Mitglied der Helmholtz-Gemeinschaft

JÜLICH FORSCHUNGSZENTRUM

New developments around Scalasca

July 20th 2009

Felix Wolf

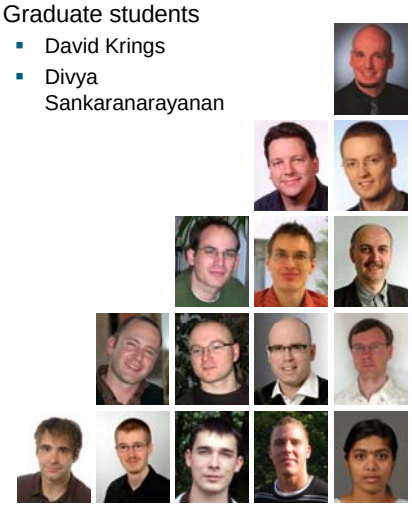
JÜLICH FORSCHUNGSZENTRUM

Outline

- Short (!) introduction to Scalasca
- Space efficient time-series call-path profiling
- Scalable parallel task-local file I/O
- Configurable source-code instrumentation

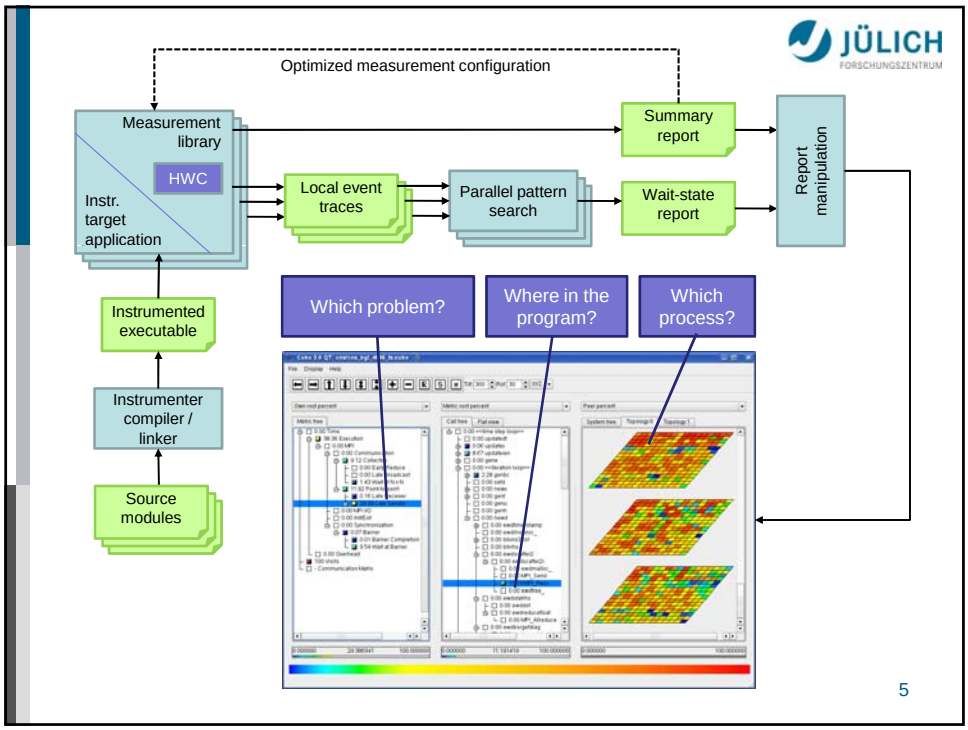
The Scalasca team

- Scientific staff
  - Dominic Eschweiler
  - Wolfgang Frings
  - Markus Geimer
  - Marc-Andre Hermanns
  - Michael Knobloch
  - Daniel Lorenz
  - Bernd Mohr
  - Christian Rössel
  - Pavel Saviankou
  - Felix Wolf
  - Brian Wylie
- Ph.D. fellows
  - Daniel Becker
  - David Böhme
  - Zoltan Szebenyi
- Graduate students
  - David Krings
  - Divya Sankaranarayanan



scalasca

- Scalable performance-analysis toolset for parallel codes
- Integrated performance analysis process
  - Performance overview on call-path level via runtime summarization
  - In-depth study of application behavior via event tracing
    - *Automatic identification of wait states*
  - Switching between both options without recompilation or relinking
- Supported programming models
  - MPI-1, basic OpenMP
  - MPI-2, other one-sided models, and more complex OpenMP features in progress
- Available under the New BSD open-source license
  - http://www.scalasca.org/



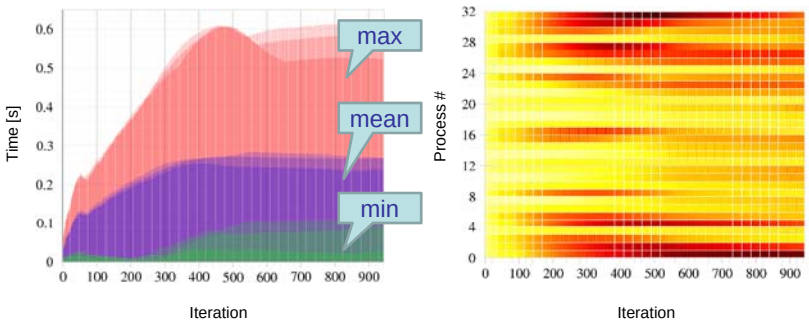
5

Releases

- 1.2 (July 2009) - mostly robustness and portability
- 1.3 (October 2009)
  - Instrumentation
    - *Tau/PDT instrumentor integration*
    - *Robustness of OPARI improved when working with multiple directories*
  - Scalability
    - *Process-local traces mapped onto few physical files*
    - *Improved communicator management*
  - Performance analysis
    - *MPI RMA statistics & trace analysis*
    - *Communication matrices*
    - *Trace-based simulator*
  - Utilities - EPILOG to slog2 trace conversion

6

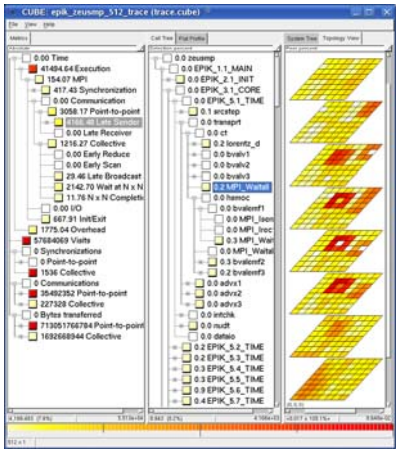
Time-dependent performance behavior



MPI point-to-point time of 129.tera_tf

Time-series call-path profiling

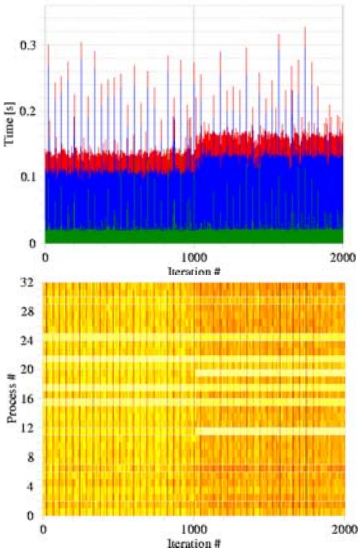
- Manual instrumentation to distinguish iterations of the main loop
- Complete call-tree recorded for each iteration
  - With multiple metrics collected for every call-path
- Huge growth in the amount of data collected
  - Reduced scalability



Incremental on-line clustering

- Exploits that many iterations are very similar
  - Summarizes similar iterations in a single iteration, their average
- On-line to save memory at run-time
- Process-local to
  - Avoid communication
  - Adjust to local temporal patterns
- The number of clusters can never exceed a predefined maximum
  - Merging of the two closest ones

MPI point-to-point time in 107.leslie3d

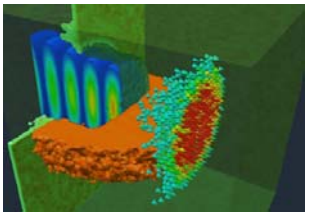


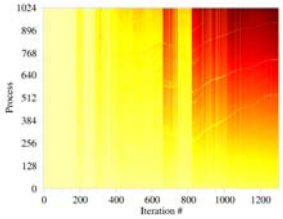
Challenges addressed

- Distance calculation overhead
  - Distance based on condensed version of iteration profile
- Phantom call paths
  - Call tree equivalence preserved
- Small changes of the baseline obscured by large-scale noise
  - Distance function takes cluster size into account

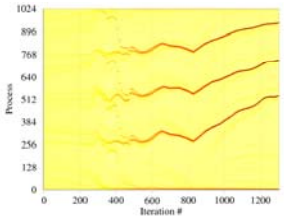
Pretty Efficient Parallel Coulomb-solver (PEPC)

- Multi-purpose parallel tree code
  - Molecular dynamics
  - Laser-plasma interactions
- Developed at JSC by Paul Gibbon
- Runs on IBM SP and Blue Gene





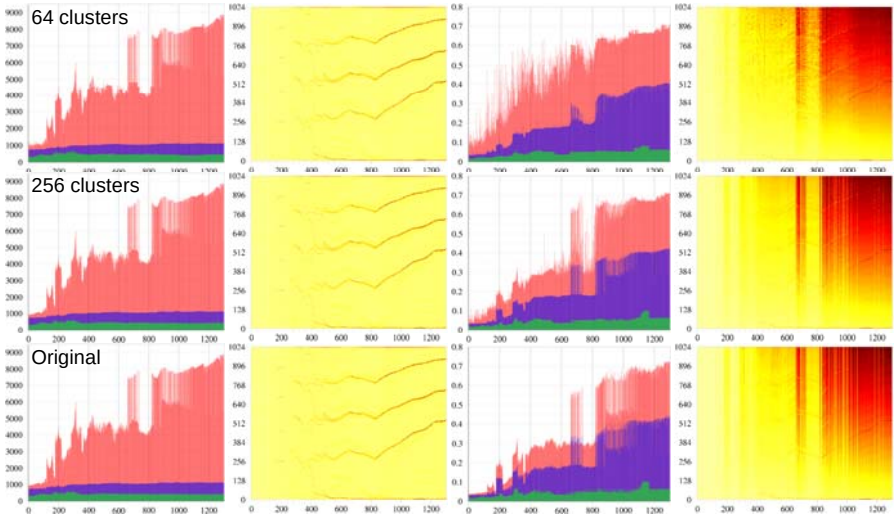
Late Sender



particles owned by a process

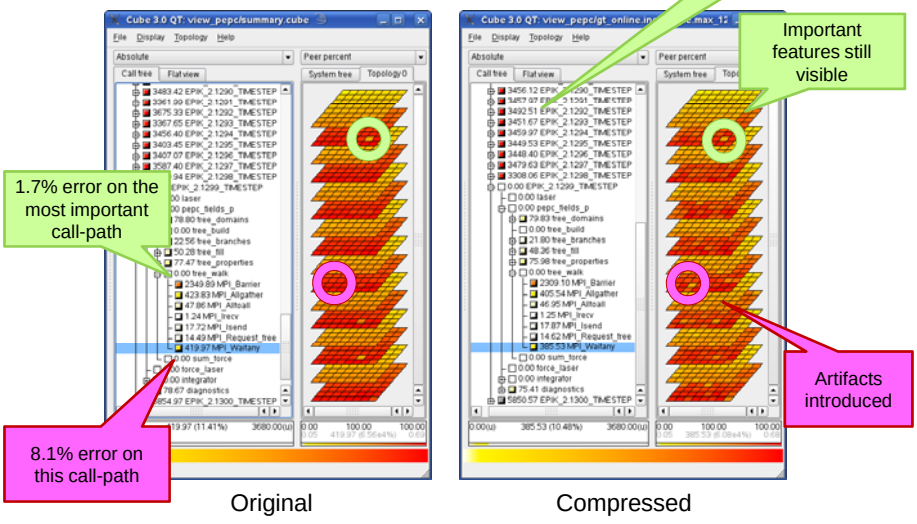
11

Examples of reconstructed graphs (PEPC)



12

Example of reconstructed call paths



Time spent in MPI by PEPC with 128 clusters

13

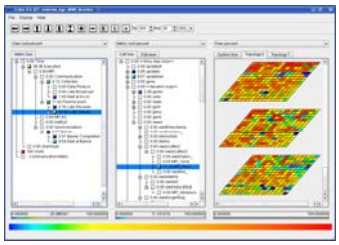
Discussion

- Lossy compression method for time-series call path profiles
  - Based on incremental on-line clustering
  - Enables analysis of time-dependent behavior
    - *Will enable full PEPC time-series call-path profile*
  - Accurate representation both at iteration and call-path level
    - *64 clusters reasonable default*
  - Low overhead
- Future work
  - Actual on-line implementation in measurement library
  - Develop strategies on how to dynamically determine the optimal number of clusters (64 not always adequate)
  - Scalability of analysis and presentations

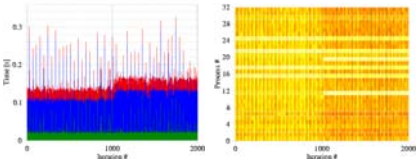
14

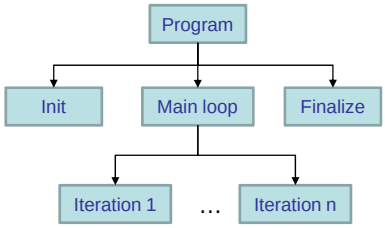
CUBE 4 roadmap

- Main goals
  - Scalability of reading and writing
  - Time dimension in addition to metric, call path, and system dimension
  - Time-series compression
  - Display of temporal patterns



CUBE 3

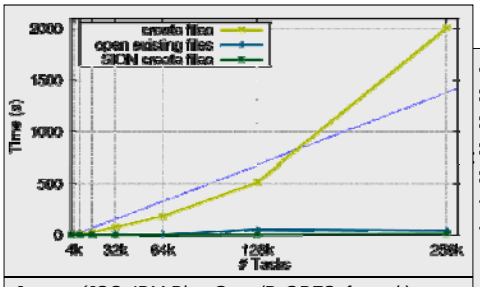




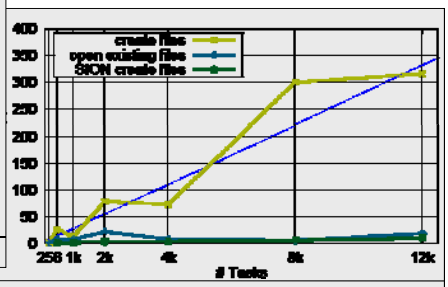
15

The file creation problem

- Metadata contention when creating thousands of files simultaneously in one directory
  - 256 K files can take > 30 min on JUGENE
- Even if the contention problem could be solved
  - Handling 256 K physical files remains a challenge



Jugene (JSC, IBM Blue Gene/P, GPFS, fs:work)

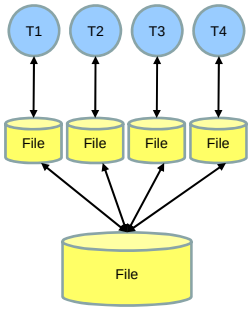


Jaguar (Oakridge, Cray XT4, Lustre, fs:scr72b)

16

SIONlib: Scalable parallel I/O for task-local data

- Use cases
  - Trace files
  - Scratch/checkpoint files
- Map many logical files onto a few physical files
  - Application-level file system
  - Optimized I/O via block alignment
- Release 1.1
  - Multiple physical files
  - Fortran interface
  - Documentation
  - Convenience functions
- http://www.fz-juelich.de/jsc/sionlib/



17

Configurable source-code instrumentor

- Motivation
  - Every tools provides its own source-code instrumentation solution
- Configurable solution based on the TAU instrumentor
  - Developed in collaboration with the TAU group (U. Oregon)
  - Robust, well-tested, extensive filtering capabilities
- Distributed as part of PDT (i.e., separate from TAU)

18

Features

- Allows specification of code to be inserted for
  - Entering/leaving a routine
  - Additional declarations
  - Inclusion of header files
  - Aborting the application
  - Initialization (i.e., at begin of “main”)
- Provides keyword substitution (to insert “instrumentor knowledge”)
- Work in progress
  - Handle Fortran line-length limit correctly
  - Complete C++ template support

19

Thank you!

scalasca
www.scalasca.org

Please download and try
Version 1.2

HELMHOLTZ ASSOCIATION

Deutsche Forschungsgemeinschaft DFG

Bundesministerium für Bildung und Forschung

20