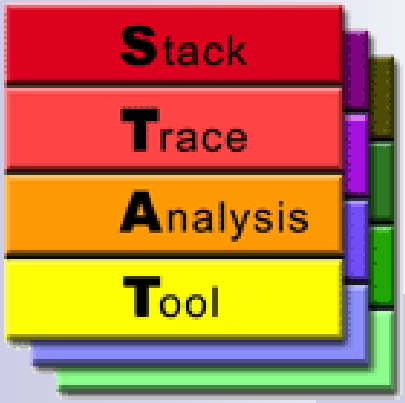


Scalable Emulation for Tool Performance Tuning

Gregory L. Lee, Dong H. Ahn,
Bronis R. de Supinski, Martin Schulz
Lawrence Livermore National Laboratory

Dorian C. Arnold, Barton P. Miller
University of Wisconsin

Parallel Computing 2007
Forschungszentrum Jülich and RWTH
Aachen University, Germany
September 4-7 2007

This work was performed under the auspices of the U.S. Department of Energy by University of California, Lawrence Livermore National Laboratory under Contract W-7405-Eng-48. UCRL-PRES-234131

Overview

- The Stack Trace Analysis Tool (STAT) is a scalable, lightweight debugging tool
- STAT is effective at debugging large scale applications
- Validating STAT's scalability is difficult

Large Scale Tool Validation is a Challenge

- Limited machine resources
  - Access to large scales
  - Machine time
  - Low priority for iterative tool development
- Goal: identify STAT scalability issues using minimal resources

Tool Scalability is Lagging Behind System Scaling

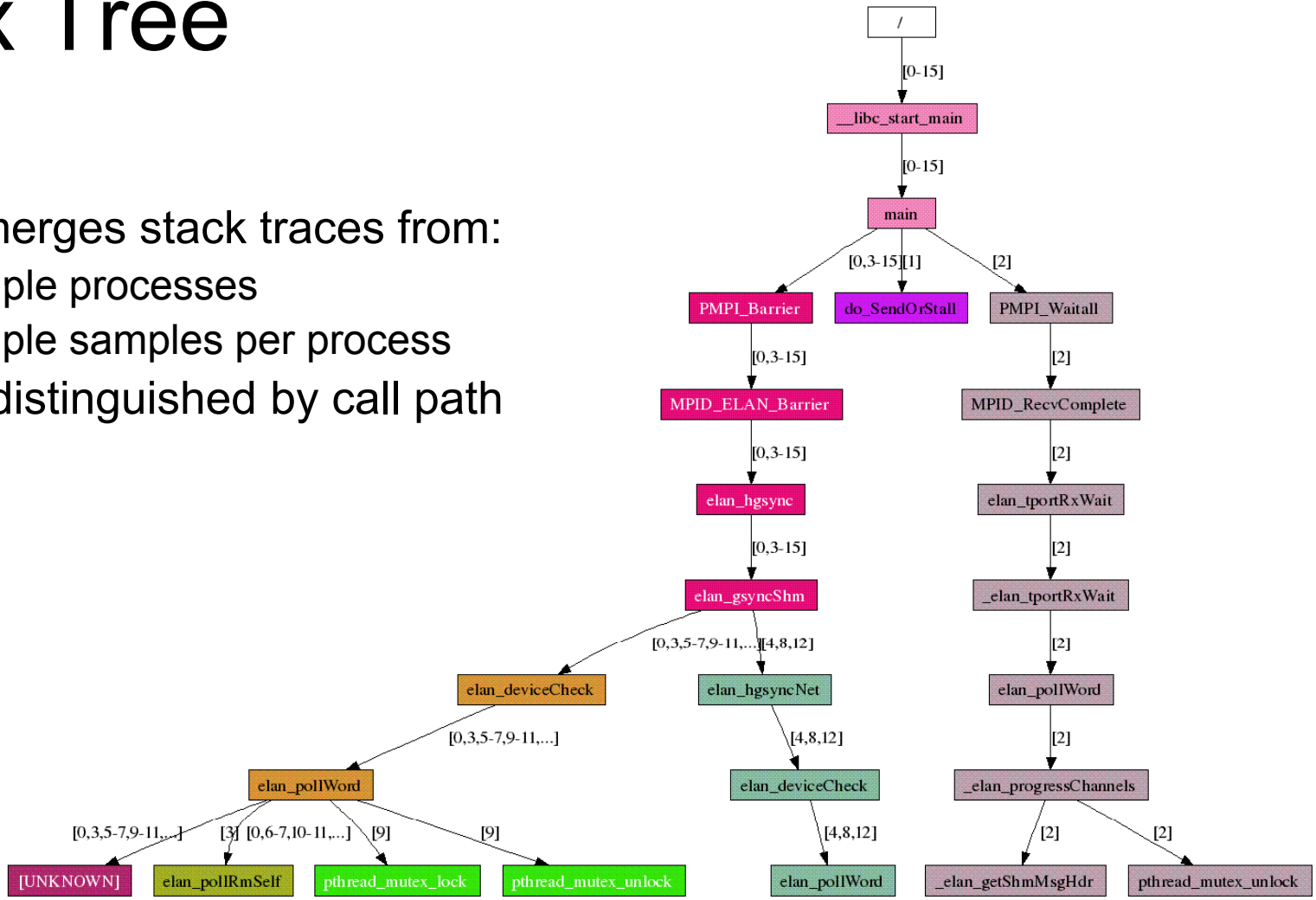
- System scales ever increasing
  - 131,072 processors for BlueGene/L at LLNL
  - O(1M) on the horizon
- Tool scalability is lagging
  - TotalView Debugger operations on BG/L at 4,096 tasks:
    - single step (~15-20 secs.)
    - breakpoint insertion (~30 secs.)
    - stack trace sampling (~120 secs)
  - 4,096 is only 3.125% of BG/L

Why are tools lagging?

- Large volumes of data
- Tool front-end bottleneck
- Vendor licensing costs and limitations
- Large scale resource limitations
- Solution: scalable, lightweight tools
  - Reduce exploration space to a small subset of tasks
  - Full-featured tool for in depth analysis

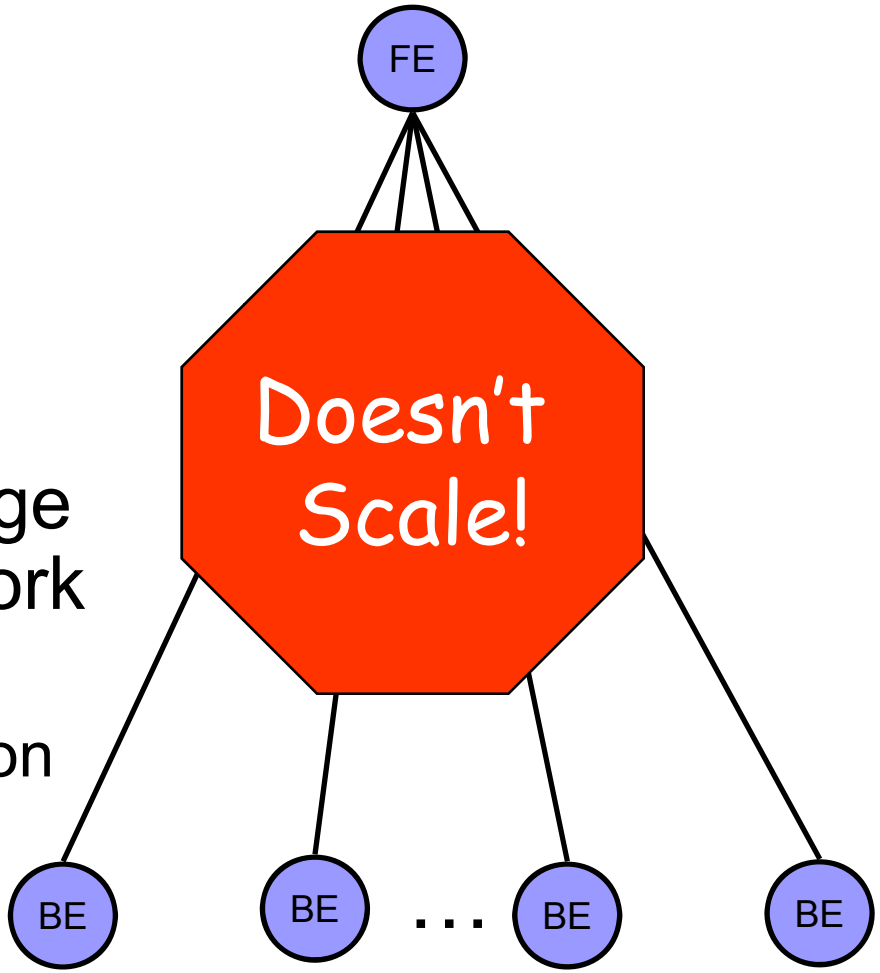
3D-Trace/Space/Time Call Graph Prefix Tree

- STAT merges stack traces from:
  - Multiple processes
  - Multiple samples per process
- Nodes distinguished by call path

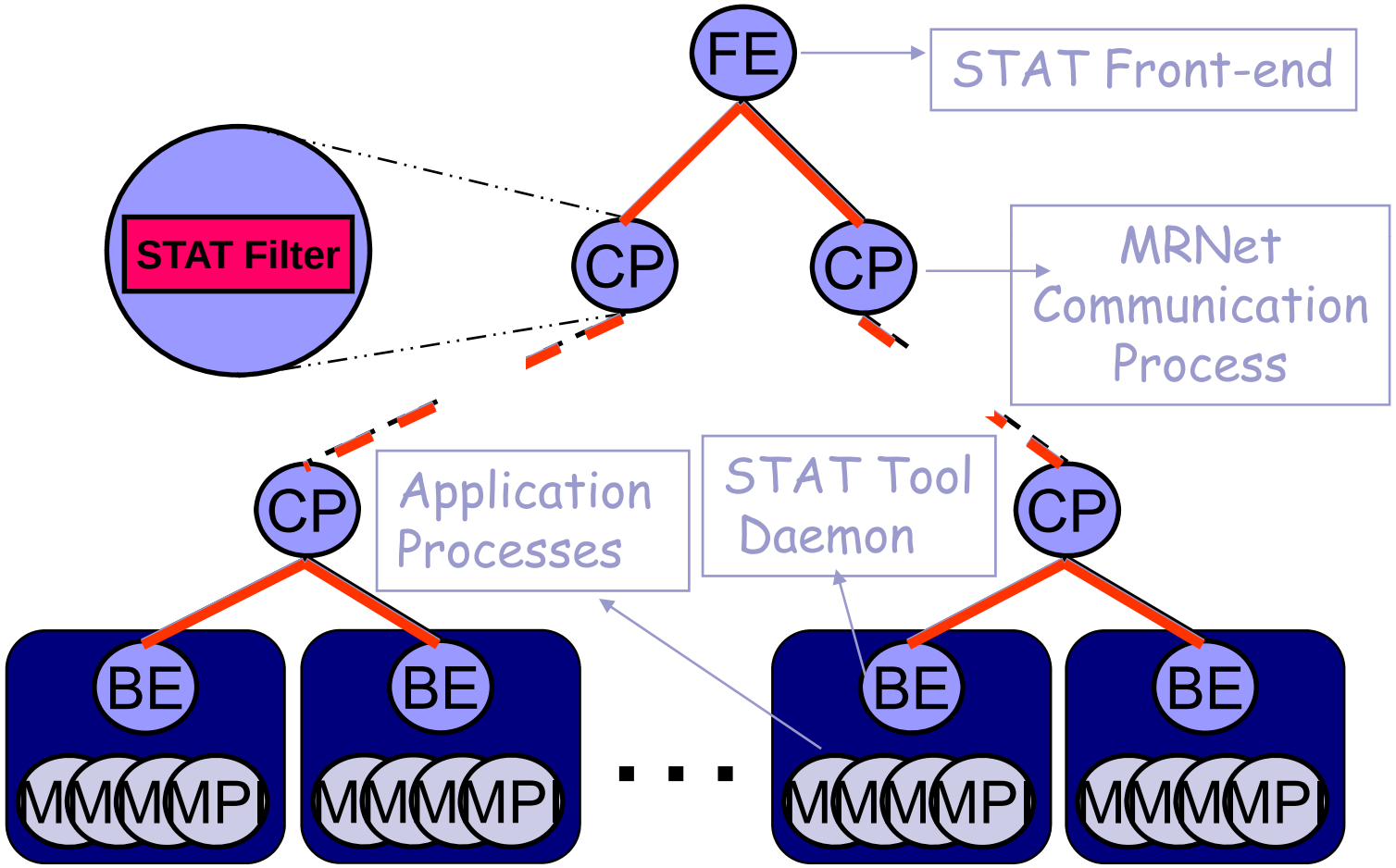


Implementing a Scalable Tool

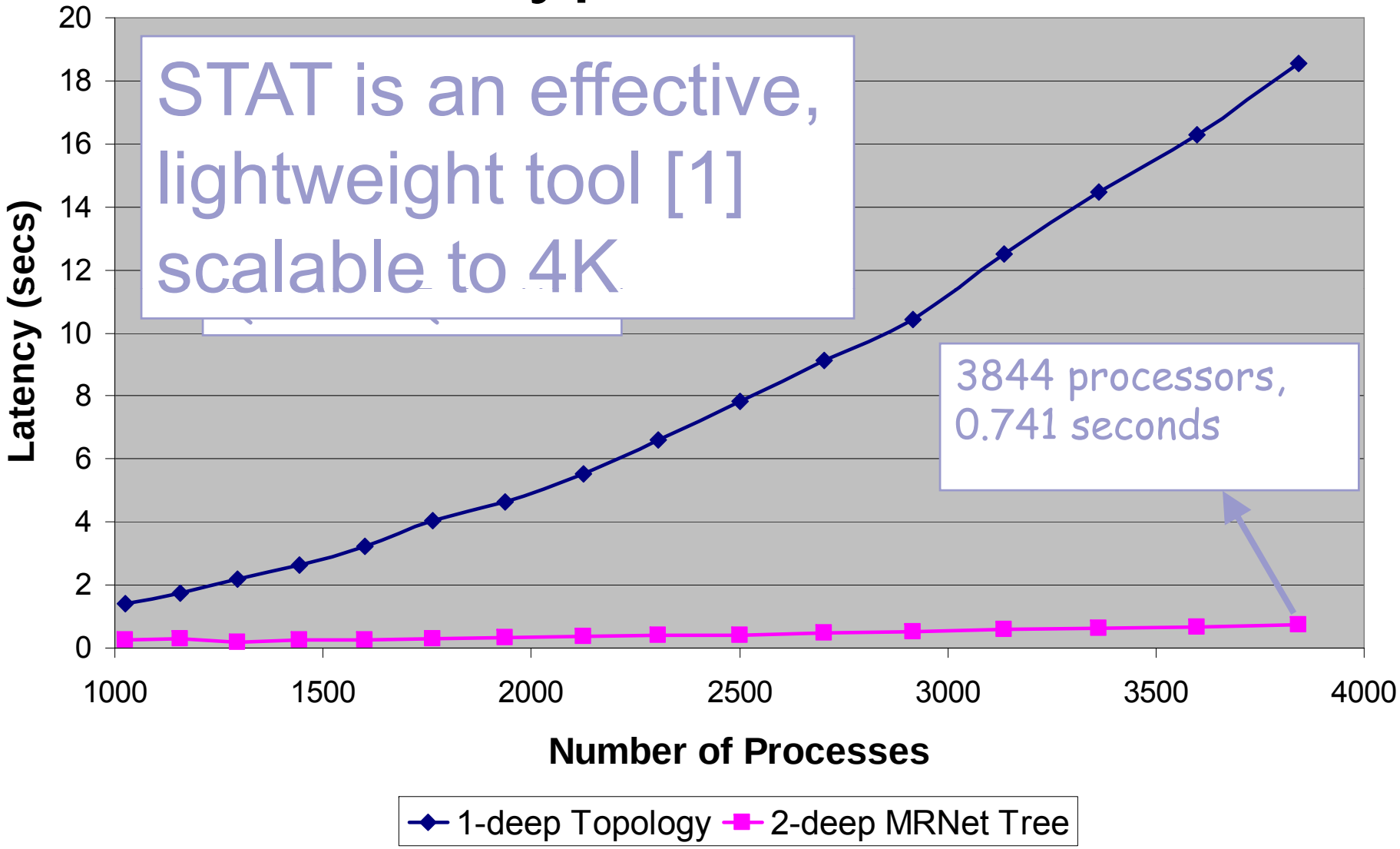
- Hierarchical Topologies
  - Application Control
  - Data collection
  - Data analysis
- A scalable solution: leverage Tree-Based Overlay Network (TBŌN) model
  - MRNet: A Multicast Reduction Network



STAT Architecture



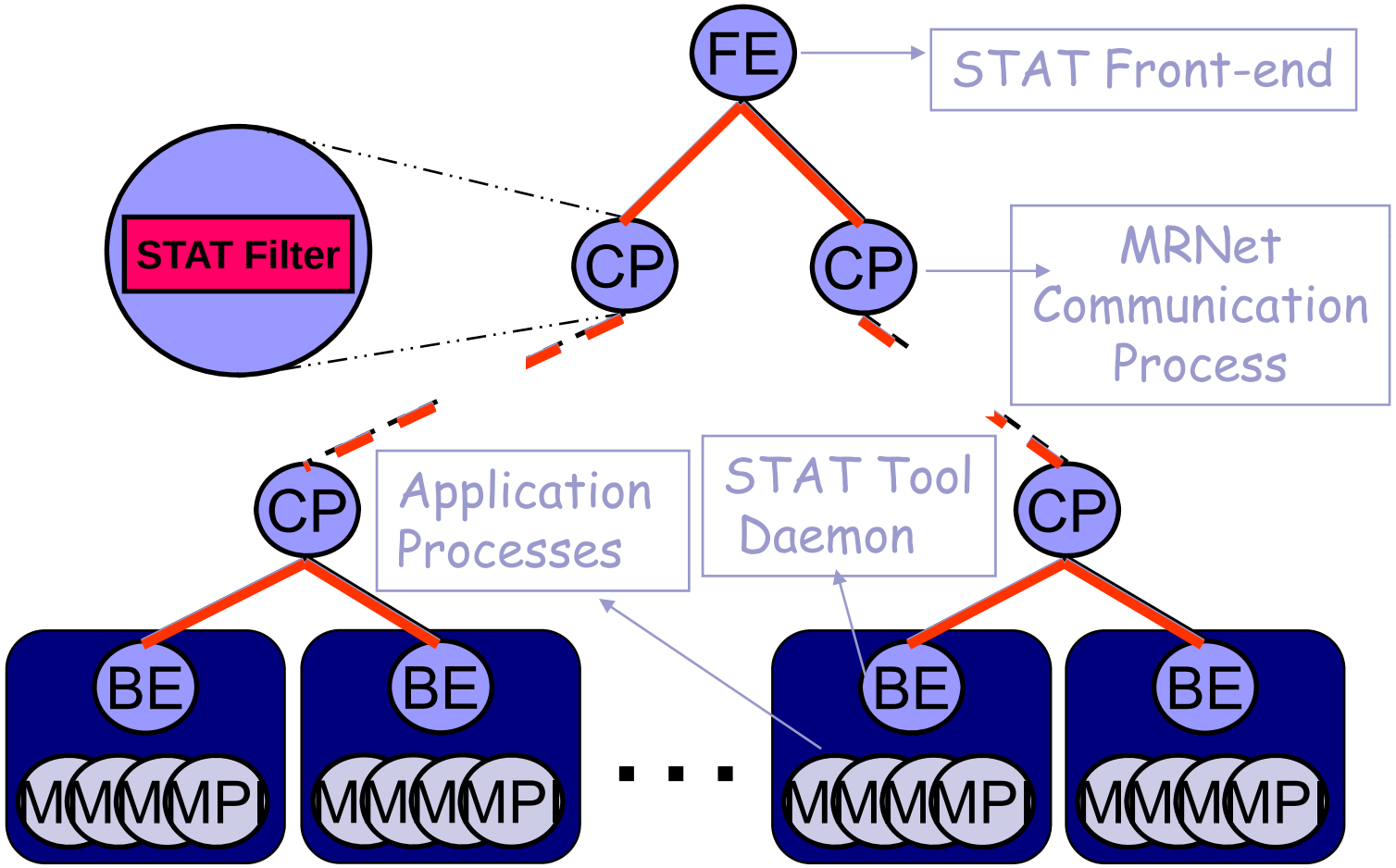
STAT Prototype Performance



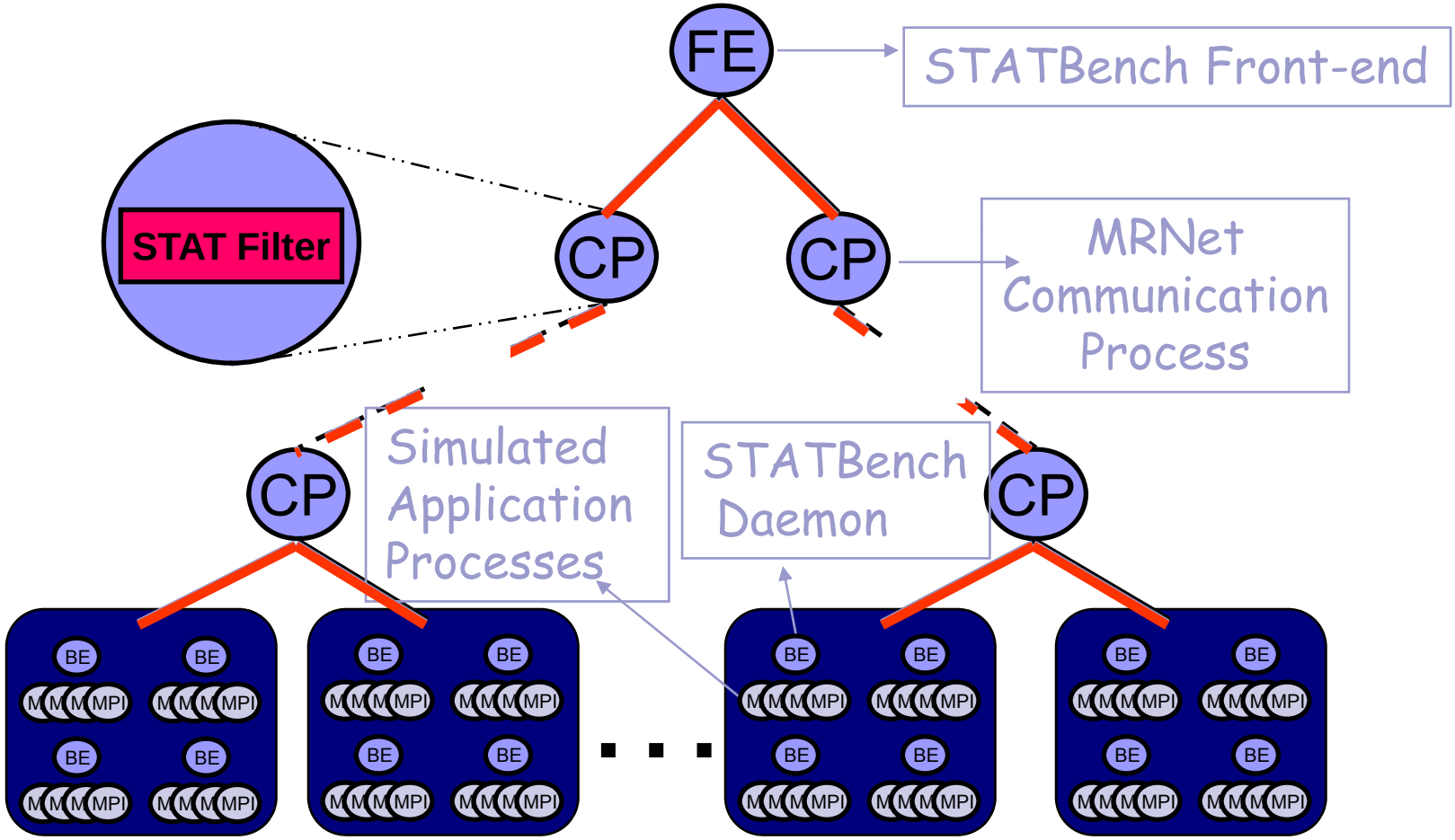
STATBench Emulates Large Scale STAT Runs

- Inherits code from our STAT Prototype
  - Communication via MRNet
  - Same merging/analysis routines
- 2 major differences
  - STATBench architecture
  - Stack trace generator

STAT Architecture



STATBench Architecture

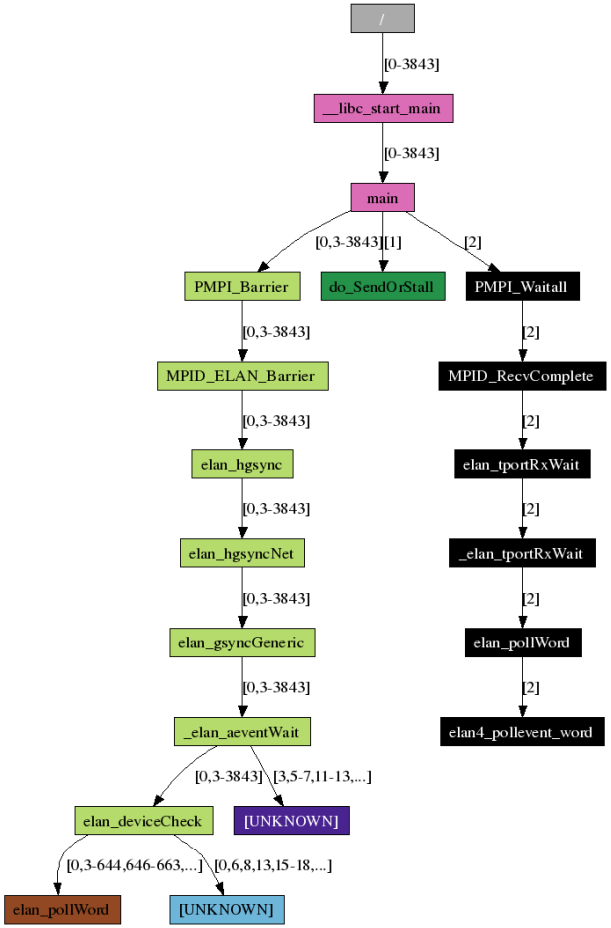


STATBench Stack Traces Model Application Behavior

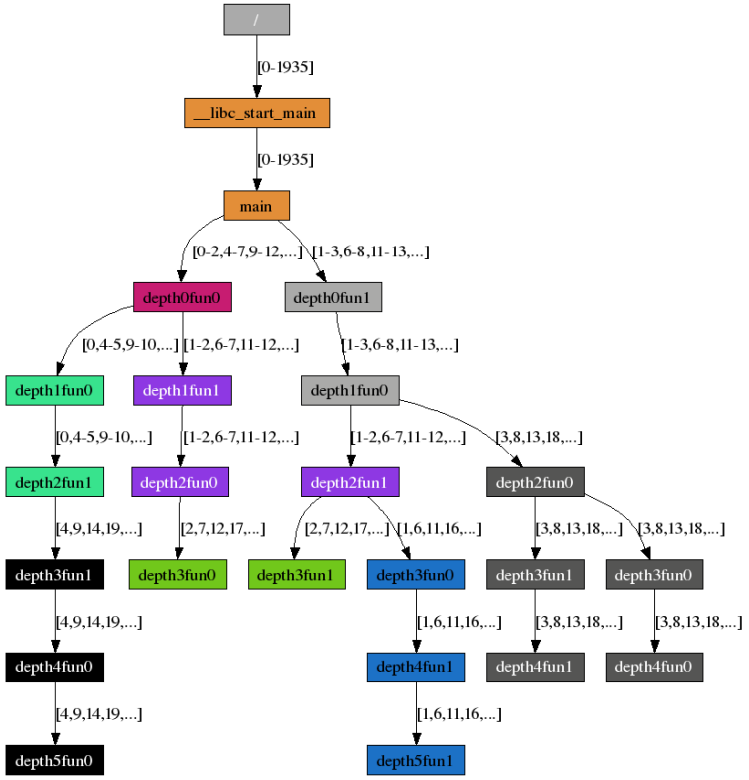
- Randomized call traces generated
- Controlled by several parameters:
  - Number of tasks per daemon
  - Number of traces per simulated task
  - Max call depth
  - Function branching factor
  - Number of equivalence classes

STATBench Validation

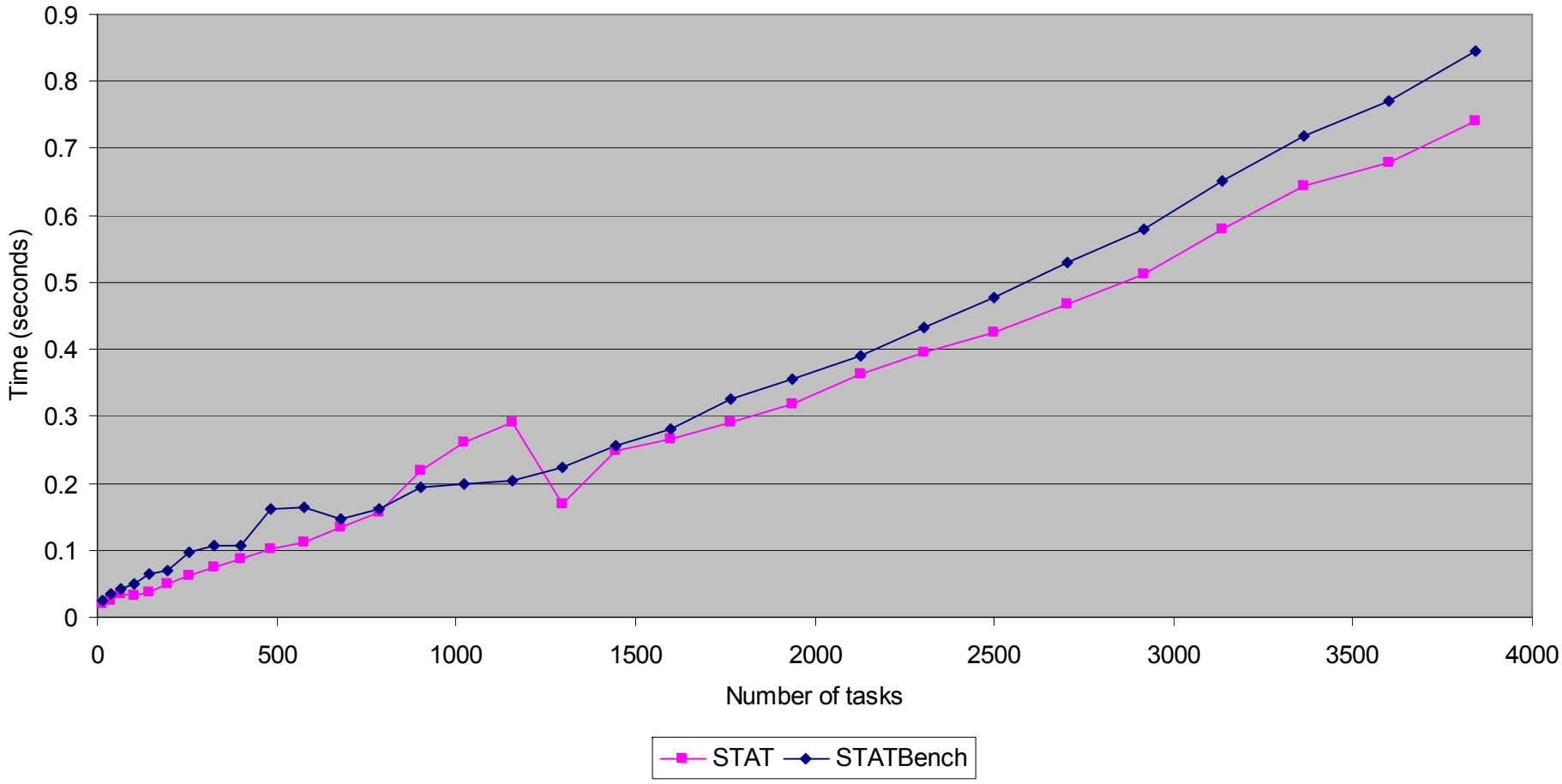
STAT on target application



STATBench approximation



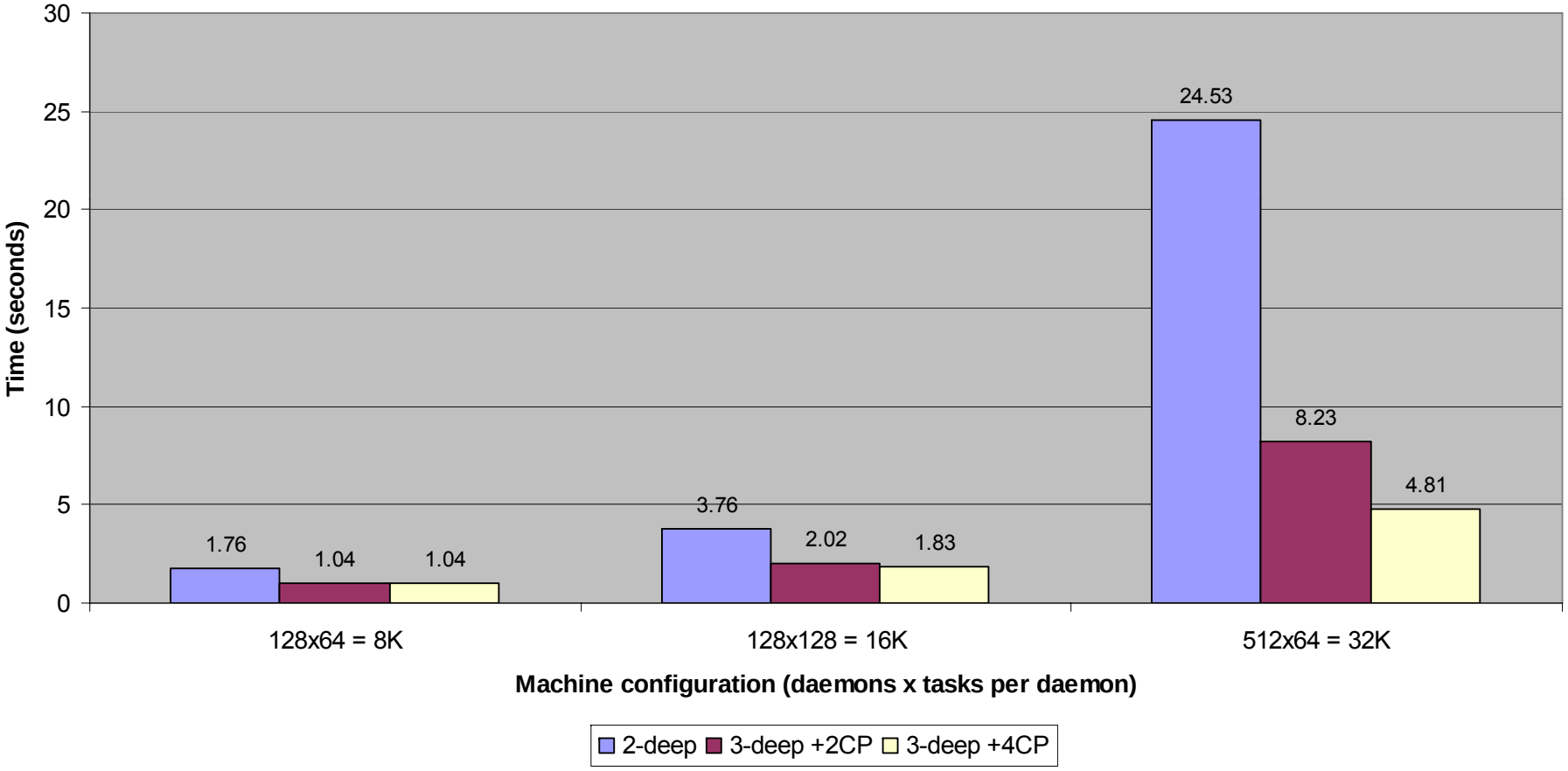
STATBench Performance Validation



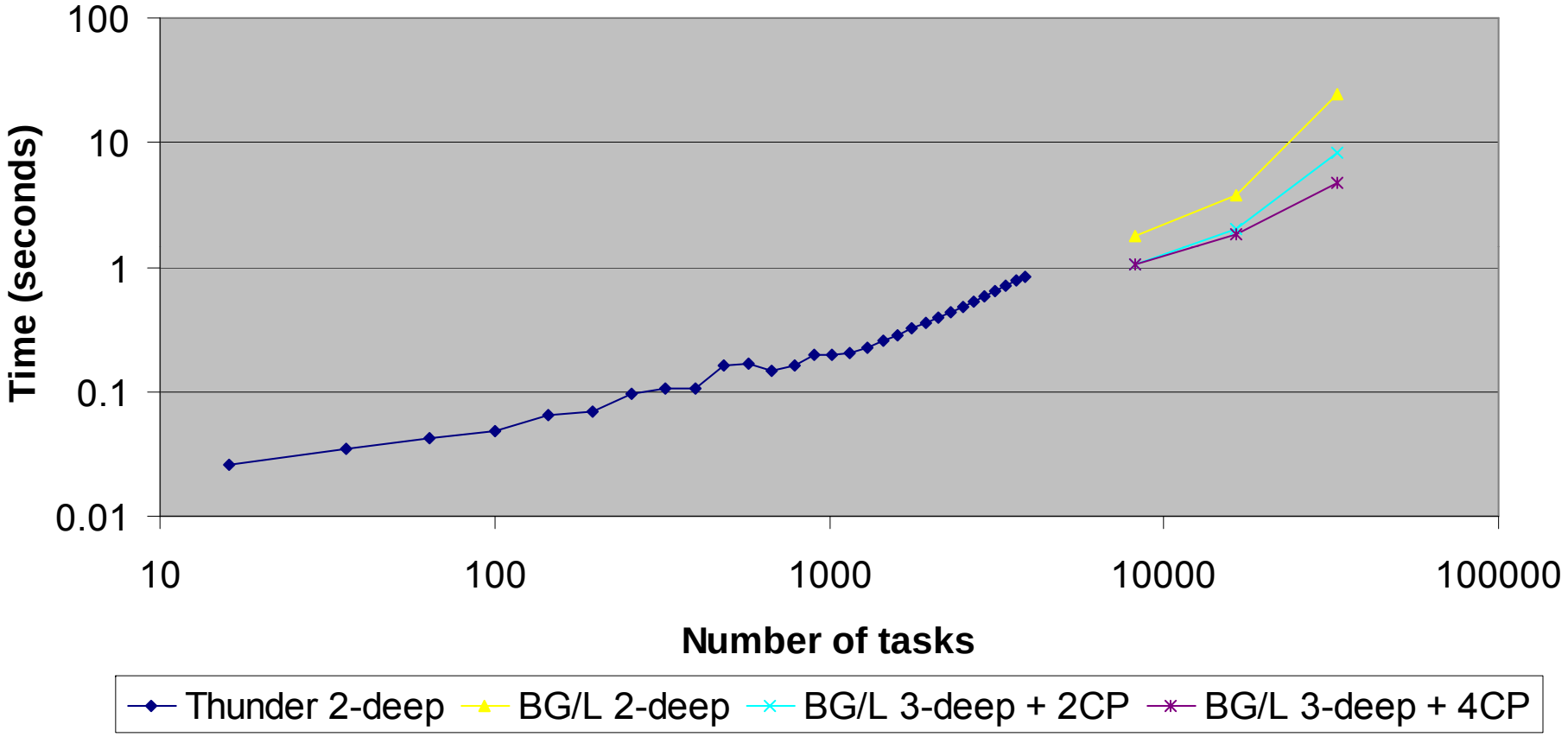
Emulating STAT on BG/L

BG/L Properties	STATBench Model
1024 I/O Nodes where tool daemons run	Ran on single rack, 1024 CN BG/L system
64 tasks per daemon in coprocessor mode	Simulate 64 tasks per STATBench daemon
128 tasks per daemon in virtual node mode	Simulate 128 tasks per STATBench daemon

STATBench BlueGene/L Scaling



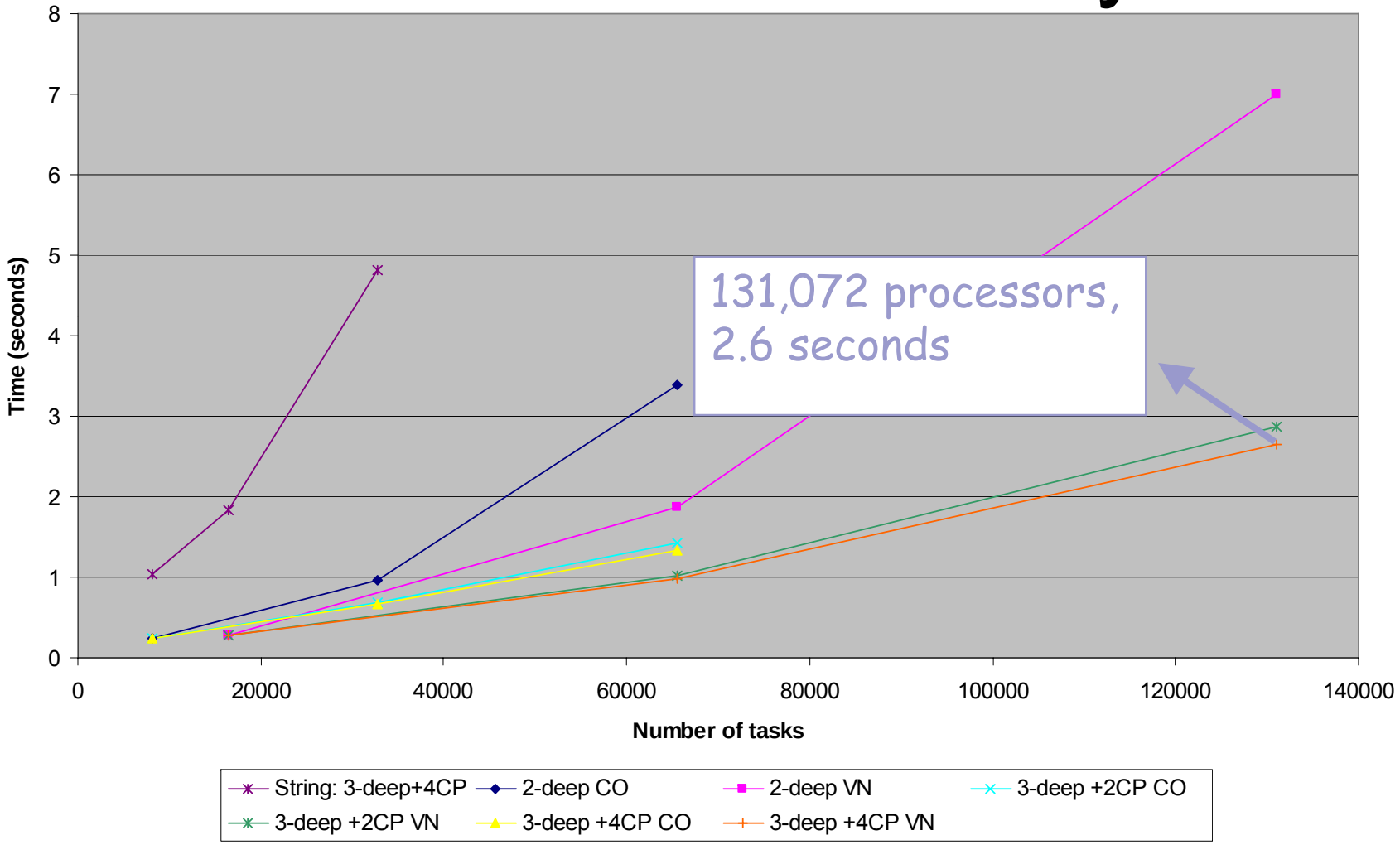
STATBench Performance Validation



STATBench Revealed a STAT Scalability Issue

- Edge labels represented by task lists
  - Original implementation as strings
  - [1,3,4,5,6,9,10,11,15] -> “[1,3-6,9-11,15]”
  - Up to 75KB at 32,768 tasks
- Re-implemented edge label as a bit vector
  - 1 bit per task
    - Set to 1 if the task is in the list
    - Set to 0 otherwise
  - Merging done with *bitwise or*

Full BlueGene/L Scalability



Conclusions

- STAT merges traces in an estimated 2.6 seconds running on full BG/L
- Emulation a powerful technique to test tool scalability
  - Run tool on all compute resources
  - Simulate application data
  - Decreased time and resource requirements to identify scaling issues

References

1. D. C. Arnold, D. H. Ahn, B. R. de Supinski, G. L. Lee, B. P. Miller, and M. Schulz, *Stack Trace Analysis for Large Scale Debugging*, 21st International Parallel and Distributed Processing Symposium 2007 **mar**, 2007
2. P. Roth, D. Arnold, and B. Miller, *MRNet: A Software-Based Multicast/Reduction Network for Scalable Tools*, Proceedings of the IEEE/ACM Supercomputing '03 **nov**,2003