Scalable Parallel Trace-Based Performance Analysis

Markus Geimer Felix Wolf Brian J. N. Wylie Bernd Mohr

{m.geimer, f.wolf, b.wylie, b.mohr}@fz-juelich.de

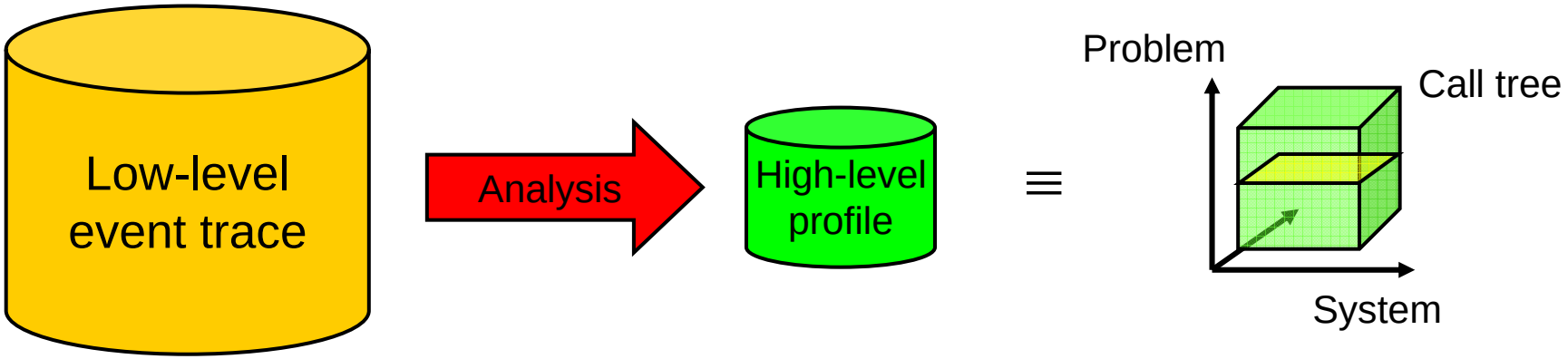
Forschungszentrum Jülich
Germany

Outline

- Introduction
- Parallel trace-based performance analysis
- Early results
- Conclusion
- Future work

Automatic Off-Line Trace Analysis

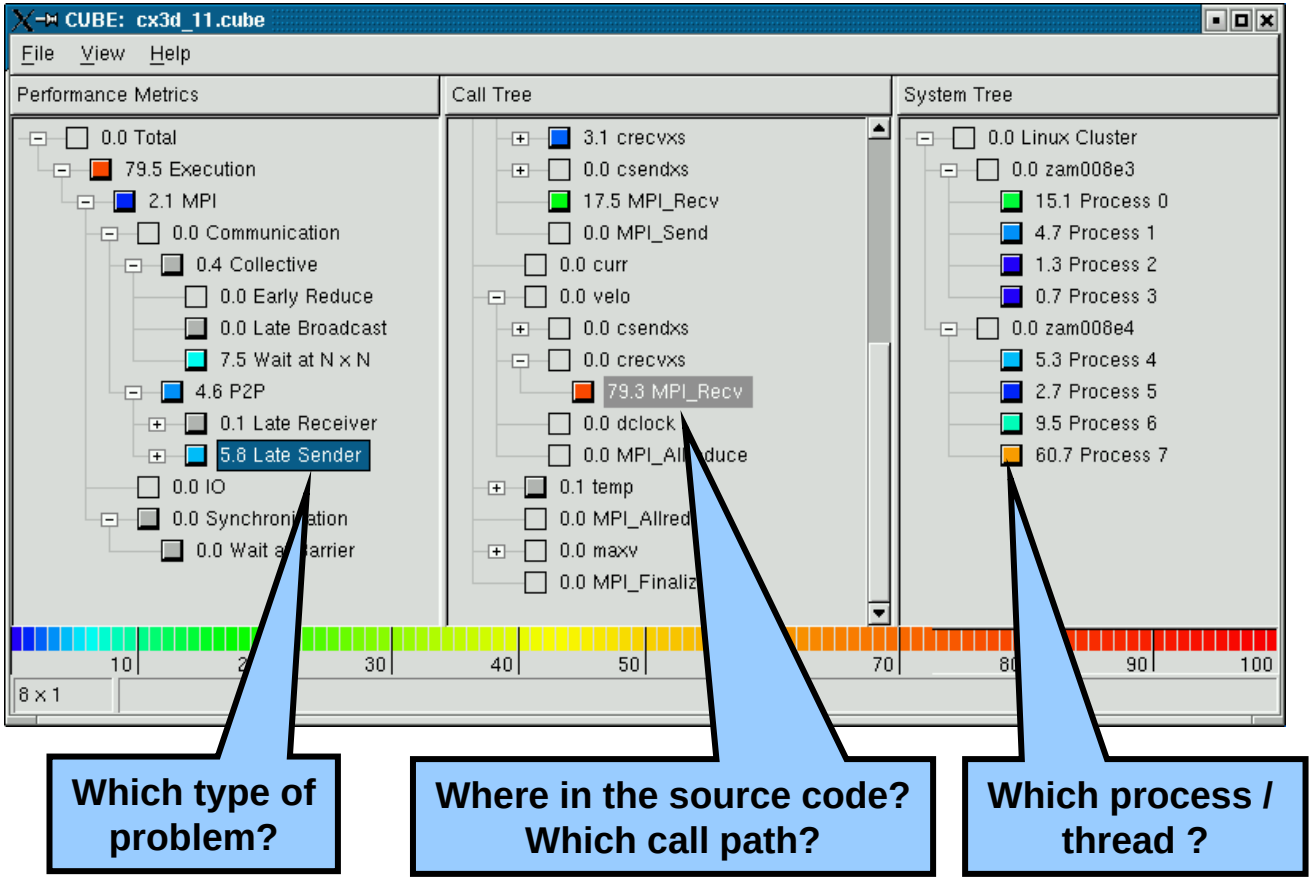
- Idea:
  - Automatic search for *patterns* of inefficient behavior
  - Quantification of significance
- Example: EXPERT analyzer from the KOJAK toolset



➔ Data distillation

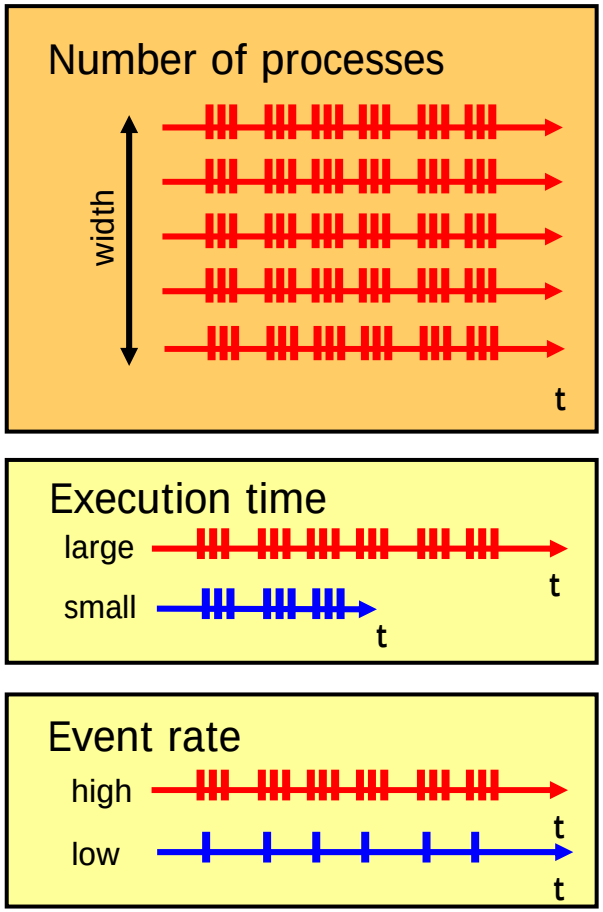
➔ Guaranteed to cover the entire trace

Analysis Report

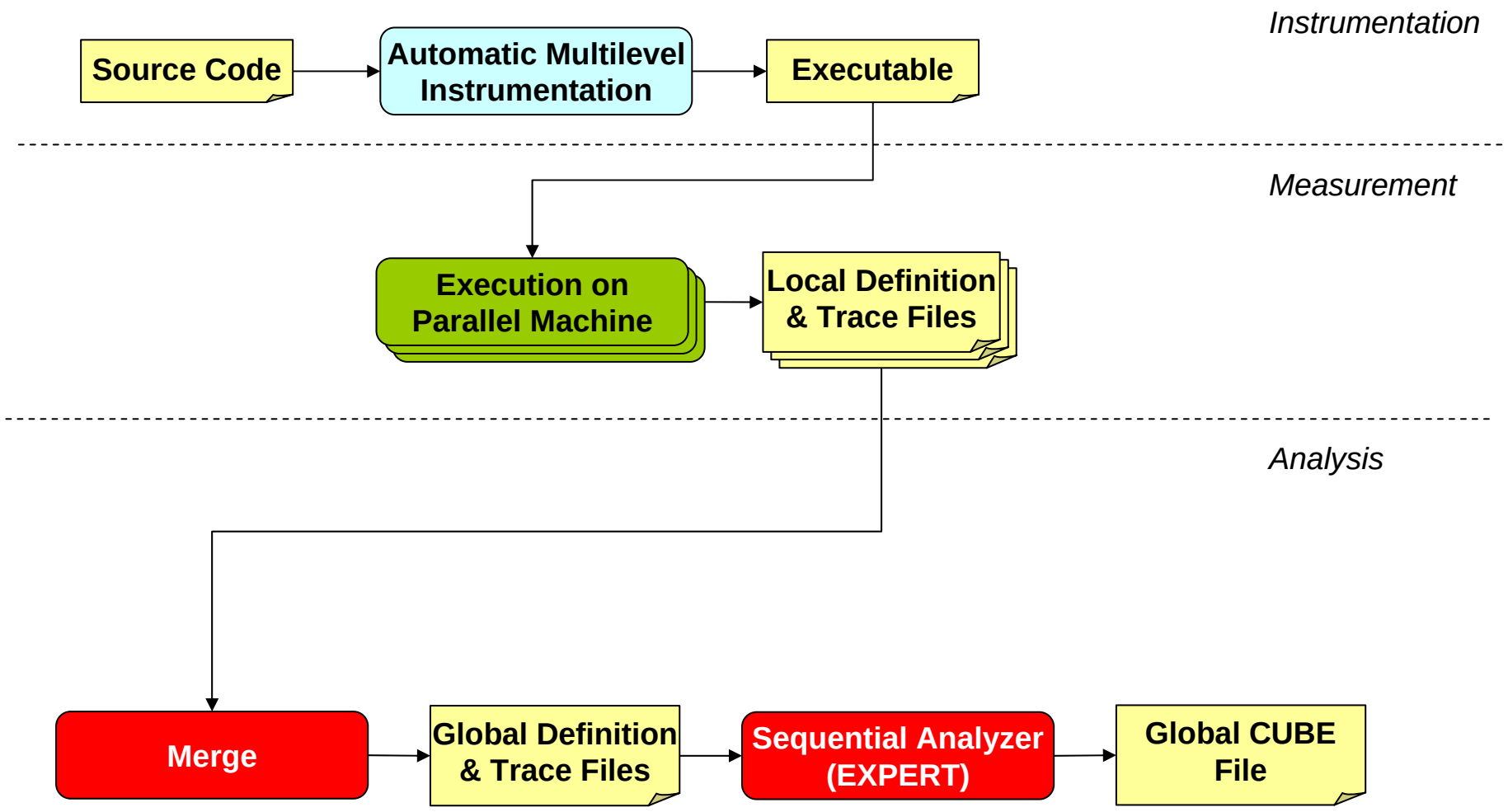


Scalability Problems

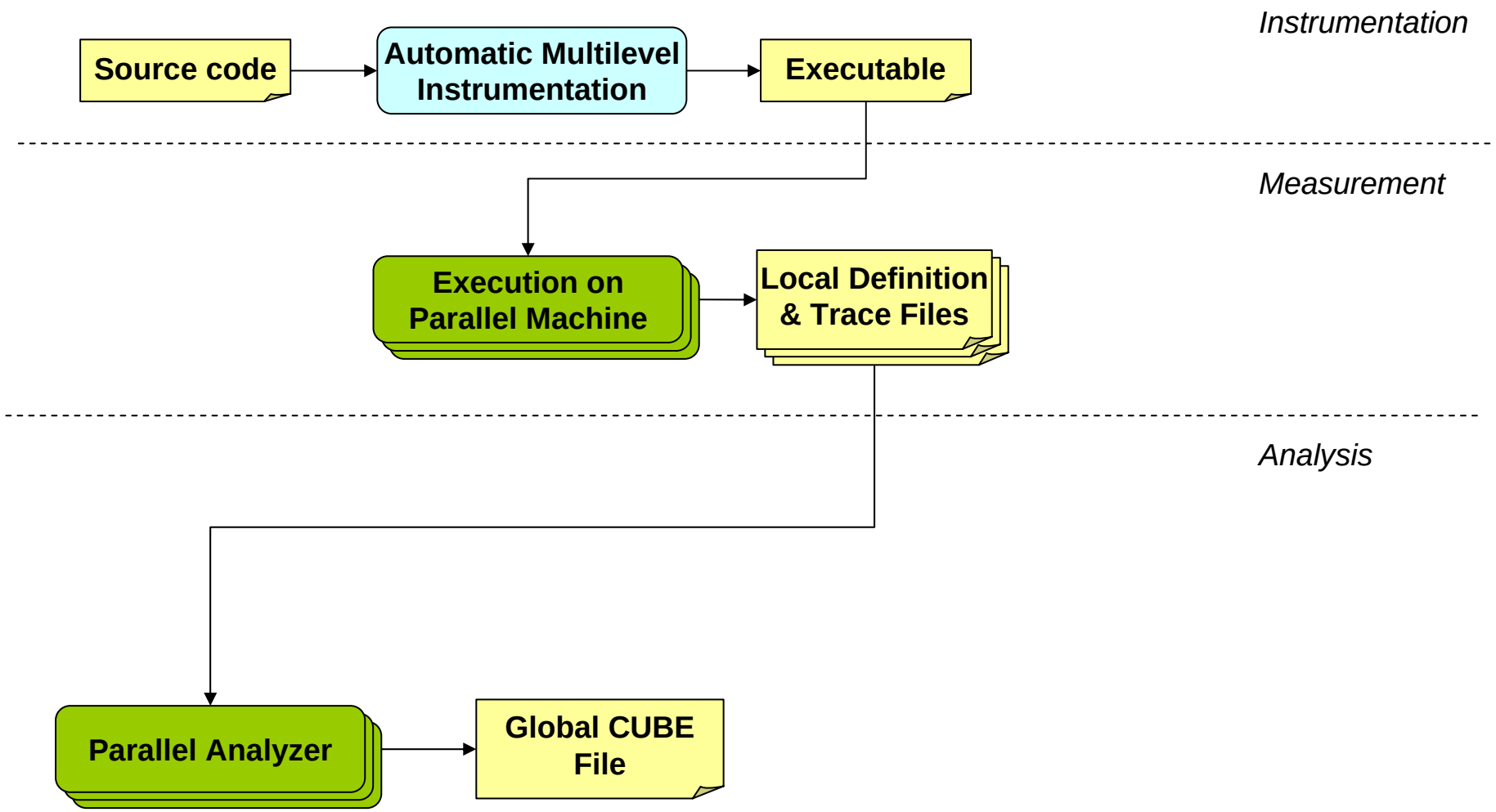
- Serially analyzing a single and potentially large global trace file does not scale to 1000s of processors
- Even if locality is exploited, main memory might be insufficient to store current working set
- Amount of trace data might not fit into single file



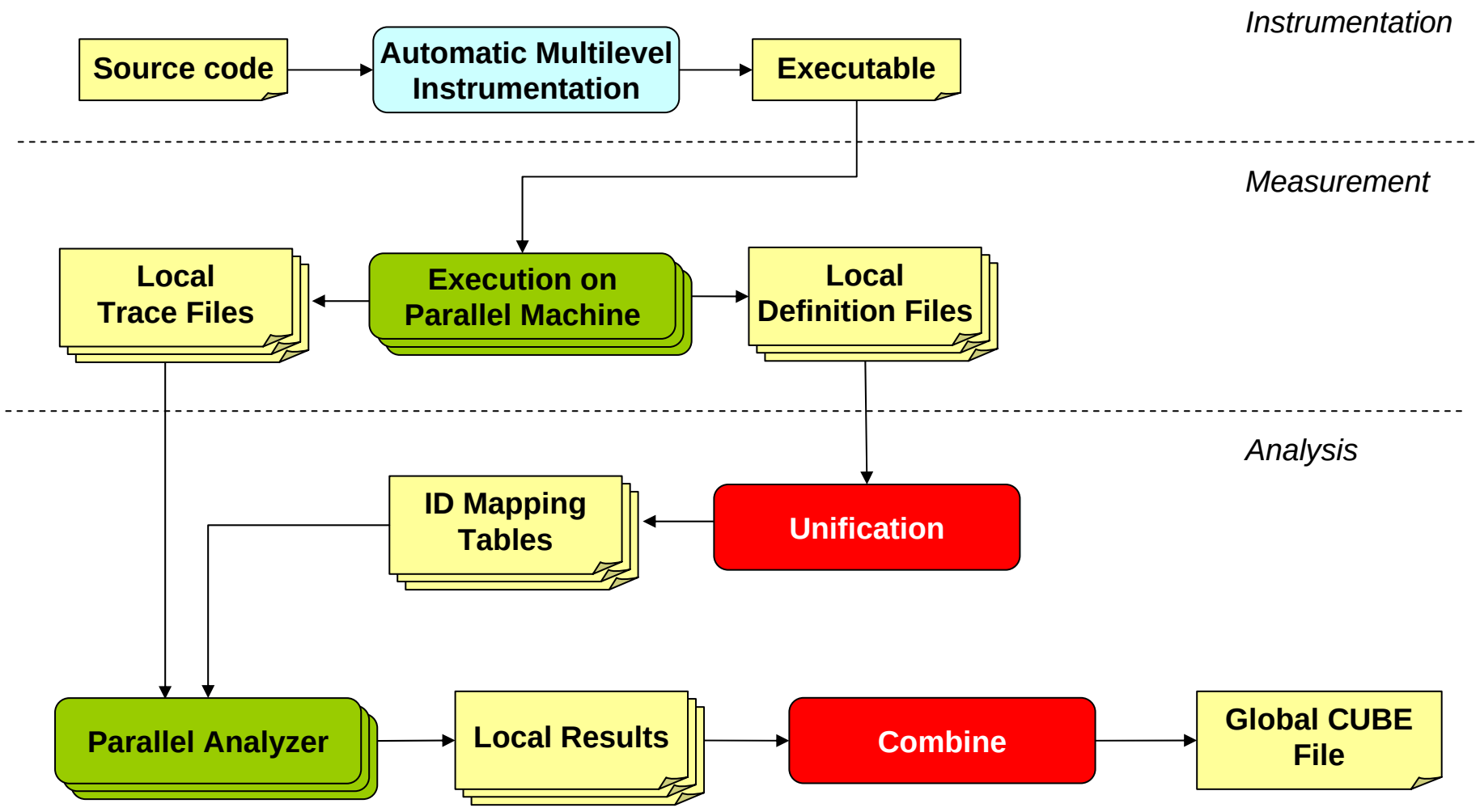
Sequential Analysis Process



Parallel Analysis Process



Prototype Implementation

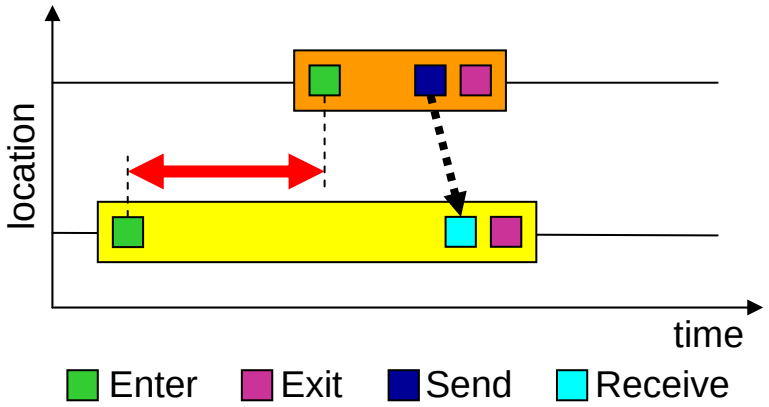


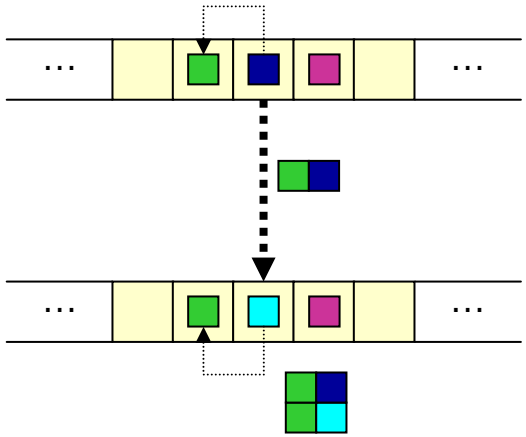
Parallel Pattern Analysis

- Analyze separate local trace files in parallel
  - Exploits distributed memory & processing capabilities
  - Often allows keeping whole trace in main memory

- *Parallel Replay* of target application‘s communication behavior
  - Analyze communication with an operation of same type
  - Parallel traversal of event streams
  - Exchange of data at synchronization points of target application

Example: Late Sender





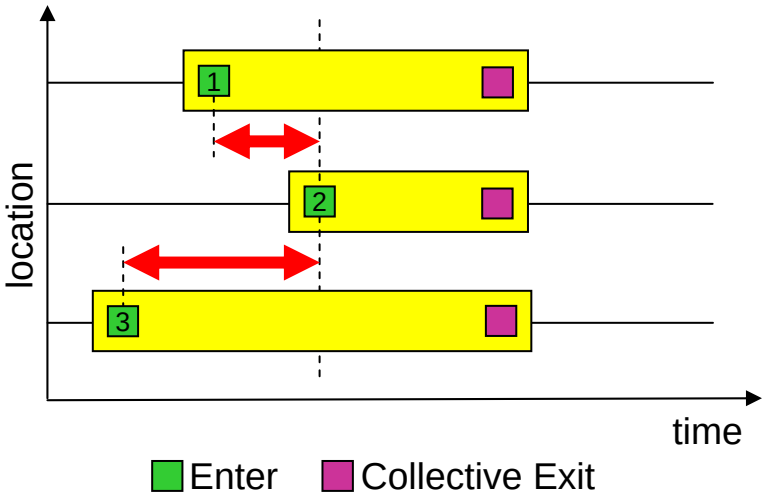
Sender:

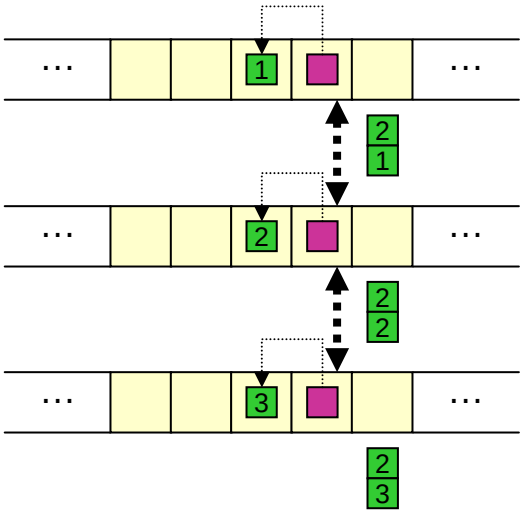
- Triggered by send event
- Determine enter event
- Send both events to receiver

Receiver:

- Triggered by receive event
- Determine enter event
- Receive remote events
- Detect *Late Sender* situation
- Calculate & store waiting time

Example: Wait at N x N





- Waiting time due to inherent synchronization in N-to-N operations (e.g., MPI_Allreduce)
- Algorithm:
  - Triggered by collective exit event
  - Determine enter events
  - Determine & distribute latest enter event (max-reduction)
  - Calculate & store waiting time

Experimental Evaluation

- PEPC-B
  - Parallel tree code for computing long-range forces in N-body problems
  - Fixed overall problem size
  - Strong scaling behavior

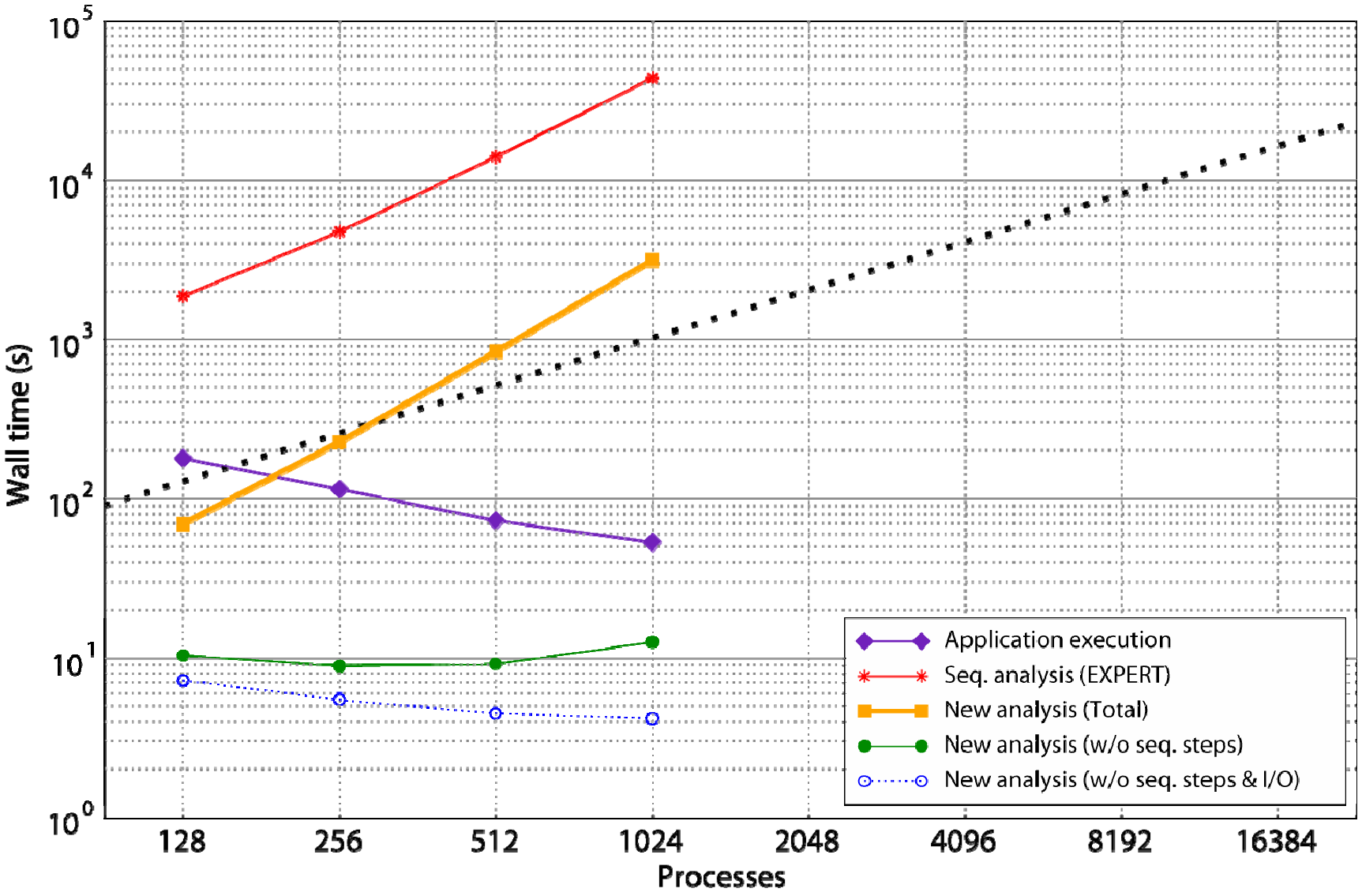
- ASC SMG2000 benchmark
  - Semi-coarsening multigrid solver
  - Fixed 64×64×32 problem size per process, 5 solver iterations
  - Weak scaling behavior

Test Platform

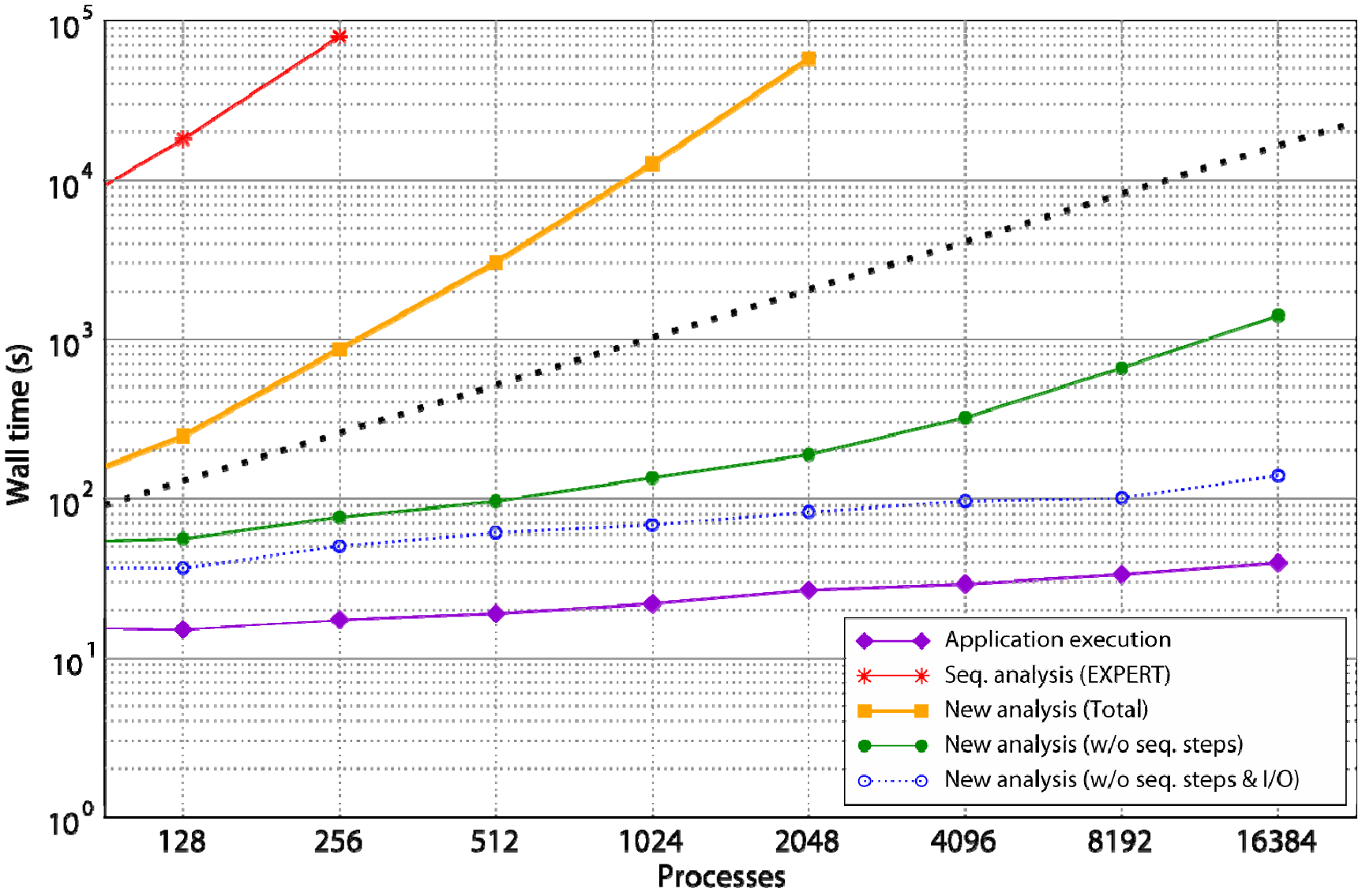
Jülicher BlueGene/L (JUBL)

- 8 Racks with 8192 dual-core nodes
- 288 I/O nodes
- p720 service and login nodes (8×1.6 GHz Power5 CPUs each)

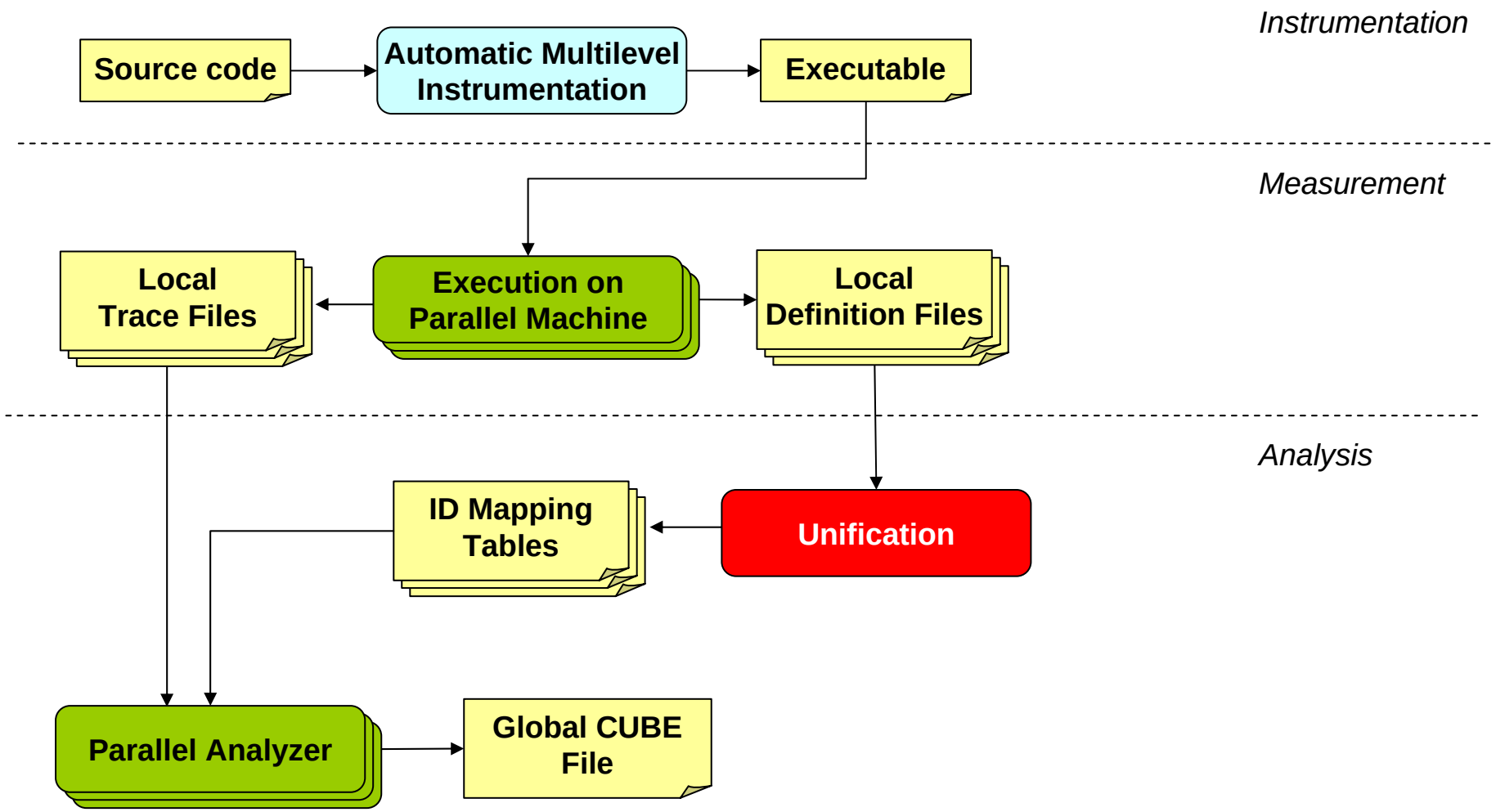
Results: PEPC-B



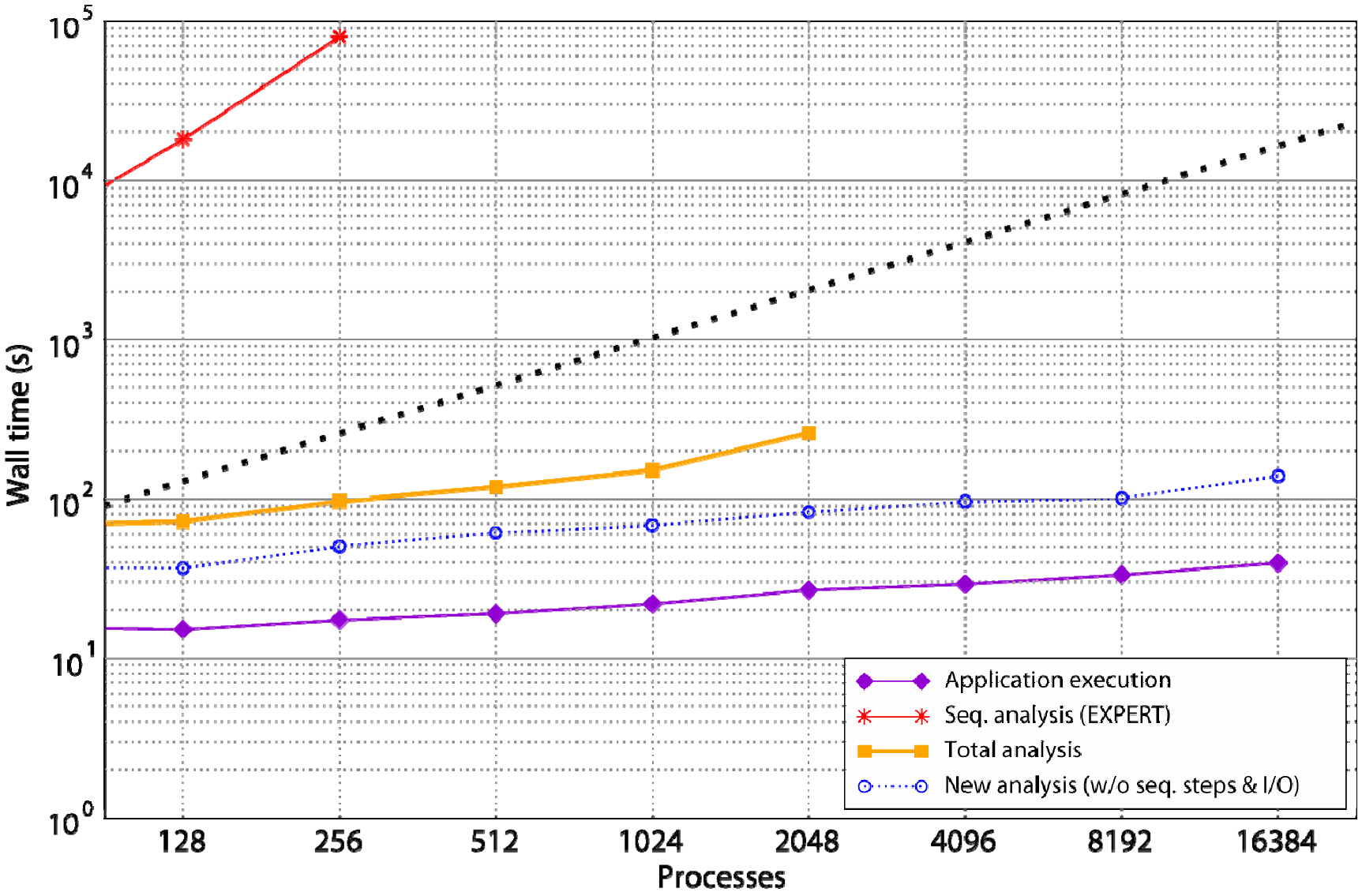
Results: SMG2000



Current Prototype



Recent Results: SMG2000



Conclusion

- Scalability can be addressed by parallelization
  - Process local trace files in parallel
  - Replay target applications communication behavior
- Promising results with prototype implementations
  - Core analysis scales up to 16,384 processes
  - Enables analyzing traces of previously impractical size
  - Example: SMG2000 on 16,364 CPUs
    - 230 GBytes trace data
    - >40,000 million events

Future Work

- Integrate & parallelize remaining sequential parts
- Address long traces
  - Selective tracing
  - Sophisticated data structures (e.g. cCCGs)
- Extend to other programming paradigms
  - OpenMP
  - MPI-2

Thank you!

For more information, visit our project home page:

http://www.scalasca.org