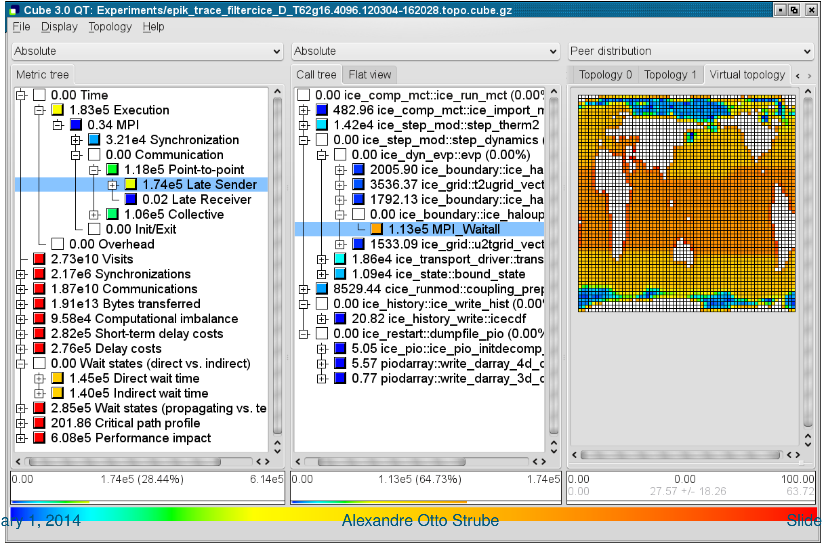


```
□ 0.00 Time
  🟨 1.83e5 Execution
    🟦 0.34 MPI
      🟦 3.21e4 Synchronization
      □ 0.00 Communication
        🟩 1.18e5 Point-to-point
          🟨 1.74e5 Late Sender
          🟦 0.02 Late Receiver
        🟩 1.06e5 Collective
      □ 0.00 Init/Exit
  □ 0.00 Overhead
🟥 2.73e10 Visits
🟥 2.17e6 Synchronizations
🟥 1.87e10 Communications
🟥 1.91e13 Bytes transferred
🟥 9.58e4 Computational imbalance
🟥 2.82e5 Short-term delay costs
🟥 2.76e5 Delay costs
□ 0.00 Wait states (direct vs. indirect)
  🟨 1.45e5 Direct wait time
  🟨 1.40e5 Indirect wait time
🟥 2.85e5 Wait states (propagating vs. te
🟥 201.86 Critical path profile
🟥 6.08e5 Performance impact

0.00        1.74e5 (28.44%)        6.14e5
```

```
□ 0.00 ice_comp_mct::ice_run_mct (0.00%
🟦 482.96 ice_comp_mct::ice_import_m
🟦 1.42e4 ice_step_mod::step_therm2
□ 0.00 ice_step_mod::step_dynamics (
  □ 0.00 ice_dyn_evp::evp (0.00%)
    🟦 2005.90 ice_boundary::ice_ha
    🟦 3536.37 ice_grid::t2ugrid_vect
    🟦 1792.13 ice_boundary::ice_ha
    □ 0.00 ice_boundary::ice_haloup
      🟧 1.13e5 MPI_Waitall
    🟦 1533.09 ice_grid::u2tgrid_vect
  🟦 1.86e4 ice_transport_driver::trans
  🟦 1.09e4 ice_state::bound_state
🟦 8529.44 cice_runmod::coupling_prep
□ 0.00 ice_history::ice_write_hist (0.00
  🟦 20.82 ice_history_write::icecdf
□ 0.00 ice_restart::dumpfile_pio (0.00%
  🟦 5.05 ice_pio::ice_pio_initdecomp_
  🟦 5.57 piodarray::write_darray_4d_c
  🟦 0.77 piodarray::write_darray_3d_c

0.00        1.13e5 (64.73%)        1.74e5
```

```
□ - IBM BG/P
  🟦 2.04 R11-M0-N0
  □ 0.00 R11-M0-N4
  🟩 15.73 R11-M0-N8
  🟩 24.49 R11-M0-Nc
  🟦 4.53 R13-M0-N0
  🟦 0.52 R13-M0-N4
  🟨 30.56 R13-M0-N8
  🟨 33.64 R13-M0-Nc
  🟦 9.72 R17-M0-N0
  🟩 15.73 R17-M0-N4
  🟨 28.74 R17-M0-N8
  □ - R17-M0-Nc
    🟨 51.32 Process 164
    🟩 14.23 Process 165
    🟦 3.53 Process 166
    🟦 1.83 Process 167
    🟦 6.61 Process 172
    🟦 6.85 Process 173
    🟦 6.49 Process 174
    🟦 5.28 Process 175
    🟩 15.15 Process 180
    🟨 36.34 Process 181
    🟨 33.16 Process 182

0.00                               100.00
```

Late sender and application topology



Cube 3.0 QT: Experiments/epik_trace_filtercice_D_T62g16.4096.120304-162028.topo.cube.gz

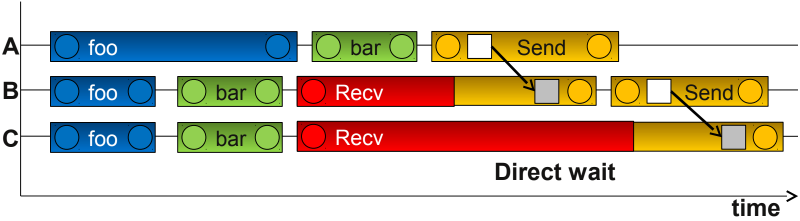
```
⊟ □ 0.00 Time
   ⊟ 🟨 1.83e5 Execution
      ⊟ 🟦 0.34 MPI
         ⊞ 🟦 3.21e4 Synchronization
         ⊟ □ 0.00 Communication
            ⊟ 🟩 1.18e5 Point-to-point
               ⊞ 🟨 1.74e5 Late Sender
                 🟦 0.02 Late Receiver
            ⊞ 🟩 1.06e5 Collective
           □ 0.00 Init/Exit
     □ 0.00 Overhead
  🟥 2.73e10 Visits
⊞ 🟥 2.17e6 Synchronizations
⊞ 🟥 1.87e10 Communications
⊞ 🟥 1.91e13 Bytes transferred
⊞ 🟥 9.58e4 Computational imbalance
⊞ 🟥 2.82e5 Short-term delay costs
⊞ 🟥 2.76e5 Delay costs
⊟ □ 0.00 Wait states (direct vs. indirect)
   ⊞ 🟨 1.45e5 Direct wait time
   ⊞ 🟨 1.40e5 Indirect wait time
⊞ 🟥 2.85e5 Wait states (propagating vs. te
⊞ 🟥 201.86 Critical path profile
⊞ 🟥 6.08e5 Performance impact
```

```
□ 0.00 ice_comp_mct::ice_run_mct (0.00
⊞ 🟦 482.96 ice_comp_mct::ice_import_m
⊞ 🟦 1.42e4 ice_step_mod::step_therm2
⊟ □ 0.00 ice_step_mod::step_dynamics (
   ⊟ □ 0.00 ice_dyn_evp::evp (0.00%)
      ⊞ 🟦 2005.90 ice_boundary::ice_ha
      ⊞ 🟦 3536.37 ice_grid::t2ugrid_vect
      ⊞ 🟦 1792.13 ice_boundary::ice_ha
      ⊟ □ 0.00 ice_boundary::ice_haloup
           🟧 1.13e5 MPI_Waitall
      ⊞ 🟦 1533.09 ice_grid::u2tgrid_vect
   ⊞ 🟦 1.86e4 ice_transport_driver::trans
   ⊞ 🟦 1.09e4 ice_state::bound_state
⊞ 🟦 8529.44 cice_runmod::coupling_prep
⊟ □ 0.00 ice_history::ice_write_hist (0.00
   ⊞ 🟦 20.82 ice_history_write::icecdf
⊟ □ 0.00 ice_restart::dumpfile_pio (0.00%
   ⊞ 🟦 5.05 ice_pio::ice_pio_initdecomp
   ⊞ 🟦 5.57 piodarray::write_darray_4d_c
   ⊞ 🟦 0.77 piodarray::write_darray_3d_c
```

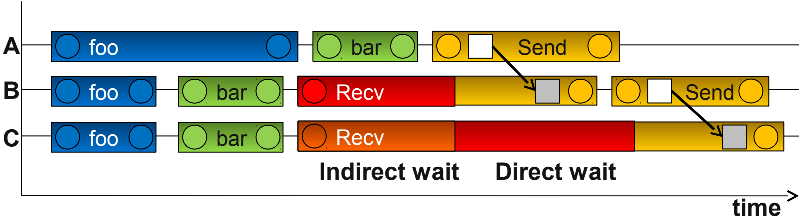
```
0.00        1.74e5 (28.44%)        6.14e5
0.00        1.13e5 (64.73%)        1.74e5
0.00        0.00                   100.00
0.00        27.57 +/- 18.26        63.72
```

]

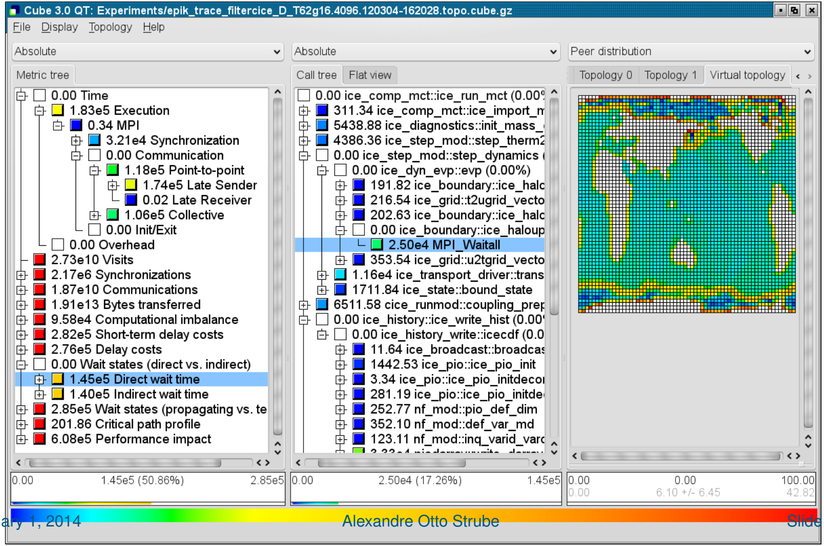
Direct wait time analysis



Indirect wait time analysis



Direct wait time



Cube 3.0 QT: Experiments/epik_trace_filtercice_D_T62g16.4096.120304-162028.topo.cube.gz

```
0.00 Time
  1.83e5 Execution
    0.34 MPI
      3.21e4 Synchronization
      0.00 Communication
        1.18e5 Point-to-point
          1.74e5 Late Sender
          0.02 Late Receiver
        1.06e5 Collective
      0.00 Init/Exit
  0.00 Overhead
2.73e10 Visits
2.17e6 Synchronizations
1.87e10 Communications
1.91e13 Bytes transferred
9.58e4 Computational imbalance
2.82e5 Short-term delay costs
2.76e5 Delay costs
0.00 Wait states (direct vs. indirect)
  1.45e5 Direct wait time
  1.40e5 Indirect wait time
2.85e5 Wait states (propagating vs. te
201.86 Critical path profile
6.08e5 Performance impact
```

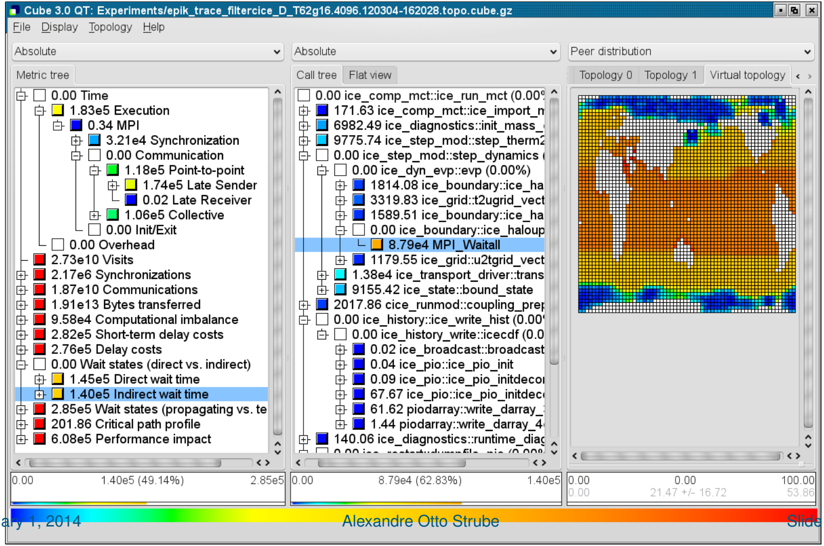
```
0.00 ice_comp_mct::ice_run_mct (0.00%
311.34 ice_comp_mct::ice_import_m
5438.88 ice_diagnostics::init_mass_
4386.36 ice_step_mod::step_therm2
0.00 ice_step_mod::step_dynamics (
  0.00 ice_dyn_evp::evp (0.00%)
    191.82 ice_boundary::ice_halo
    216.54 ice_grid::t2ugrid_vecto
    202.63 ice_boundary::ice_halo
    0.00 ice_boundary::ice_haloup
      2.50e4 MPI_Waitall
    353.54 ice_grid::u2tgrid_vecto
  1.16e4 ice_transport_driver::trans
  1711.84 ice_state::bound_state
6511.58 cice_runmod::coupling_prep
0.00 ice_history::ice_write_hist (0.00%
  0.00 ice_history_write::icecdf (0.0
    11.64 ice_broadcast::broadcas
    1442.53 ice_pio::ice_pio_init
    3.34 ice_pio::ice_pio_initdecom
    281.19 ice_pio::ice_pio_initdec
    252.77 nf_mod::pio_def_dim
    352.10 nf_mod::def_var_md
    123.11 nf_mod::inq_varid_varc
```

0.00	1.45e5 (50.86%)	2.85e5

0.00	2.50e4 (17.26%)	1.45e5

0.00	0.00	100.00
0.00	6.10 +/- 6.45	42.82

Indirect wait time analysis



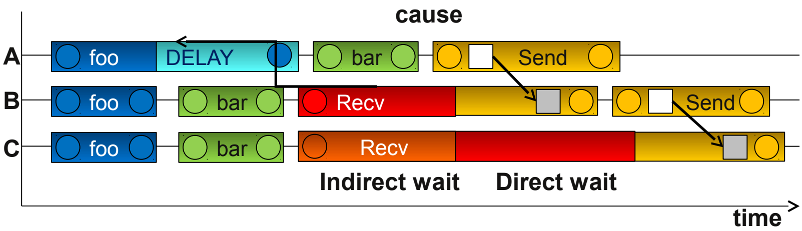
Cube 3.0 QT: Experiments/epik_trace_filtercice_D_T62g16.4096.120304-162028.topo.cube.gz

```
0.00 Time
  1.83e5 Execution
    0.34 MPI
      3.21e4 Synchronization
      0.00 Communication
        1.18e5 Point-to-point
          1.74e5 Late Sender
          0.02 Late Receiver
        1.06e5 Collective
      0.00 Init/Exit
  0.00 Overhead
2.73e10 Visits
2.17e6 Synchronizations
1.87e10 Communications
1.91e13 Bytes transferred
9.58e4 Computational imbalance
2.82e5 Short-term delay costs
2.76e5 Delay costs
0.00 Wait states (direct vs. indirect)
  1.45e5 Direct wait time
  1.40e5 Indirect wait time
2.85e5 Wait states (propagating vs. te
201.86 Critical path profile
6.08e5 Performance impact
```

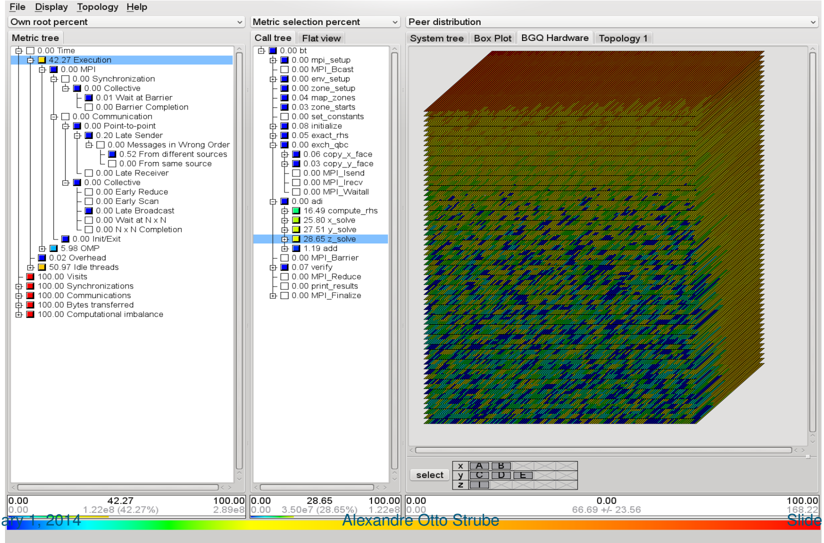
```
0.00 ice_comp_mct::ice_run_mct (0.00%
171.63 ice_comp_mct::ice_import_m
6982.49 ice_diagnostics::init_mass_
9775.74 ice_step_mod::step_therm2
0.00 ice_step_mod::step_dynamics (
  0.00 ice_dyn_evp::evp (0.00%)
    1814.08 ice_boundary::ice_ha
    3319.83 ice_grid::t2ugrid_vect
    1589.51 ice_boundary::ice_ha
    0.00 ice_boundary::ice_haloup
      8.79e4 MPI_Waitall
    1179.55 ice_grid::u2tgrid_vect
  1.38e4 ice_transport_driver::trans
  9155.42 ice_state::bound_state
2017.86 cice_runmod::coupling_prep
0.00 ice_history::ice_write_hist (0.00
  0.00 ice_history_write::icecdf (0.0
    0.02 ice_broadcast::broadcast
    0.04 ice_pio::ice_pio_init
    0.09 ice_pio::ice_pio_initdeco
    67.67 ice_pio::ice_pio_initdec
    61.62 piodarray::write_darray_
    1.44 piodarray::write_darray_4
140.06 ice_diagnostics::runtime_diag
0.00 ice_restartdumpfile_pio (0.00%
```

```
0.00        1.40e5 (49.14%)      2.85e5
0.00        8.79e4 (62.83%)      1.40e5
0.00        0.00                 100.00
0.00        21.47 +/- 16.72      53.86
```

Root cause analysis



6D Hardware topology



```
□ 0.00 Time
  42.27 Execution
    0.00 MPI
      □ 0.00 Synchronization
        0.00 Collective
          0.01 Wait at Barrier
          □ 0.00 Barrier Completion
      □ 0.00 Communication
        0.00 Point-to-point
          0.20 Late Sender
            □ 0.00 Messages in Wrong Order
              0.52 From different sources
              □ 0.00 From same source
          □ 0.00 Late Receiver
        0.00 Collective
          □ 0.00 Early Reduce
          □ 0.00 Early Scan
          0.00 Late Broadcast
          □ 0.00 Wait at N x N
          □ 0.00 N x N Completion
      0.00 Init/Exit
    5.98 OMP
  0.02 Overhead
  50.97 Idle threads
100.00 Visits
100.00 Synchronizations
100.00 Communications
100.00 Bytes transferred
100.00 Computational imbalance
```

```
0.00 bt
  0.00 mpi_setup
  □ 0.00 MPI_Bcast
  0.00 env_setup
  0.00 zone_setup
  0.04 map_zones
  0.03 zone_starts
  □ 0.00 set_constants
  0.08 initialize
  0.05 exact_rhs
  0.00 exch_qbc
    0.06 copy_x_face
    0.03 copy_y_face
    □ 0.00 MPI_Isend
    □ 0.00 MPI_Irecv
    □ 0.00 MPI_Waitall
  0.00 adi
    16.49 compute_rhs
    25.80 x_solve
    27.51 y_solve
    28.65 z_solve
    1.19 add
  □ 0.00 MPI_Barrier
  0.07 verify
  □ 0.00 MPI_Reduce
  □ 0.00 print_results
  □ 0.00 MPI_Finalize
```

0.00	42.27	100.00
0.00	1.22e8 (42.27%)	2.89e8

0.00	28.65	100.00
0.00	3.50e7 (28.65%)	1.22e8

0.00	0.00	100.00
0.00	66.69 +/- 23.56	168.22

The Future

Member of the Helmholtz-Association

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
REG. U.S. PAT. OFF.
"That's all Folks!"
A WARNER BROS. CARTOON
A VITAPHONE® RELEASE