Scalable Collation and Presentation of Call-Path Profile Data with CUBE

Markus Geimer[1], Björn Kuhlmann[1,3], Farzona Pulatova[1,2], Felix Wolf[1,3], Brian Wylie[1]

[1] Forschungszentrum Jülich GmbH
[2] University of Tennessee, Knoxville
[3] RWTH Aachen University

Forschungszentrum Jülich
in der Helmholtz-Gemeinschaft

Outline

- Introduction
- Parallel collation of input data
- Client-server architecture
- Optimized data structures and algorithms
- Conclusion

Motivation

- Advanced numerical simulations harness higher degrees of parallelism
  - Custom-built large-scale systems
  - More CPU cores instead of higher clock speeds



- Scalability is a major concern

This does not only apply to simulations, but also to tools!

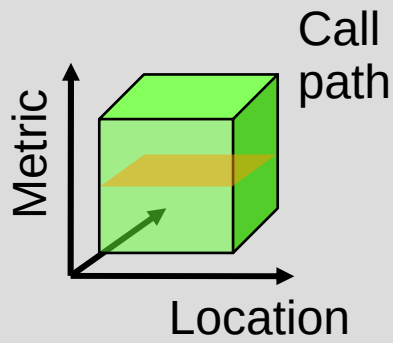
What is CUBE?

- Data model to represent runtime data from parallel programs
- XML-based file format
- C++ library to create/read/write experiments
- Generic graphical user interface

- Primarily designed to display analysis results in the SCALASCA performance tool set
- Also used in combination with TAU and MARMOT

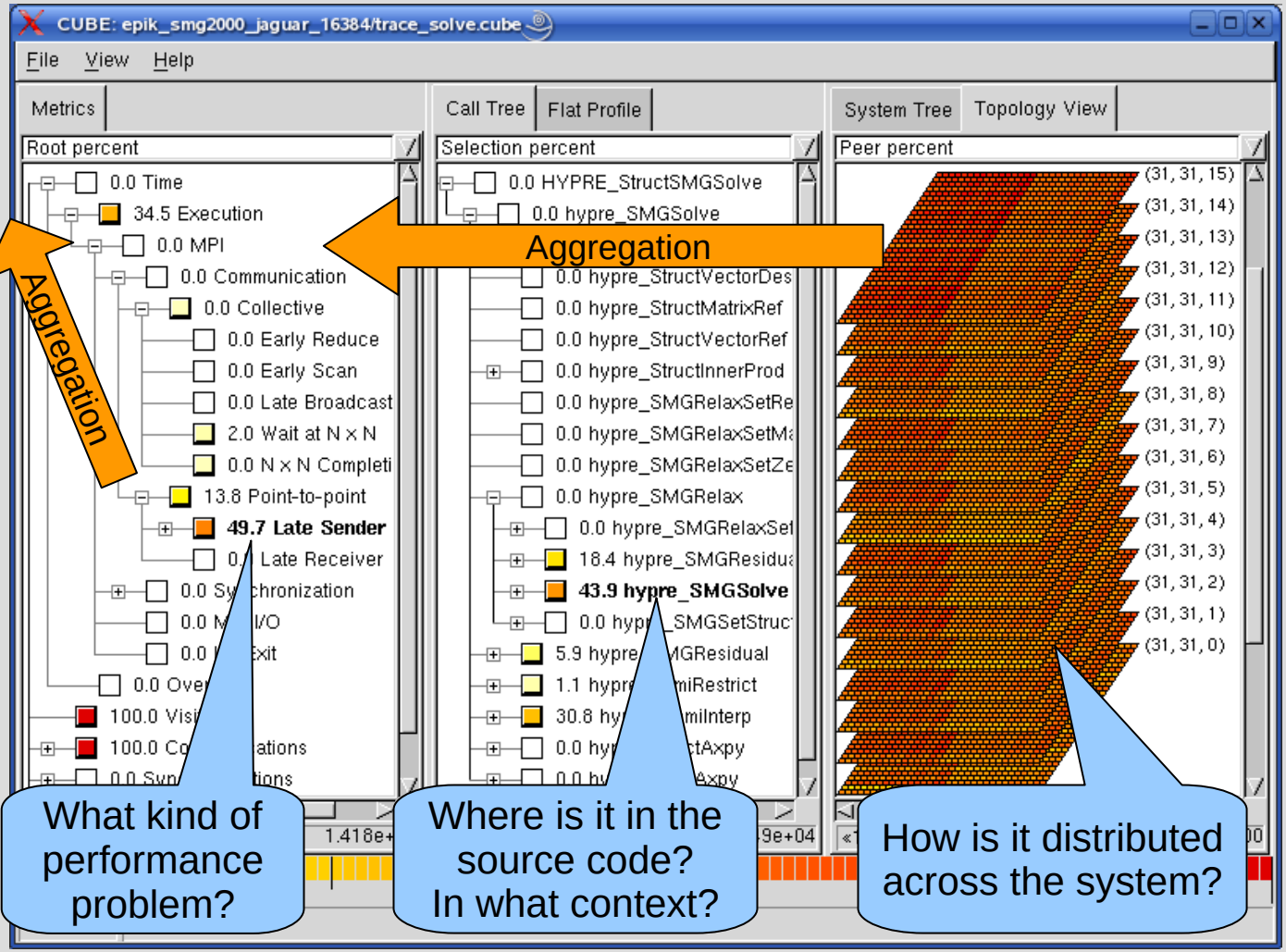
CUBE data model

- Three-dimensional representation
  - Metrics
  - Call paths
  - System locations (threads)
- Each dimension arranged in a hierarchy



- Discrete severity function:
  (metric *m*, call path *c*, thread *t*) $\rightarrow$ severity value *v*
- CUBE experiments consist of
  - Definition part defining the hierarchies
  - Severity function values

CUBE user interface

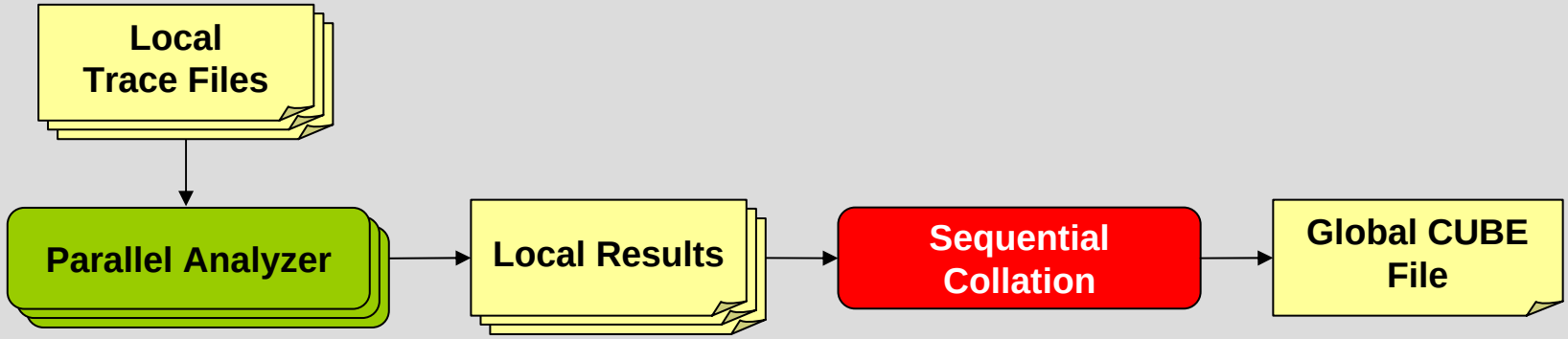


Scalability issues in CUBE

- Collation (generation) of data sets
- Copying data between file systems
- Memory usage of the display
- Interactive response times

Collation of data sets

- Data flow of previous SCALASCA trace analyzer prototype:



- Limited scalability
  - Redundant definition data in local result files
  - Linear scaling of sequential collation
  - Unnecessary file I/O

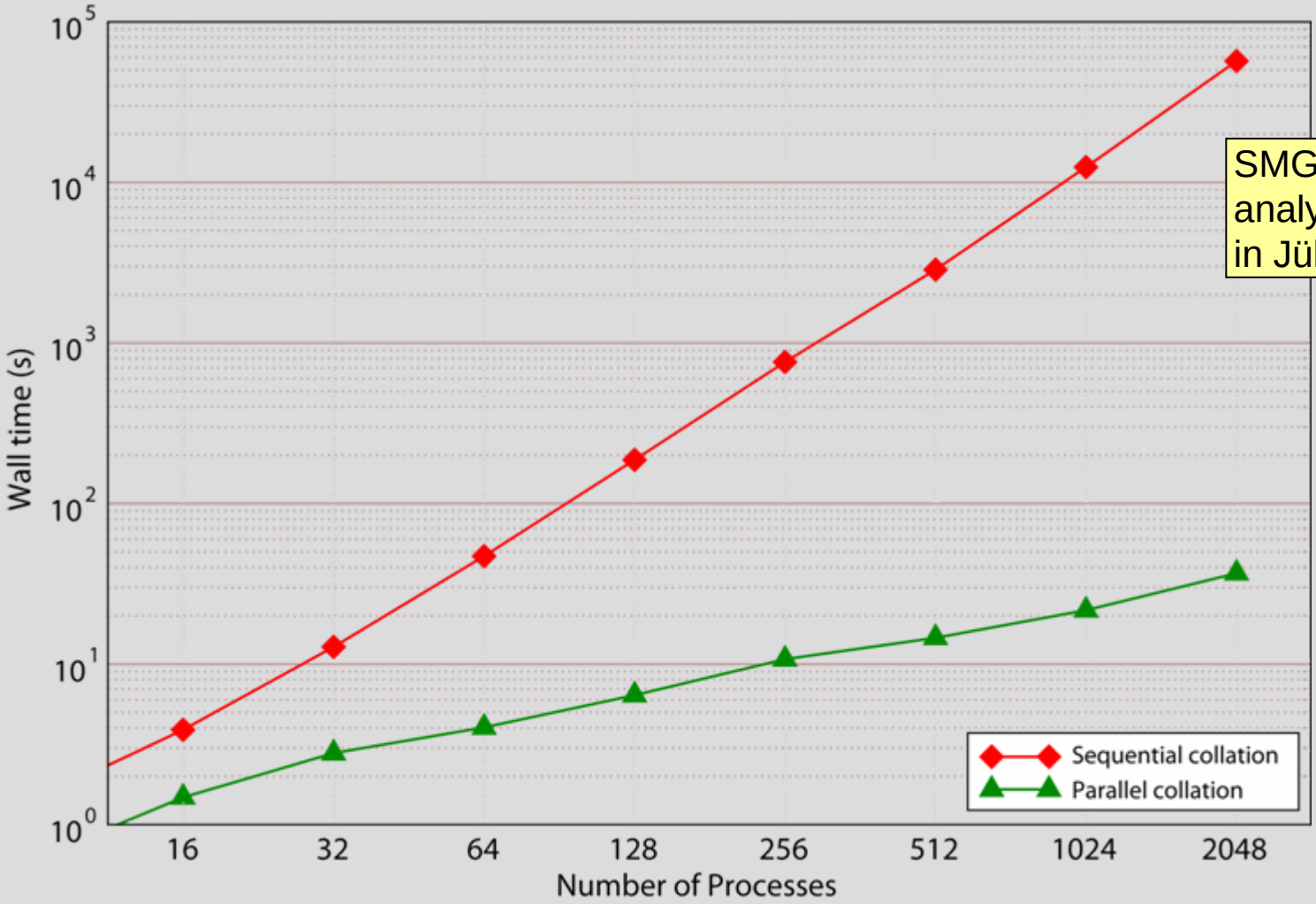
Parallel collation approach

- Online gather of local results on single master process, writing only global CUBE file
- Problem: May exceed memory capacity of master
- Algorithm:

```
For each metric m
    For each call path c
        Gather single severity from each process
        Incrementally write gathered data to CUBE file
        Global barrier
    End
End
```

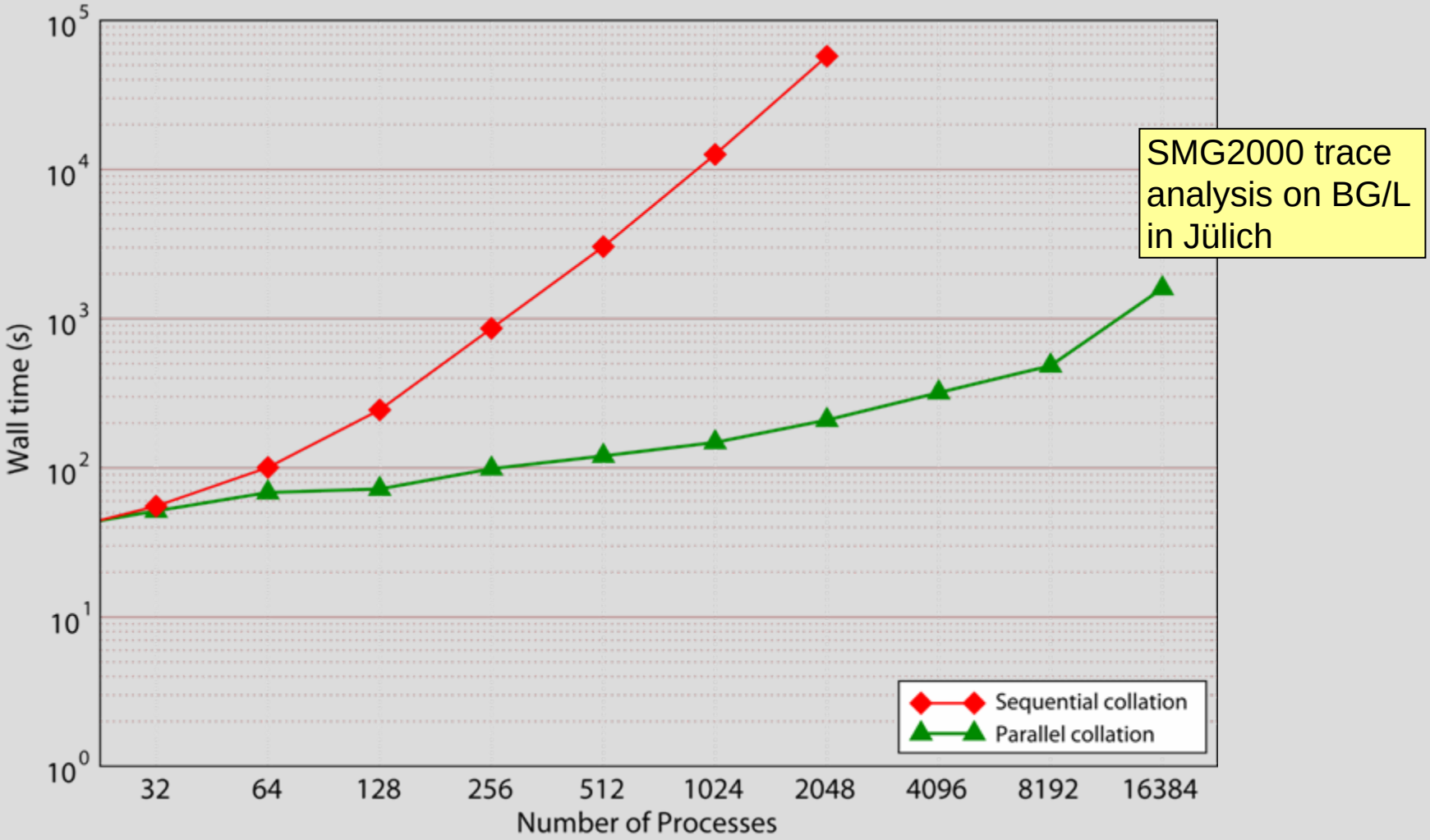
- Incremental writing implemented in C-based writer library

Comparison of collation time



SMG2000 trace analysis on BG/L in Jülich

Comparison of total analysis time



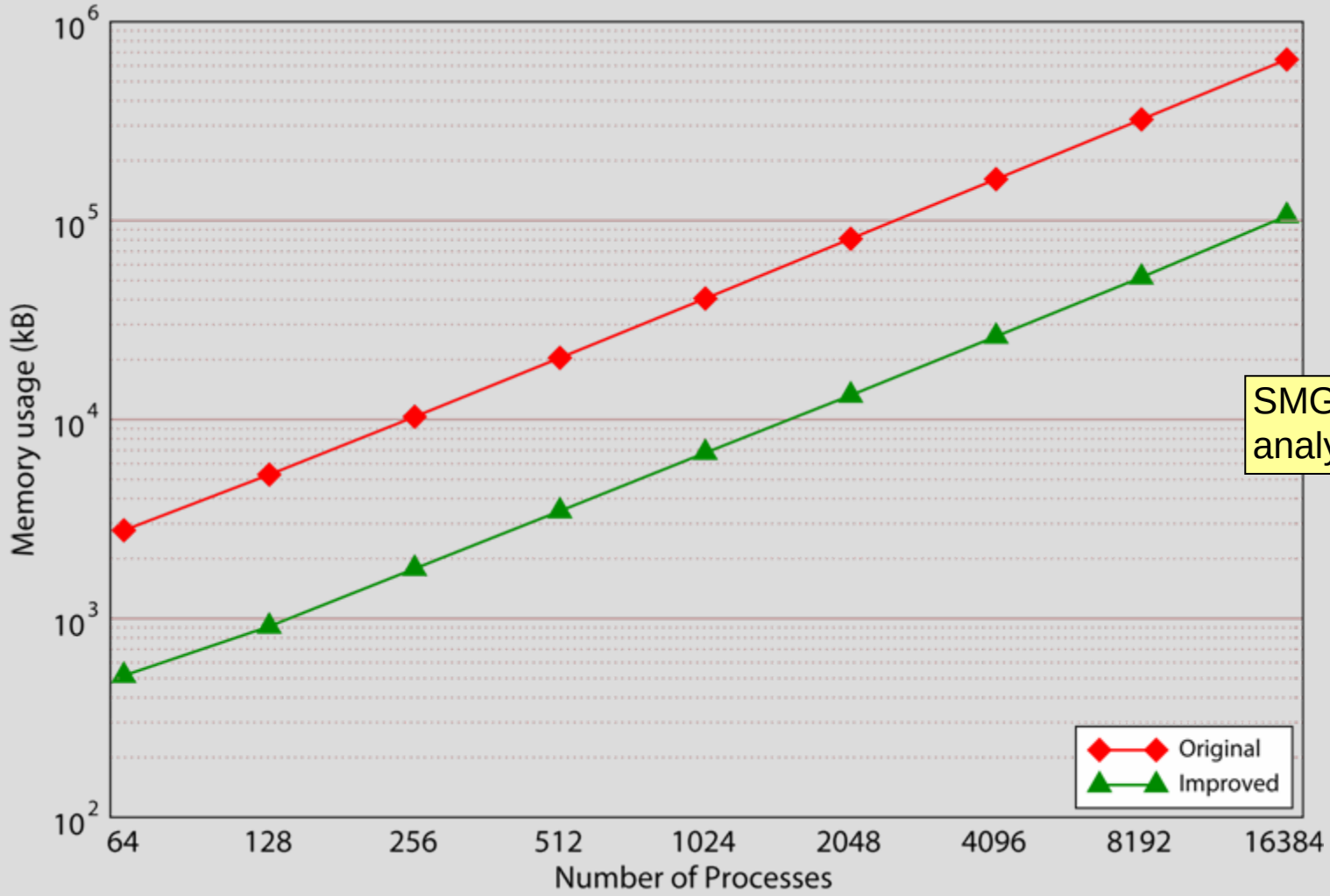
Client-server architecture

- Avoids transferring large analysis data sets from supercomputer to remote desktop
- Client
  - Running on user‘s desktop machine
  - Lightweight display component
  - Querying required data from server on demand
  - Caches static definition data
- Server
  - Running on supercomputer‘s frontend/login node
  - Takes care of data processing and aggregation
  - Can potentially be parallelized (OpenMP)

Optimized data structures

- Severity function stored as 3D matrix
  - Sparsely populated
  - Implemented using nested STL map containers
    Metric $\rightarrow$ (Call path $\rightarrow$ (Thread $\rightarrow$ Severity))
- Observation: Thread dimension is usually very dense
  - STL maps often implemented as self-balancing binary search trees
  - Replacing innermost map by an STL vector (i.e., contiguous array) saves memory overhead of tree data structure
  - Better memory locality also improves performance

Comparison of CUBE memory usage



SMG2000 trace analysis results

Improved aggregation algorithms

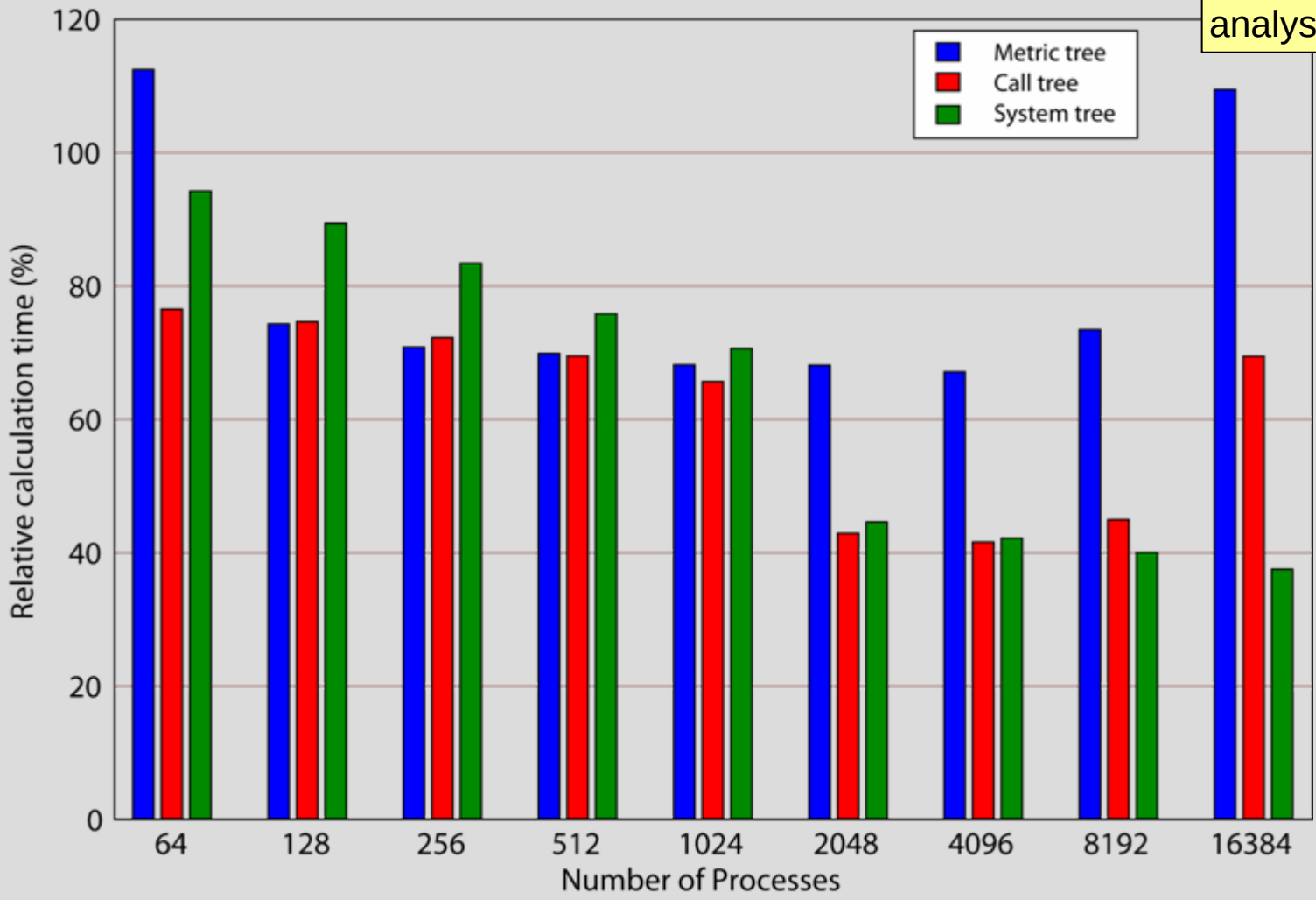
- Enumeration of metrics, call paths, and threads
- Child nodes get higher IDs than their parents
- Allows replacing recursions with iterations

- Example: Inclusive/exclusive severity values
  - Iterate from 0 to *N*-1 and calculate exclusive values
  - Iterate from *N*-1 to 0 and calculate inclusive values, reusing already calculated sums at deeper levels

- Implemented as proof-of-concept for the most important algorithms

Comparison of calculation time

SMG2000 trace analysis results



Conclusion

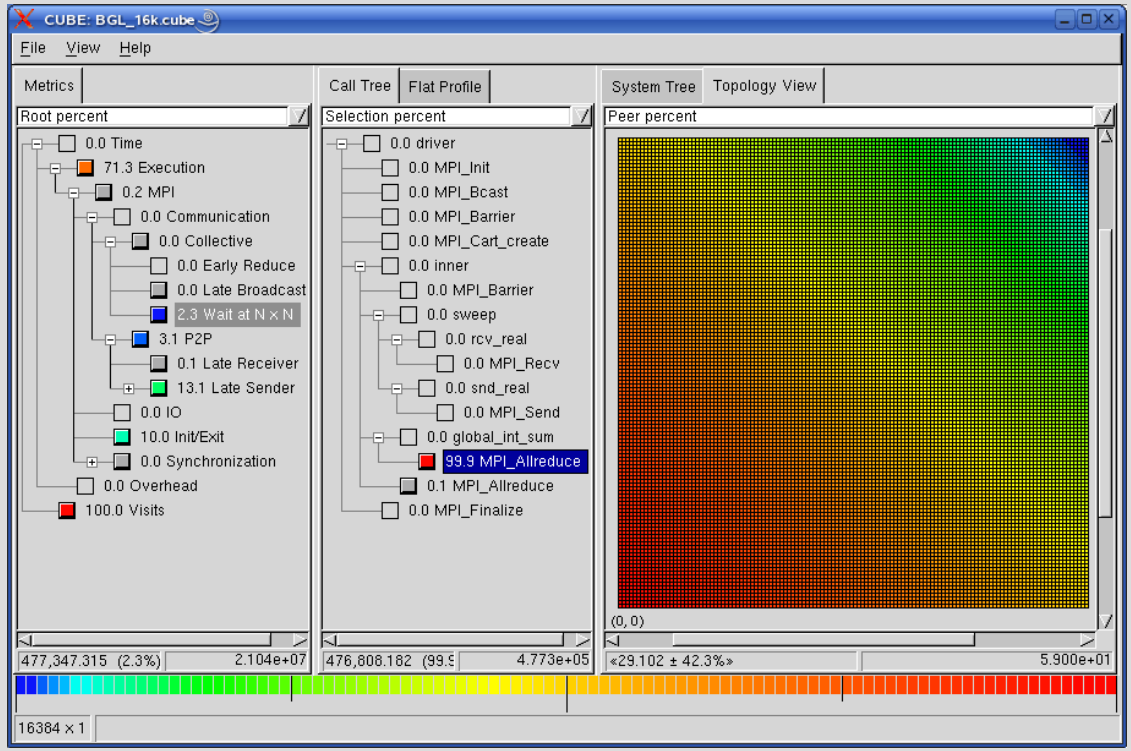
- Various scalability improvements
  - Parallel collation of experiment results
  - Client-server architecture
  - Improved data structures & algorithms
- Allows analysis of experiments at substantially larger scales

Thank you!

For more information and downloads, please visit our project home page:

http://www.scalasca.org

Try out our new release SCALASCA 0.9!



SWEEP3D virtual topology, Wait at NxN, 16K CPUs