German Research School for Simulation Sciences

Recent Scalasca Research

Felix Wolf

02-08-2011

JÜLICH FORSCHUNGSZENTRUM

RWTH AACHEN UNIVERSITY

German Research School
for Simulation Sciences

- Joint venture of
  - Forschungszentrum Jülich
  - RWTH Aachen University
- Four research laboratories
  - Computational biophysics
  - Computational engineering
  - Computational materials science
  - Parallel Programming
- Education
  - M.Sc. in simulation Sciences
  - Ph.D. program
- About 50 scientific staff members

Aachen

Jülich

Rheinisch-Westfälische Technische Hochschule Aachen

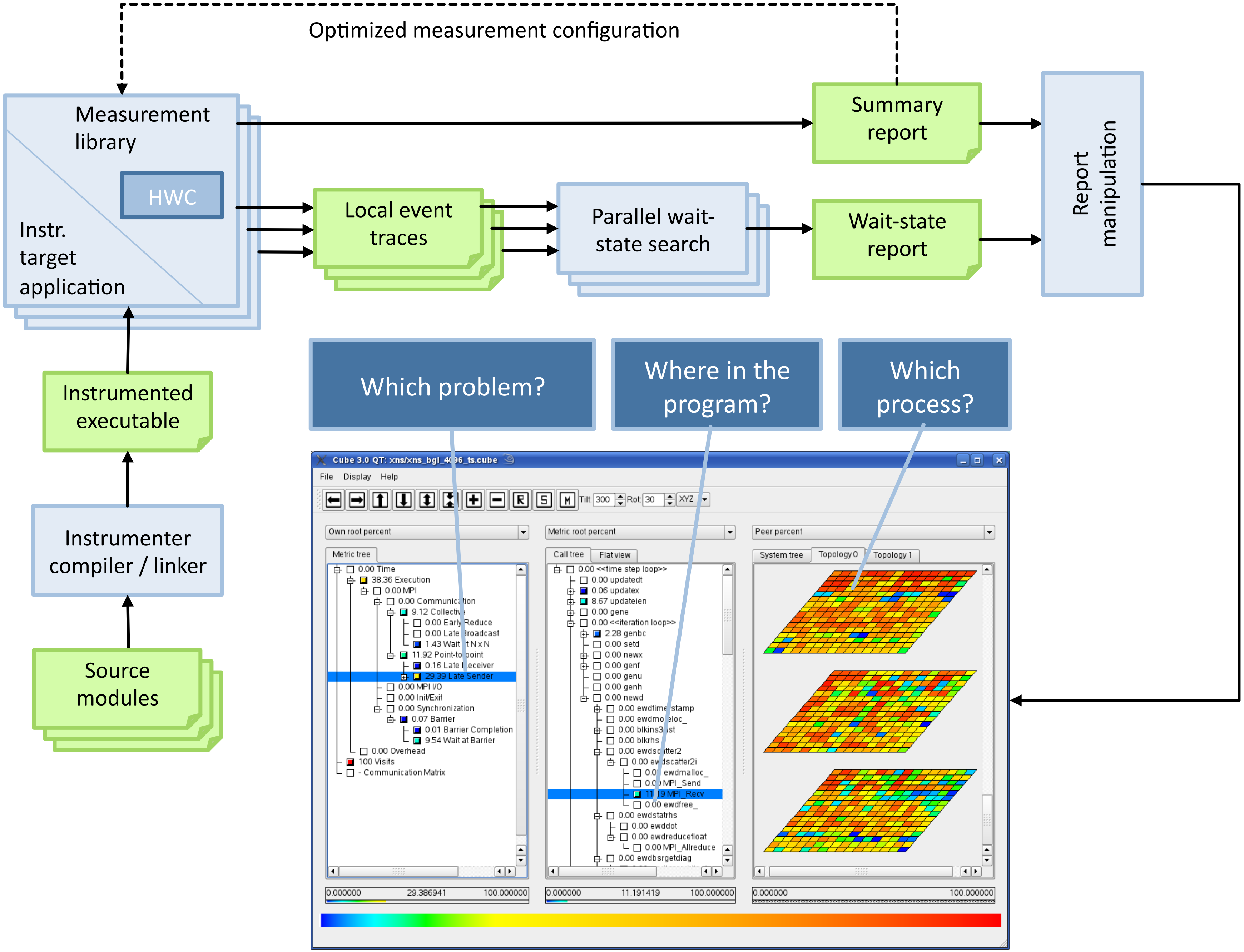
- Strong focus on engineering
- ~ 200 M€ third-party funding per year
- Around 31,000 students in over 100 academic programs
- > 5,000 international students from 120 different countries
- Cooperates with Jülich within the Jülich Aachen Research Alliance (JARA)

RWTHAACHEN UNIVERSITY

scalasca

- Scalable performance-analysis toolset for parallel codes
  - Focus on communication & synchronization
- Integrated performance analysis process
  - Performance overview on call-path level via runtime summarization
  - In-depth study of application behavior via event tracing
- Supported programming models
  - MPI-1, MPI-2 one-sided communication
  - OpenMP (basic features)
- Available for all major HPC platforms

Joint project of JÜLICH FORSCHUNGSZENTRUM German Research School for Simulation Sciences

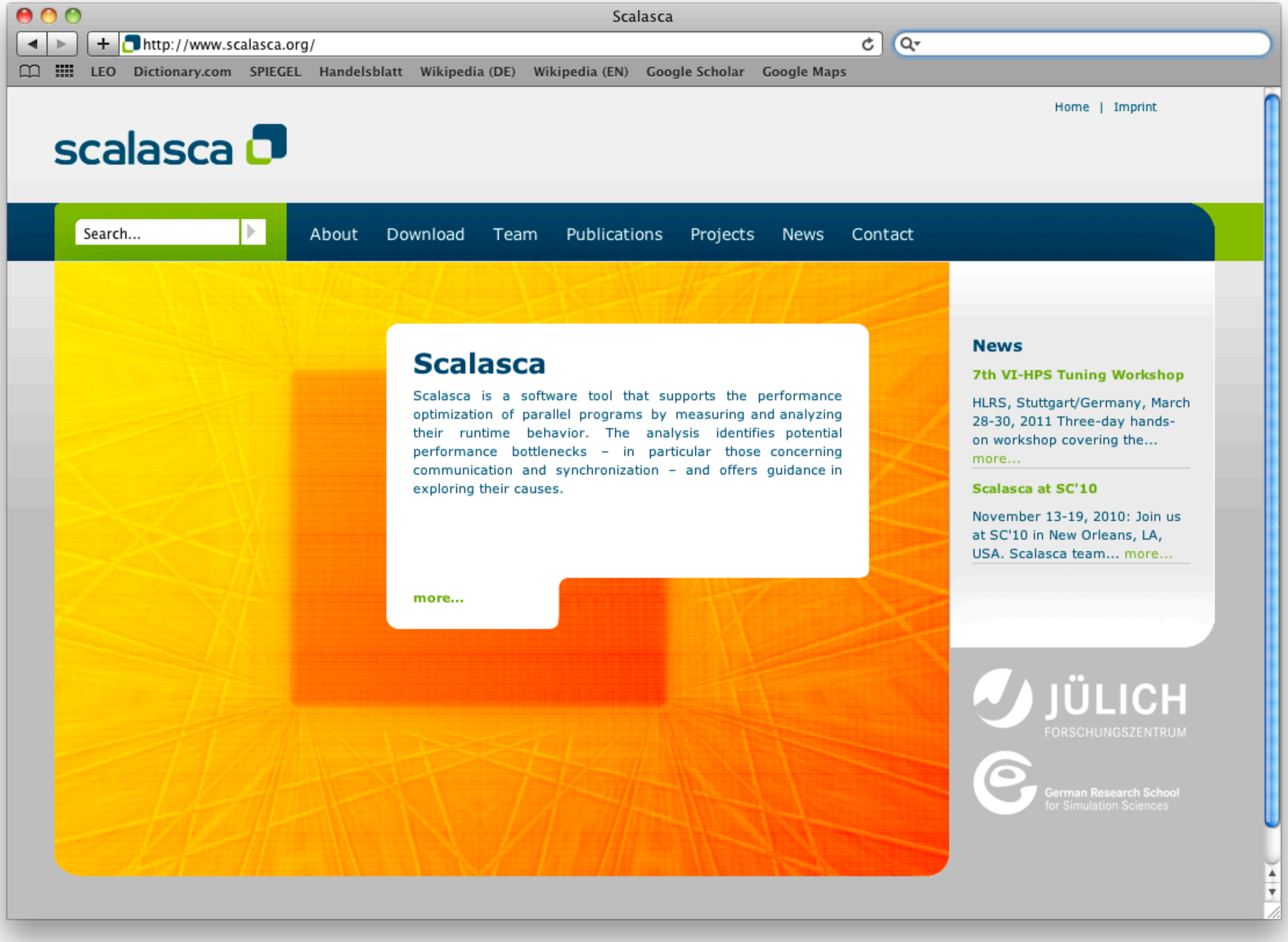


```
0.00 Time
  38.36 Execution
    0.00 MPI
      0.00 Communication
        9.12 Collective
          0.00 Early Reduce
          0.00 Late Broadcast
          1.43 Wait at N x N
        11.92 Point-to-point
          0.16 Late Receiver
          29.39 Late Sender
      0.00 MPI I/O
      0.00 Init/Exit
      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 ewdtimestamp
      0.00 ewdmoveloc_
      0.00 blkins3lst
      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
```

www.scalasca.org



Scalasca
http://www.scalasca.org/
LEO Dictionary.com SPIEGEL Handelsblatt Wikipedia (DE) Wikipedia (EN) Google Scholar Google Maps
Home | Imprint
scalasca
Search...
About Download Team Publications Projects News Contact

Scalasca

Scalasca is a software tool that supports the performance optimization of parallel programs by measuring and analyzing their runtime behavior. The analysis identifies potential performance bottlenecks – in particular those concerning communication and synchronization – and offers guidance in exploring their causes.

more...

News

7th VI-HPS Tuning Workshop

HLRS, Stuttgart/Germany, March 28-30, 2011 Three-day hands-on workshop covering the... more...

Scalasca at SC'10

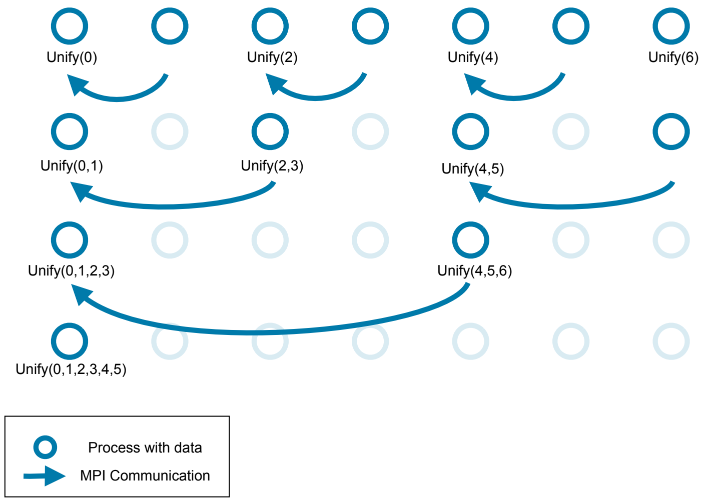
November 13-19, 2010: Join us at SC'10 in New Orleans, LA, USA. Scalasca team... more...

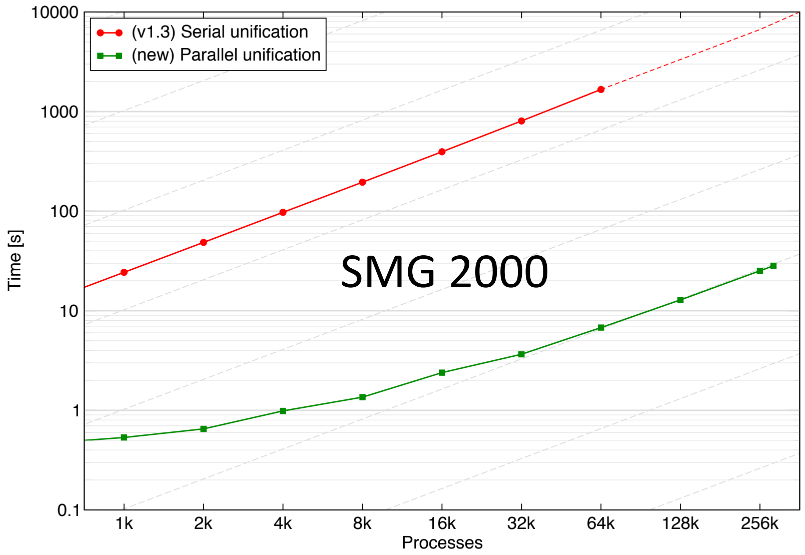
Outline

- Recent scalability improvements
- Mapping wait states onto their root cause
- Two approaches to low-overhead MPI profiling
  - Low-overhead direct instrumentation using prior static analysis
  - Reconciling sampling and direct instrumentation

Hierarchical unification

- Unification maps local identifiers of regions, call paths etc. onto global ones
  - Generation of a unified set of global definitions
  - Generation of local-to-global identifier mappings for each process
  - Writing the global definitions and the identifier mappings to disk.



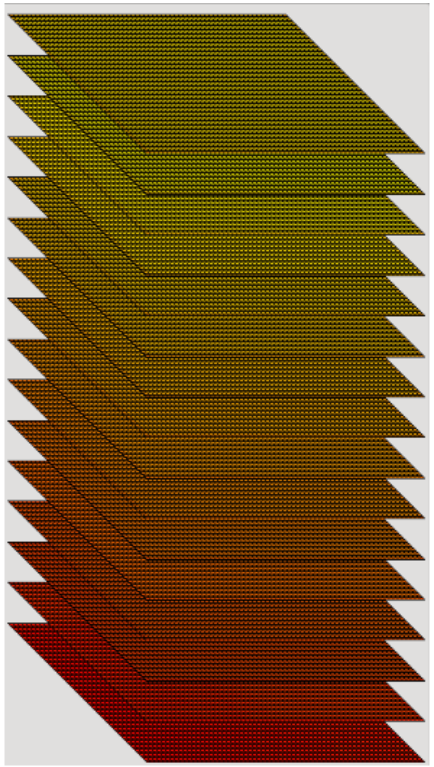


Markus Geimer et al.: Further improving the scalability of the Scalasca toolset. In Proc. of PARA 2010: Reykjavik, Iceland, June 6–9 2010, Springer, 2011. (to appear).

Communicator management

- Scalasca records communicators in event traces
  - Needed for trace replay
- Previous method created multiple types of overhead
  - Memory due to replication of data across processes
  - Measurement dilation due to runtime rank translation
  - Unification of local communicator IDs
- New method creates global comm. ID at runtime
  - Stores information only once per communicator
  - Avoids runtime rank translation by storing translation tables at the end
  - Essentially eliminates unification

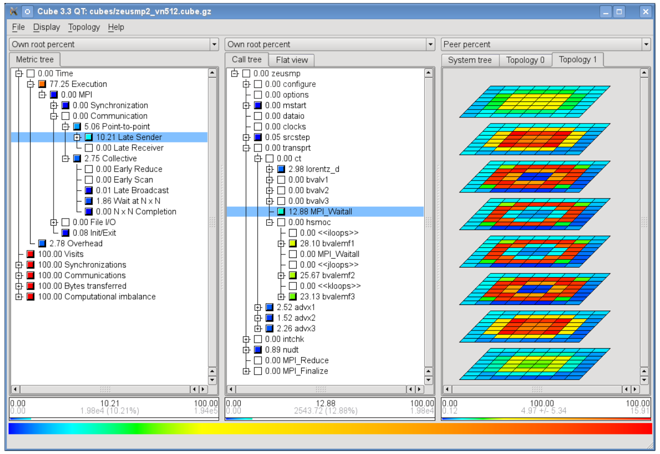
Distorted waiting times in PFLOATRAN on Jugene



Markus Geimer et al.: Scaling Performance Tool MPI Communicator Management. In Proc. of the 18th European MPI Users' Group Meeting (EuroMPI), Santorini, Greece, Springer, 2011. (to appear)

Incremental loading of report

- Currently, the entire report is loaded in to the GUI in one piece
  - Severe limitation of interactive experience
- In the future, data sets exceeding a certain size will be loaded incrementally
- At 288k processes, initial loading time reduced from 200s to 4s in prototype



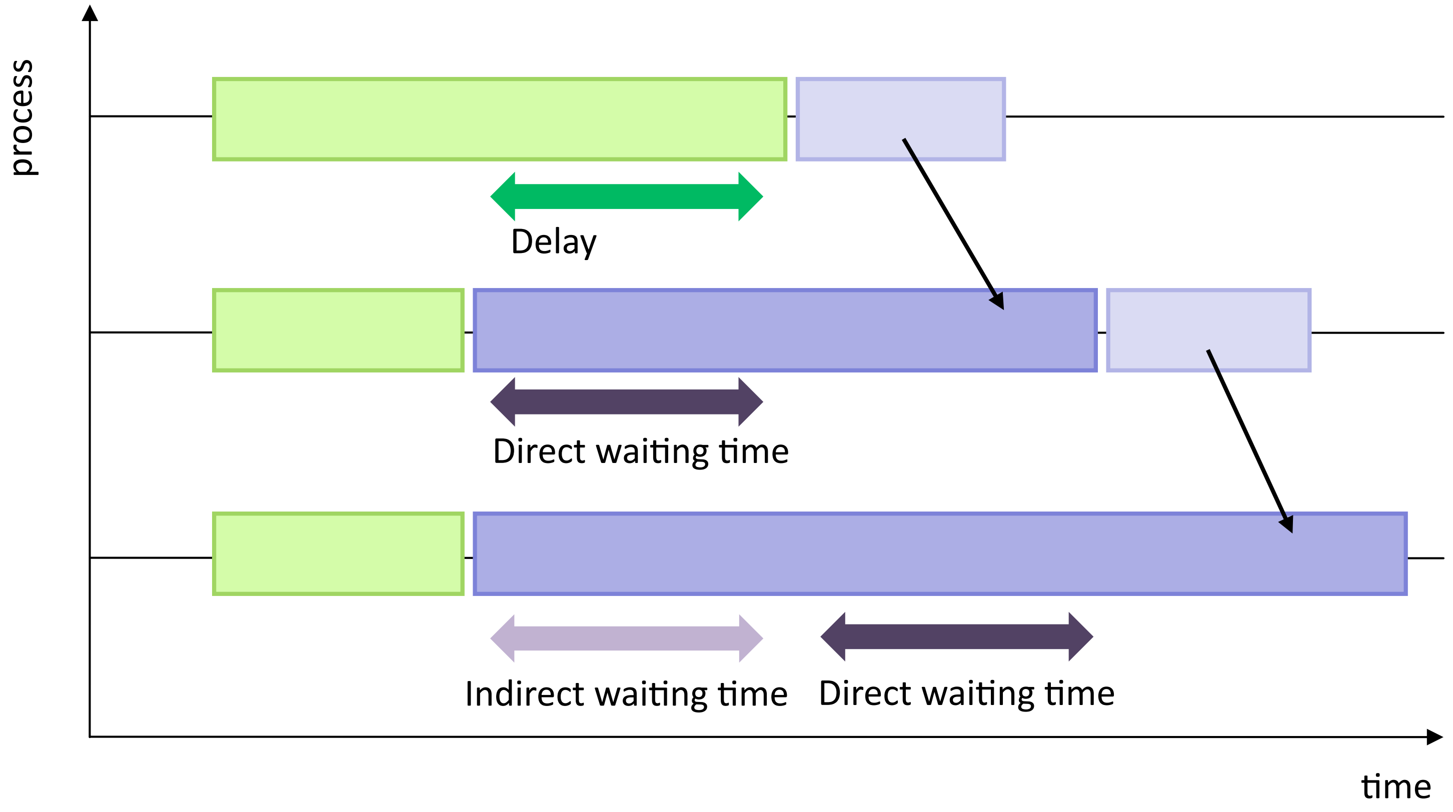
```
0.00 Time
  77.25 Execution
    0.00 MPI
      0.00 Synchronization
      0.00 Communication
        5.06 Point-to-point
          10.21 Late Sender
          0.00 Late Receiver
        2.75 Collective
          0.00 Early Reduce
          0.00 Early Scan
          0.01 Late Broadcast
          1.86 Wait at N x N
          0.00 N x N Completion
      0.00 File I/O
      0.08 Init/Exit
  2.78 Overhead
100.00 Visits
100.00 Synchronizations
100.00 Communications
100.00 Bytes transferred
100.00 Computational imbalance

0.00                10.21               100.00
0.00        1.98e4 (10.21%)             1.94e5
```

```
0.00 zeusmp
  0.00 configure
  0.00 options
  0.00 mstart
  0.00 dataio
  0.00 clocks
  0.05 srcstep
  0.00 transprt
    0.00 ct
      2.98 lorentz_d
      0.00 bvalv1
      0.00 bvalv2
      0.00 bvalv3
      12.88 MPI_Waitall
      0.00 hsmoc
        0.00 <<iloops>>
        28.10 bvalemf1
        0.00 MPI_Waitall
        0.00 <<jloops>>
        25.67 bvalemf2
        0.00 <<kloops>>
        23.13 bvalemf3
    2.52 advx1
    1.52 advx2
    2.26 advx3
  0.00 intchk
  0.89 nudt
  0.00 MPI_Reduce
  0.00 MPI_Finalize

0.00                12.88               100.00
0.00        2543.72 (12.88%)            1.98e4
```

```
0.00                100.00              100.00
0.12            4.97 +/- 5.34            15.91
```

Markus Geimer et al.: Further improving the scalability of the Scalasca toolset. In Proc. of PARA 2010: Reykjavik, Iceland, June 6–9 2010, Springer, 2011. (to appear).

Propagation of waiting time

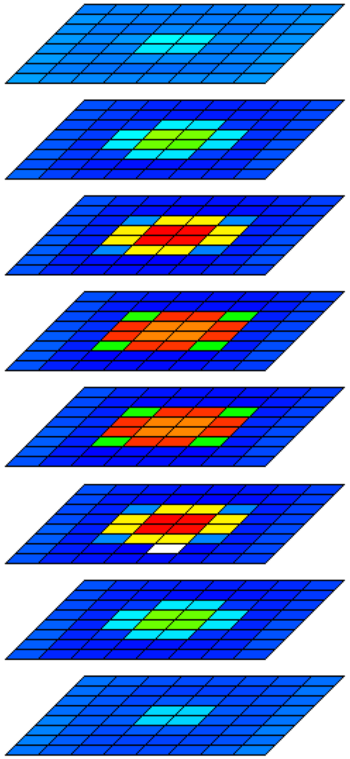


Identifying delays in traces and assessing their cost

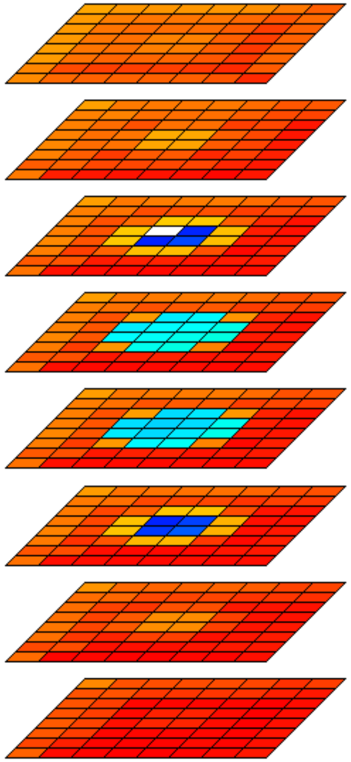
- Essentially scalable version of Meira Jr. et al.
- Classification of waiting times into
  - Direct vs. indirect
  - Propagating vs. terminal
- Attributes costs of wait states to delay intervals
  - Requires forward and backward replay

David Böhme et al.: Identifying the root causes of wait states in large-scale parallel applications. In Proc. of the 39th International Conference on Parallel Processing (ICPP), San Diego, CA,, IEEE Computer Society, September 2010.
Best Paper Award

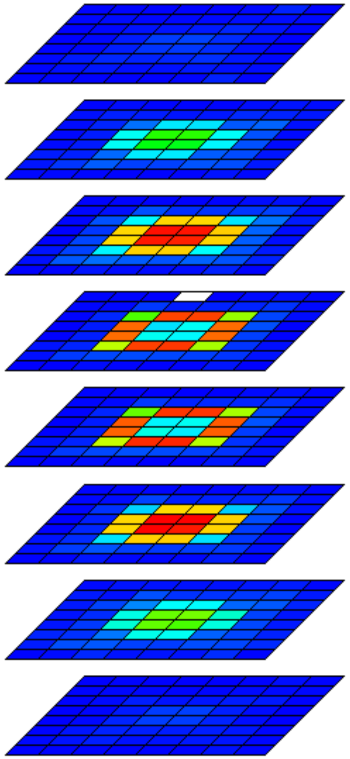
Origin of delay costs in Zeus-MP/2



Computation



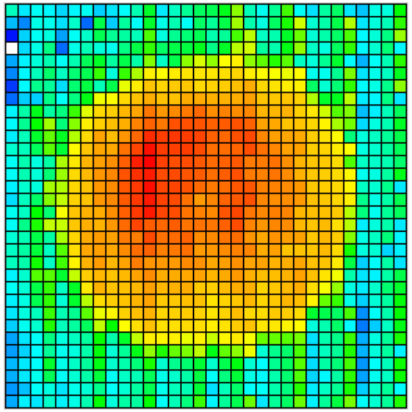
Waiting time



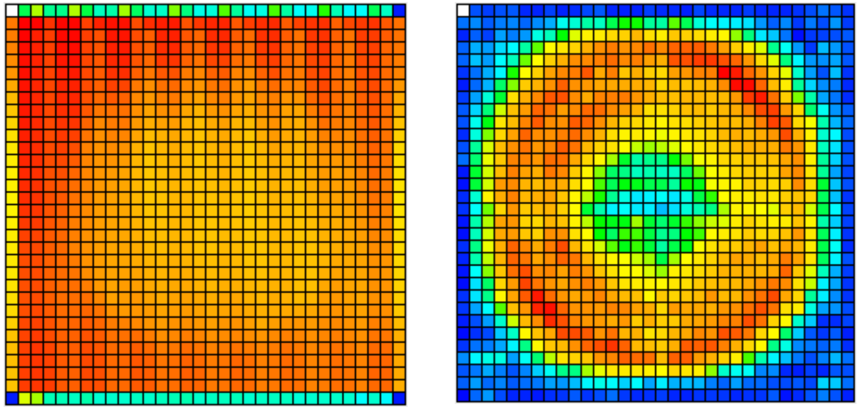
Delay costs

Delay analysis of code Illumination

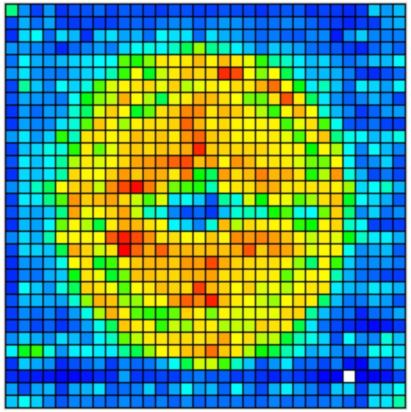
- Particle physics code (laser-plasma interaction)
- Delay analysis identified inefficient communication behavior as cause of wait states



Computation



Propagating wait states:
Original vs. optimized code



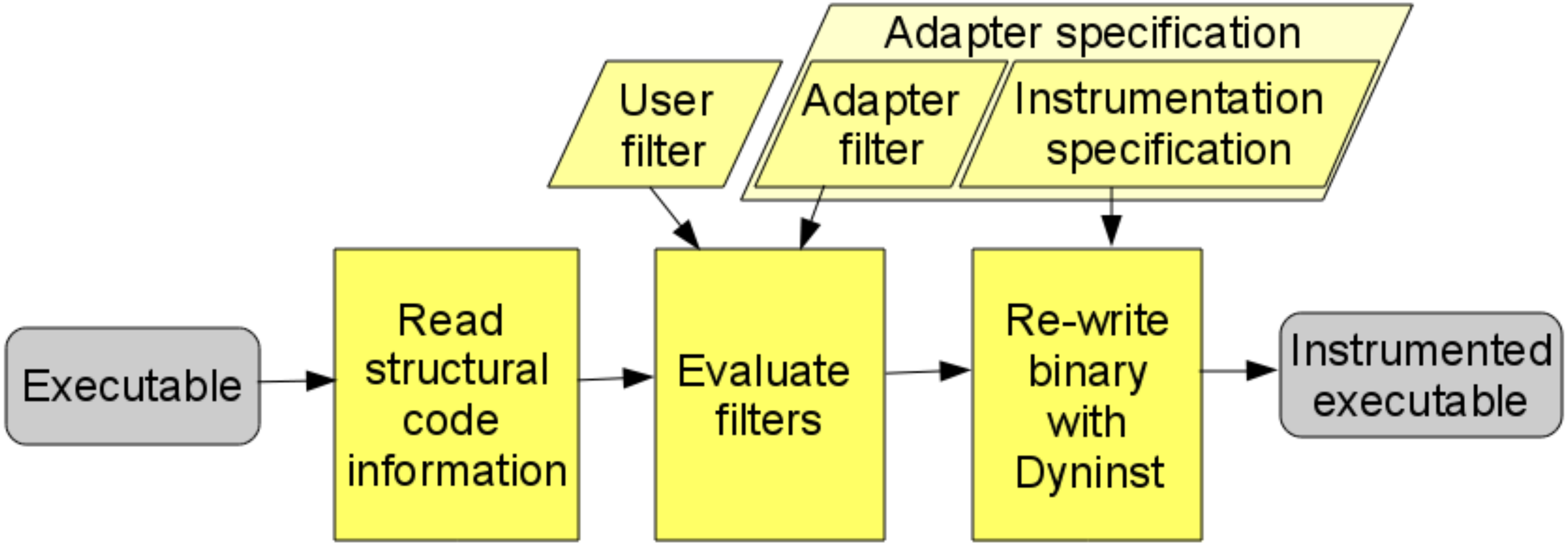
Costs of direct delay
in optimized code

Direct instrumentation

- Every entry or exit event of an instrumented function causes the measurement system to be called
- Even instrumentation of all user functions plus MPI wrappers results in reasonable overhead (< 15%) in many cases
- In others, overhead can be excessive
  - Especially C++ codes with their many tiny methods
- Filtering (black/white listing) can help reduce overhead
  - Static: filtered functions are not instrumented in the first place
  - Dynamic: invocation of the measurement system suppressed
- Tradeoff: lower overhead but loss of information

Configurable binary instrumenter (Cobi)

- Generating the filter usually requires extra run
- Idea: determine filter based on prior static analysis
  - Rules to specify analysis objectives (i.e., limit loss of information)



Cobi - filters

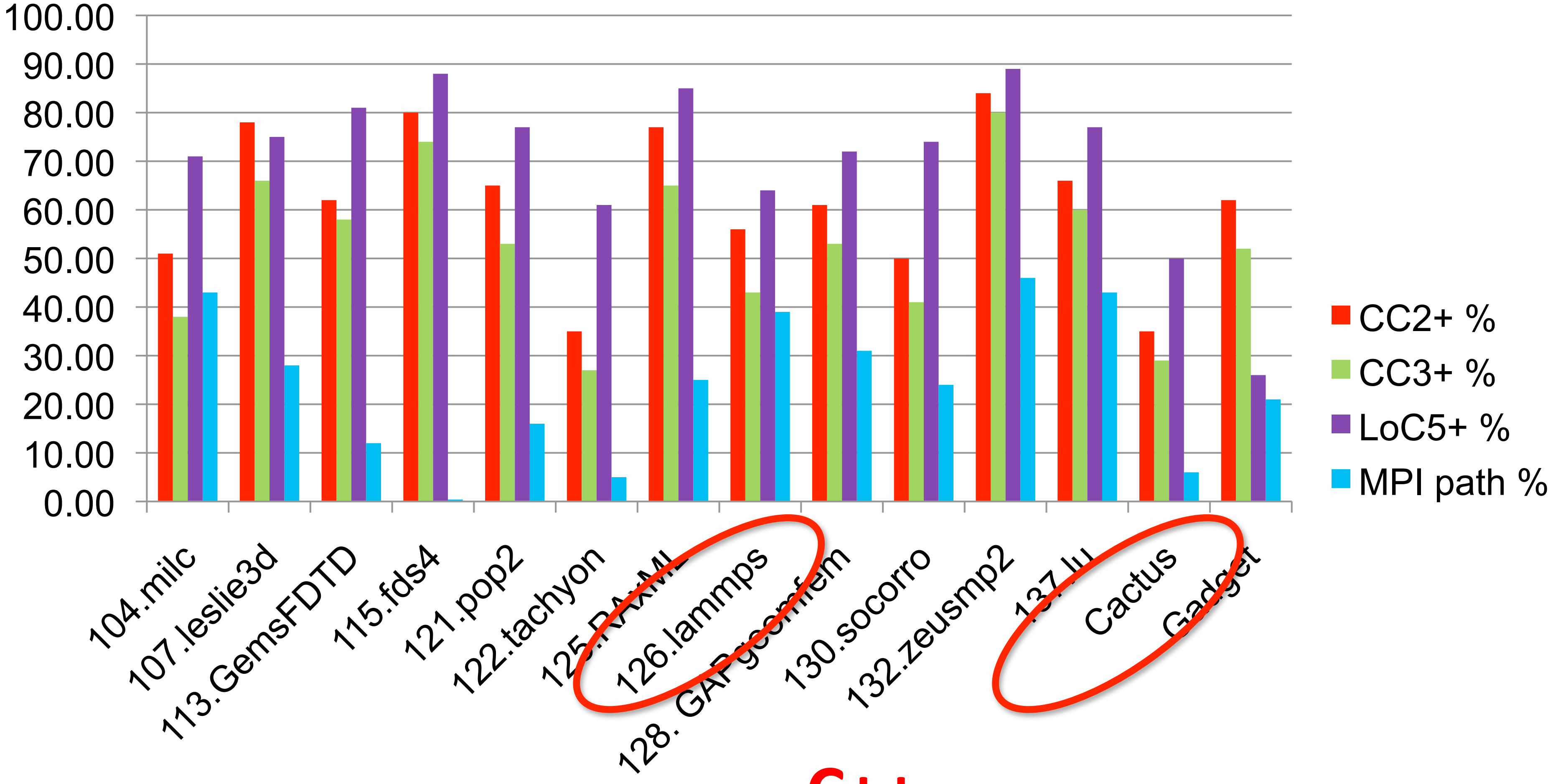
- Are specified in a separate XML file
- Start with all or no function
- Include/exclude functions with filter rules
- Rules can be combined by logical operators

- Are on call path
- Lines of source code
- Cyclomatic complexity
- Number of instructions
- Number and nesting level of loops
- Number of function calls
- Depth in call tree
- Name matching
- Prefix
- Suffix

Possible rules

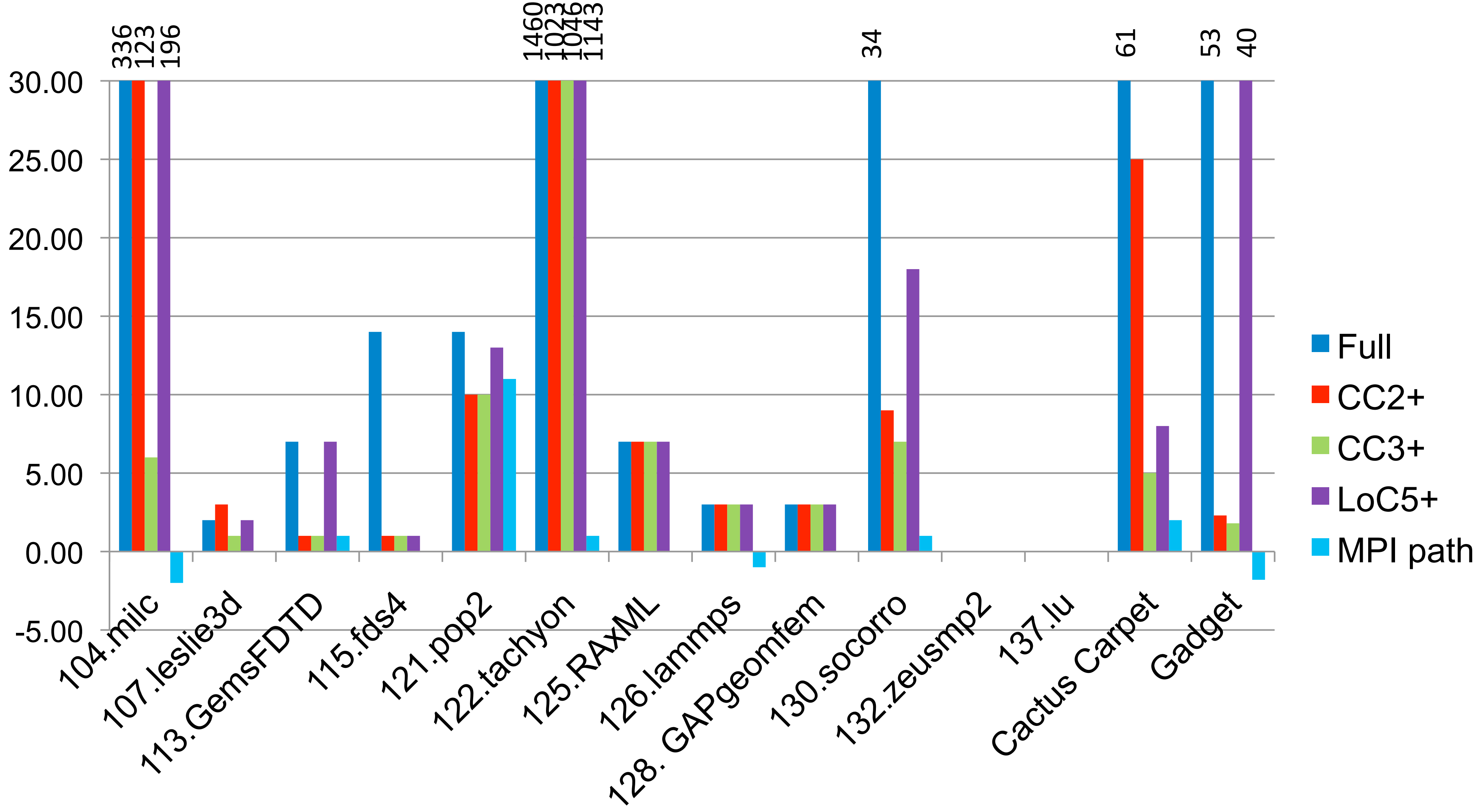
Jan Mußler et al.: Reducing the overhead of direct application instrumentation using prior static analysis. In Proc. of the Euro-Par Conference, Bordeaux, France, Springer, 2011. (to appear)

Instrumented fraction of functions with filters



C++

Runtime overhead in percent



Reconciling sampling and direct instrumentation

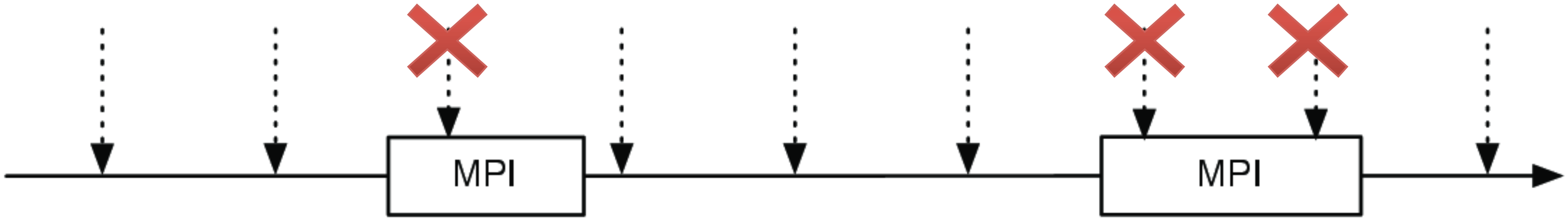
- Sampling allows better control of overhead
- But may miss critical events
  - Hard to capture accurate communication metrics
- New hybrid approach
  - Applies low-overhead sampling to user code
  - Intercepts MPI calls via direct instrumentation
  - Relies on efficient stack unwinding
  - Integrates measurements in statistically sound manner

Zoltan Szebenyi et al.: Reconciling sampling and direct instrumentation for unintrusive call-path profiling of MPI programs. In Proc. of the International Parallel and Distributed Processing Symposium (IPDPS), Anchorage, AK, USA. IEEE Computer Society, May 2011.

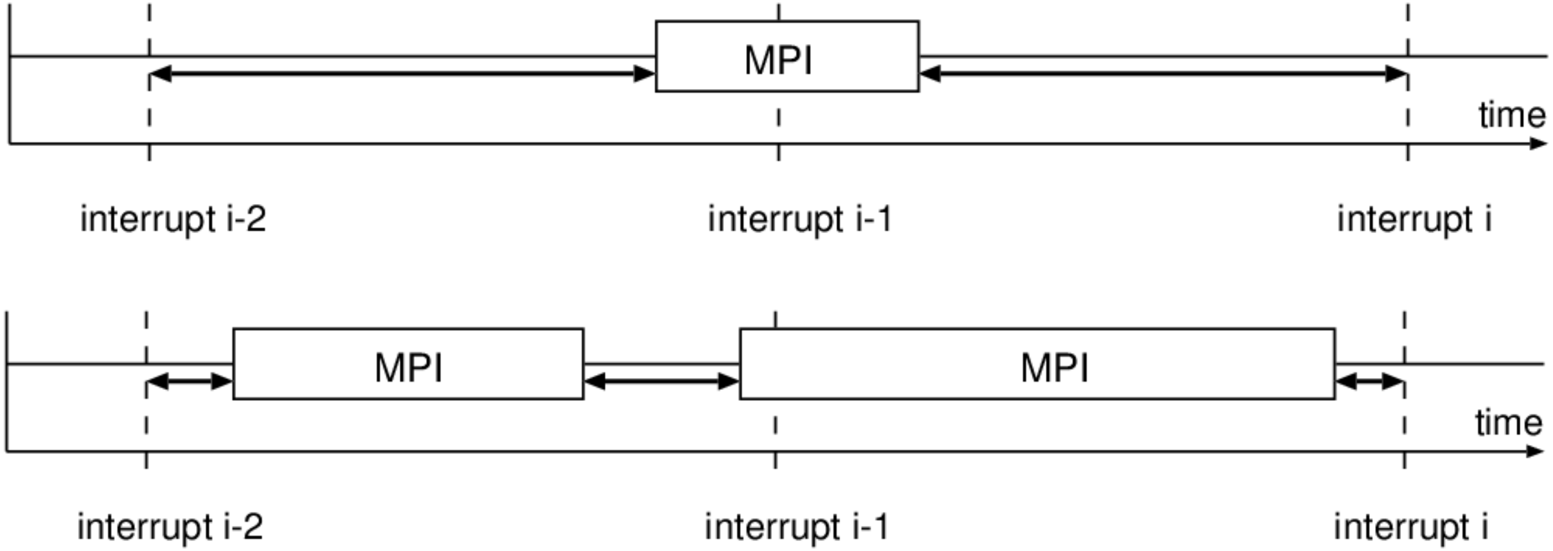
Joint work with Lawrence Livermore National Laboratory

How it works

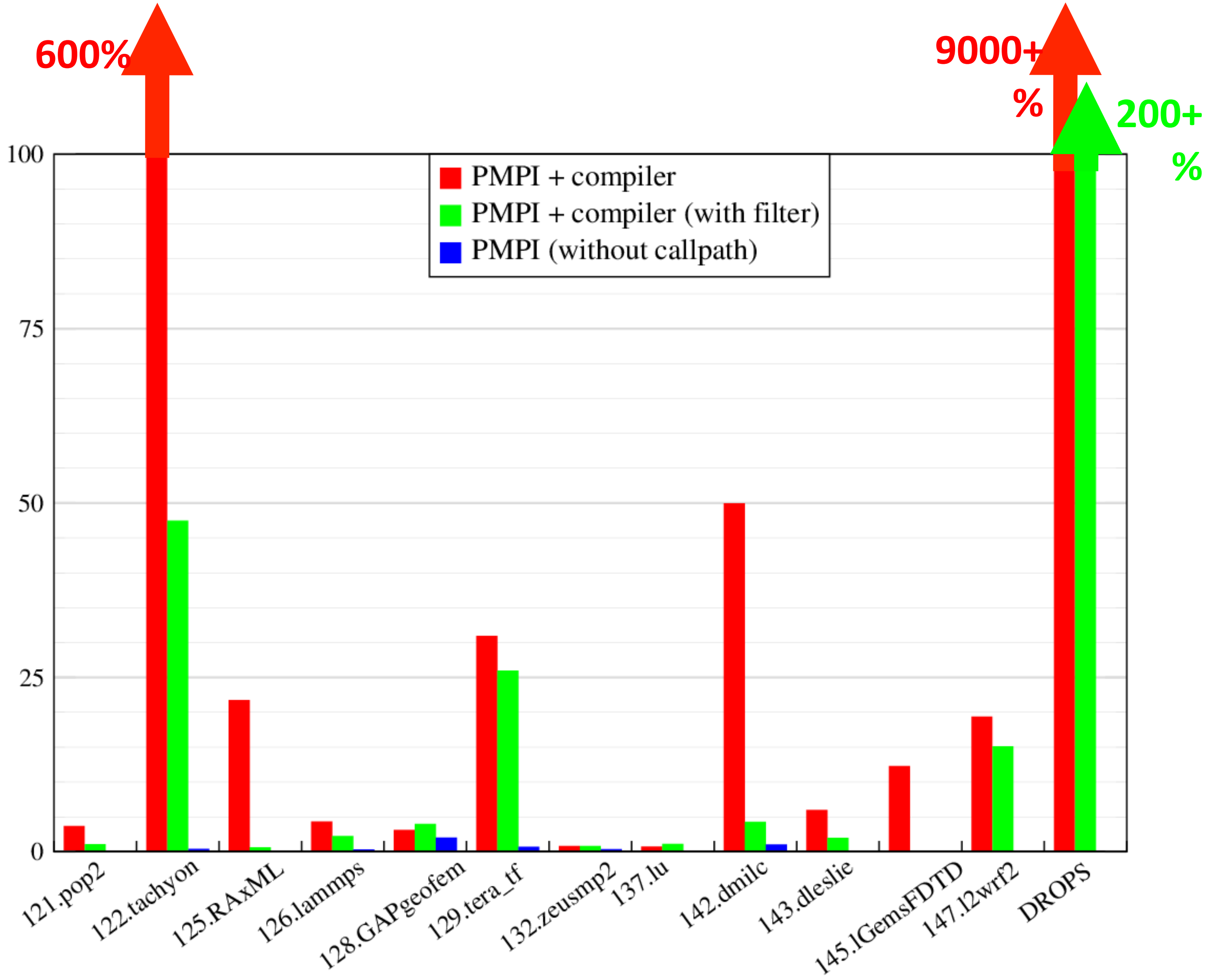
Ignore samples inside MPI calls and restart timer



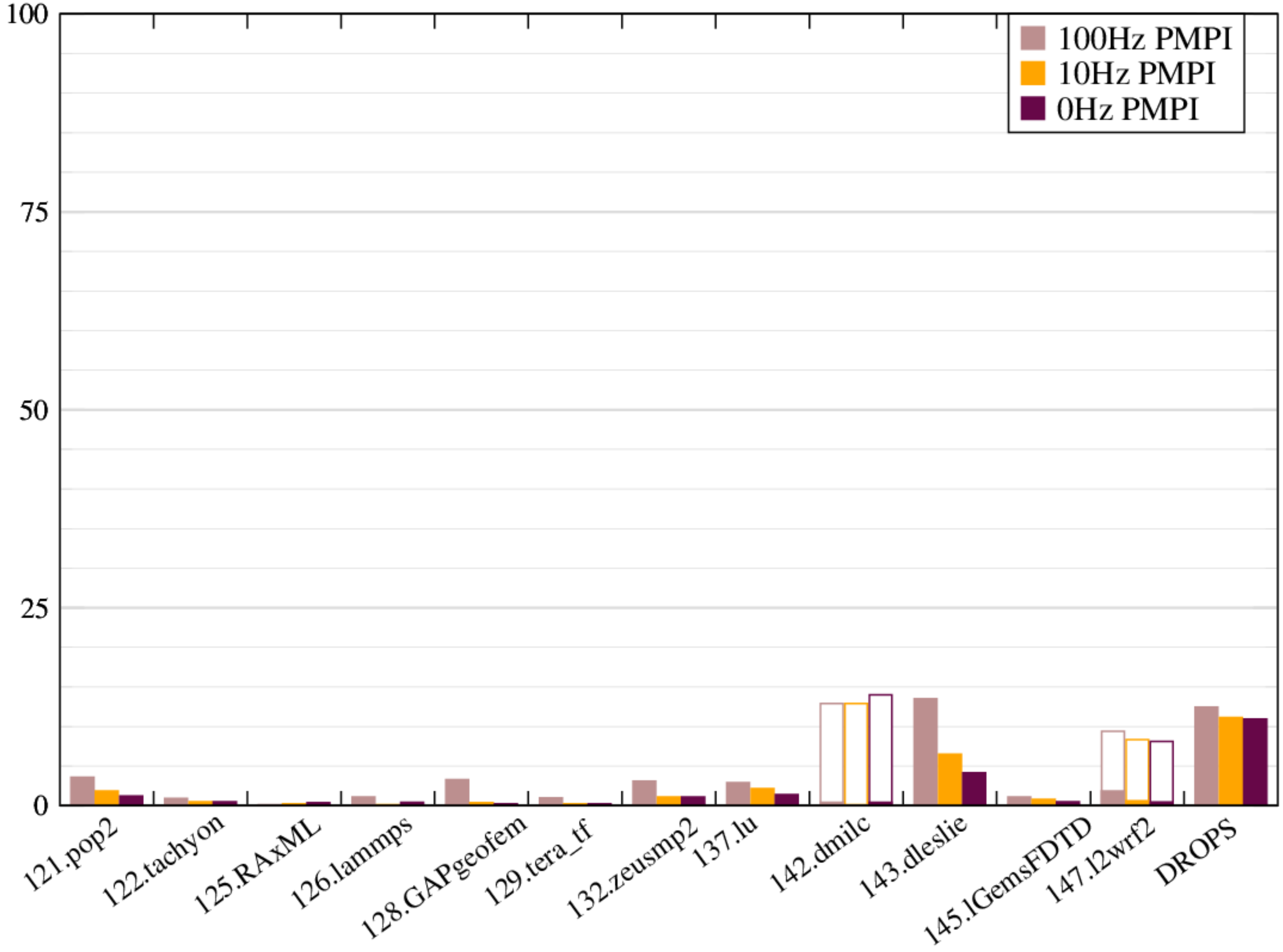
Account for extended / shortened effective interval length



Overhead of compiler instrumentation



Overhead of hybrid method



SIONlib update

- Support for OpenMP and hybrid programs (MPI/OpenMP)
- Multi-file support in serial tools and API
- New installation process with configure tool
- Fortran interface enhanced

http://www.fz-juelich.de/jsc/sionlib/

Summary

- Ensuring scalability is continuous labor-intensive effort
  - Next step: validation of new enhancements in concert
- Delay analysis offers new insight into the actual cost of load and communication imbalance
- No single cure for measurement dilation
  - However, combination of different methods often successful

Thank you!

HELMHOLTZ
ASSOCIATION

Deutsche
Forschungsgemeinschaft
DFG

Bundesministerium
für Bildung
und Forschung

DEPARTMENT OF ENERGY
UNITED STATES OF AMERICA