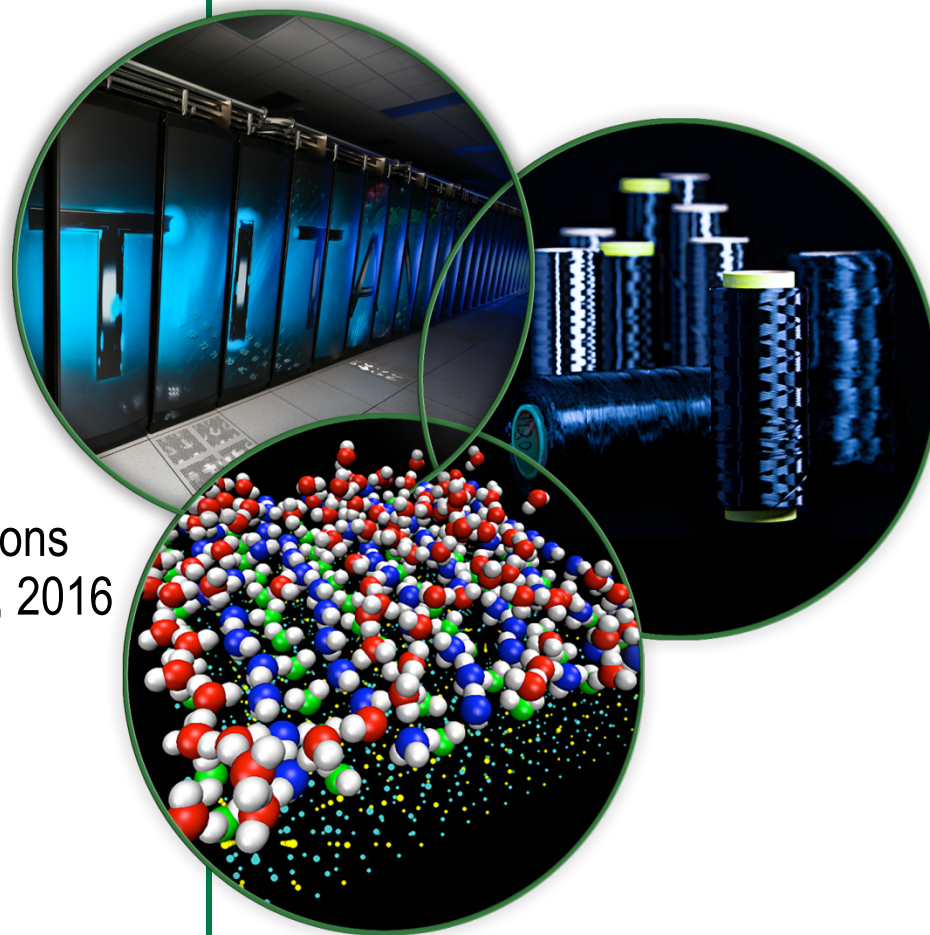


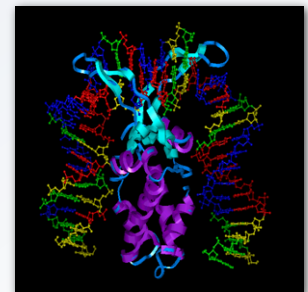
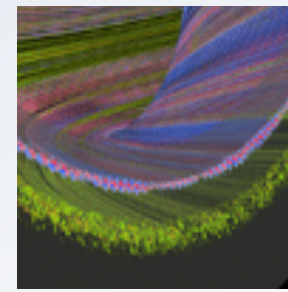
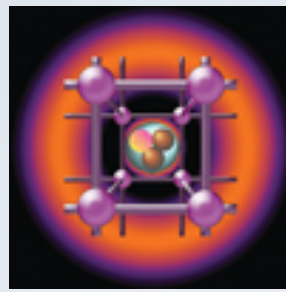
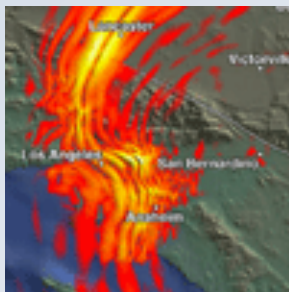
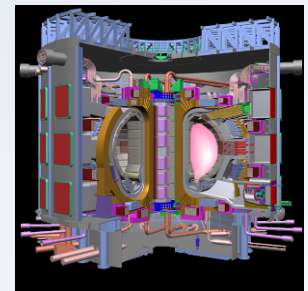
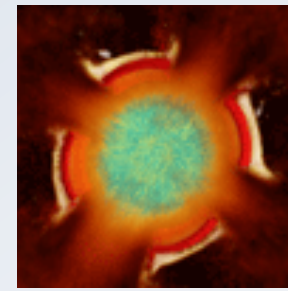
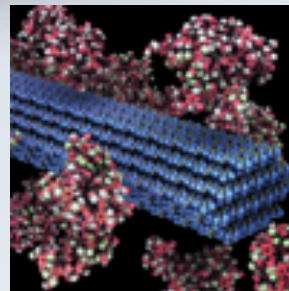
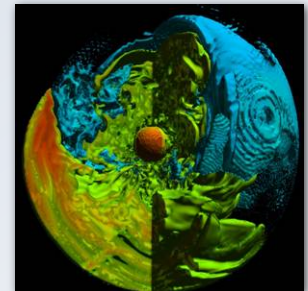
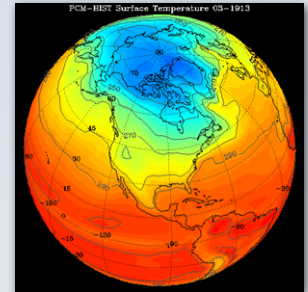
Benchmark Generation and Simulation at Extreme Scale

Mahesh Lagadapati, Frank Mueller, and Christian Engelmann

20th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications (DS-RT) 2016, London, UK, September 21-23, 2016



Scientific Computing and Simulation at ORNL



Background

- At the forefront of extreme-scale scientific computing
 - Titan at ORNL: Currently 3rd fastest supercomputer in the world
 - 560,640 cores (AMD Opteron + NVIDIA Kepler GPUs, 17.6 PFlops)
- We are on road to exascale computing: 1,000 Pflop/s by 2024
 - Potentially billions of densely-packed compute cores
- There are several major challenges:
 - *Power consumption*: Envelope of ~20-40 MW (drives everything else)
 - *Programmability*: Accelerators and PIM-like architectures
 - *Performance*: Extreme-scale parallelism (up to 1B hardware threads)
 - *Data movement*: Complex memory hierarchy and locality
 - *Data management*: Too much data to track and store
 - *Resilience*: Faults will occur continuously

HPC Hardware/Software Co-Design

- Helps closing the system-peak vs. application performance gap
- Aids in dealing with the power-performance-resilience trade off
- Execution of real applications, algorithms, communication traces or their models atop a ***simulated*** HPC environment at scale for:
 - Application/algorithm performance evaluation, including identification of scaling limitations and load imbalances
 - Architecture performance evaluation, including identification of resource contention and underutilization issues
 - Investigation of novel node architectures, beyond today's manufacturing capabilities and at lower cost than hardware prototyping
 - Investigation at extreme scale, beyond the capabilities of existing HPC systems and at lower cost than hardware prototyping

Motivation

- Execution Message Passing Interface (MPI) applications on real or simulated systems at various scales for performance analysis is limited
 - As a run at each scale takes a significant amount of execution time
 - Scaling beyond today's capabilities requires oversubscription, i.e., an even longer execution time
 - Application codes are often not easily portable to new architectures
 - Some application codes are proprietary or have other access limitations, such as industry or export controlled codes
- Replay of MPI application communication traces on real or simulated systems mitigates some of these, but has its own limitations
 - Replays are limited to the scale of the original application run
 - Replays take a lot of time due the size of MPI traces and due the requirement to emulate computation with process sleeps

Contribution

- This work combines the following technologies to create a novel infrastructure for HPC co-design
- The **ScalaTrace II** tool utilizes innovative algorithms to extract MPI traces and to expand elastic MPI traces, allowing replays at various scales
- The **ScalaBenchGen II** tool generates concise and accurate benchmark programs from ScalaTrace II traces, abstracting proprietary code
- The **xSim** tool runs ScalaBenchGen II benchmarks at various scales on a simulated HPC system, without the need to replay the original MPI traces or run the original code

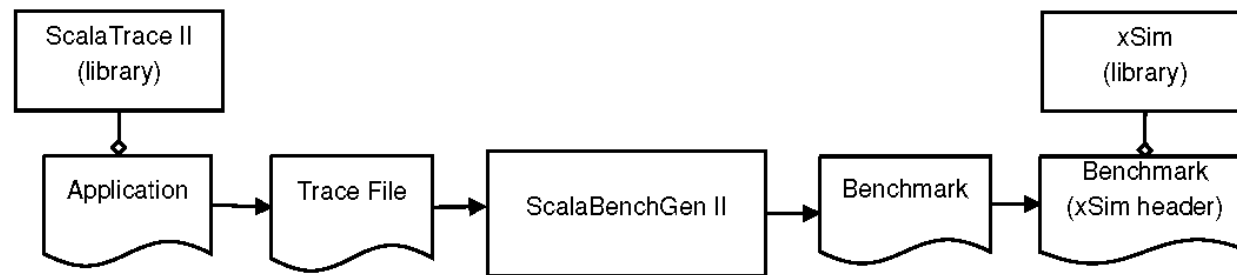


Fig. 1. Benchmark Generation and Simulation Process

ScalaTrace II

- State-of-the-art tracing framework that produces near constant size MPI communication traces for parallel applications based irrespective of application execution scale
- Traces preserve both, structural information and temporal ordering of events, and can be replayed to reproduce the original application behavior
- Message content (payload) is not recorded
- Performs trace compression in two stages: intra and inter node
 - Intra node compression is performed on the fly during tracing
 - Inter node compression is performed at MPI_Finalize()
 - Loops in the program are identified and represented in the trace
 - Computation time is preserved using statistical histogram bins

ScalaBenchGen II

- Creates communication benchmarks from ScalaTrace II traces
 - Uses ScalaTrace II's elastic data element representation to closely reconstruct program structure, including loops
 - Simulates computation time by introducing a sleep() call
 - The generated benchmarks are concise and readable
- The number of events in a loop can be different per rank and iteration of an outer loop.
 - An event can be part of a loop in one rank but may not be part of the loop in another rank.

```
Rank 0:
1:  for(i=0;i<10;i++) {
2:      E1;
3:      E2;
4:      E3;
5:  }

Rank 1:
1:  for(i=0;i<20;i++) {
2:      E1;
3:      E2;
4:  }
5:  E3;
```

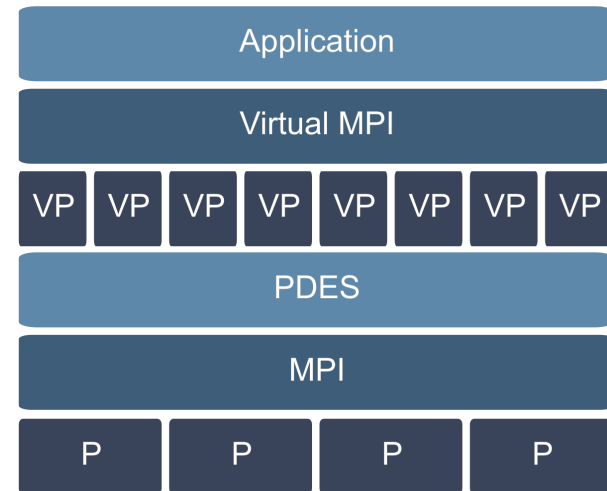
Fig. 2. Rank specific behavior in loops

```
iter_cnt = get_param('10,0;20,1');
mem_cnt = get_param('3,0;2,1');
for(iter=0; iter<iter_cnt; iter++){
    if(is_loop_member(...))
        E1;
    if(is_loop_member(...))
        E2;
    if(is_loop_member(...))
        E3;
}
```

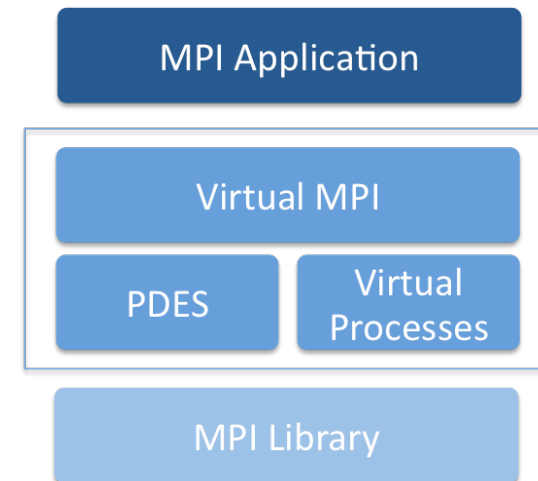
Fig. 3. Benchmark code for the loop with rank specific behavior

xSim – The Extreme-Scale Simulator

- A simulation-based performance/resilience investigation toolkit for MPI applications
- Combining oversubscribed execution, a virtualized MPI & POSIX API, and a time-accurate parallel discrete event simulation
- Support for C, C++ and Fortran applications
- Easy to use:
 - Compile the application with xSim header
 - Link the application with the xSim library
 - Run: `mpirun -np <np> <application> \`
`-xsim-np <virtual process count> \`
`<simulation parameters> \`
`<application parameters>`



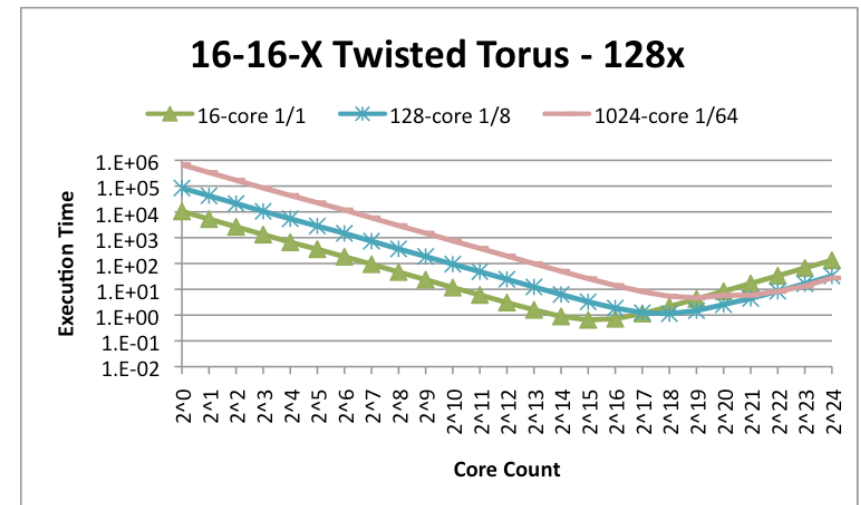
Architecture



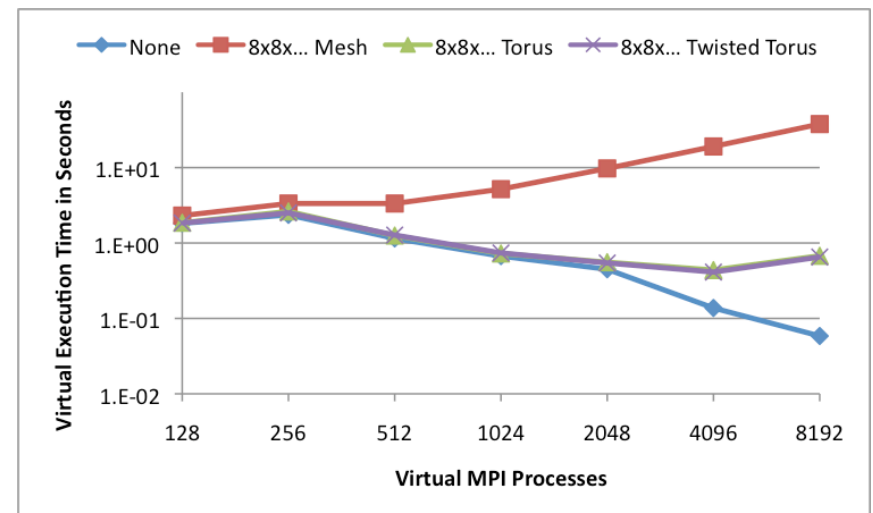
Design

xSim Performance Simulation Features

- Simulated timing with scaling processor model
 - Application executed on native processor
 - Simulated timing is relative to native
 - Operating system noise injection
 - Does not support accelerators yet
- Simulated timing with network model
 - Completely simulated network at the MPI messaging level
 - Latency, bandwidth, contention, routing, and rendezvous protocol
 - Star, ring, mesh, torus, twisted torus, and tree topologies
 - Hierarchical combinations of topologies, e.g. on-chip, on-node, & off-node



Monte Carlo solver w/ different core speeds/counts



NAS MG in a dual-core 3D mesh or twisted torus

Evaluation - Platforms

- Physical system
 - Linux cluster with 108 compute nodes (216 processors, 1,728 cores)
 - Each node has 2 AMD Opteron 6128 (Magny Core) processors
 - Each processor has 8 compute cores, i.e., each node has 16 cores
 - Fat tree InfiniBand network interconnect
 - Off-node L1 switch: 2547 MB/sec and 13 μ s
 - Off-node L2 switch: 2546 MB/sec and 14 μ s
- Simulated system (xSim)
 - 108 compute nodes with 2 processors per node and 16 cores per proc.
 - Sleep calls are simulated and just advance the simulated clock
 - Fat tree network interconnect w/o network contention & w/ linear coll.
 - Off-node L1 switch: 2,547 MB/sec and 13 μ s
 - Off-node L2 switch: 2,546 MB/sec and 14 μ s

Evaluation - Applications

- NASA Advanced Supercomputing Division Parallel Benchmark
- Mini applications derived from HPC application codes, including computational fluid dynamics, unstructured adaptive mesh, parallel I/O, multi-zone, and computational grid applications
 - BT, a Block Tri-diagonal solver (64, 256 and 1024 MPI procs.)
 - SP, a Scalar Penta-diagonal solver (64, 256 and 1024 MPI procs.)
 - MG, a multi-grid solver on a sequence of meshes with long- and short-distance communication patterns and memory intensive operations (256, 512 and 1024 MPI procs.)
 - CG, a conjugate gradient solver with irregular memory and communication patterns (256, 512 and 1024 MPI procs.)
 - LU, a Lower-Upper Gauss-Seidel solver (256, 512 and 1024 MPI procs.)

Accuracy of ScalaGen II Benchmarks

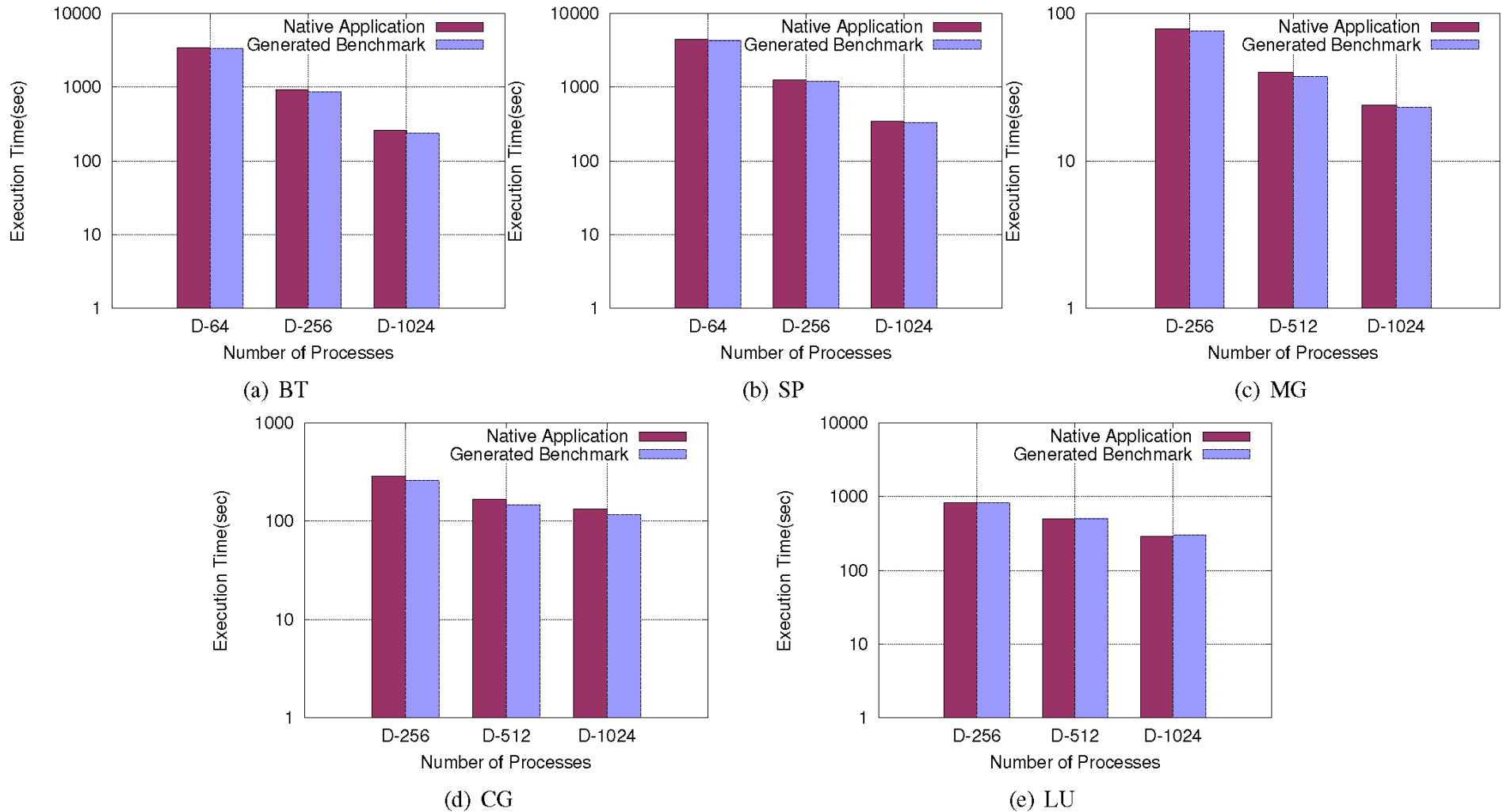


Fig. 4. Accuracy of ScalaBenchGen II Benchmarks

Accuracy of Benchmarks in xSim

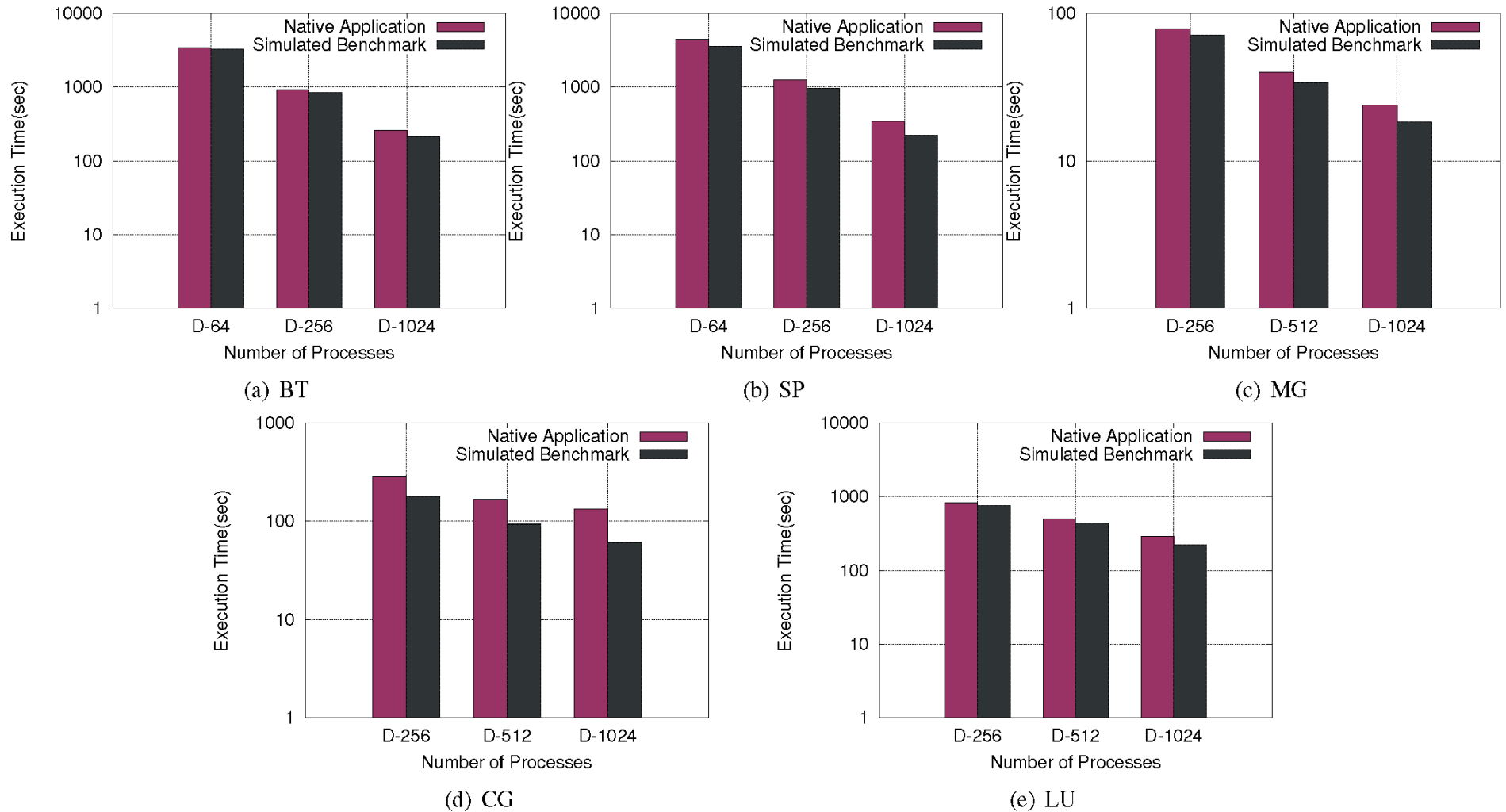


Fig. 8. Accuracy of Simulation

Simulation Time for Benchmarks in xSim

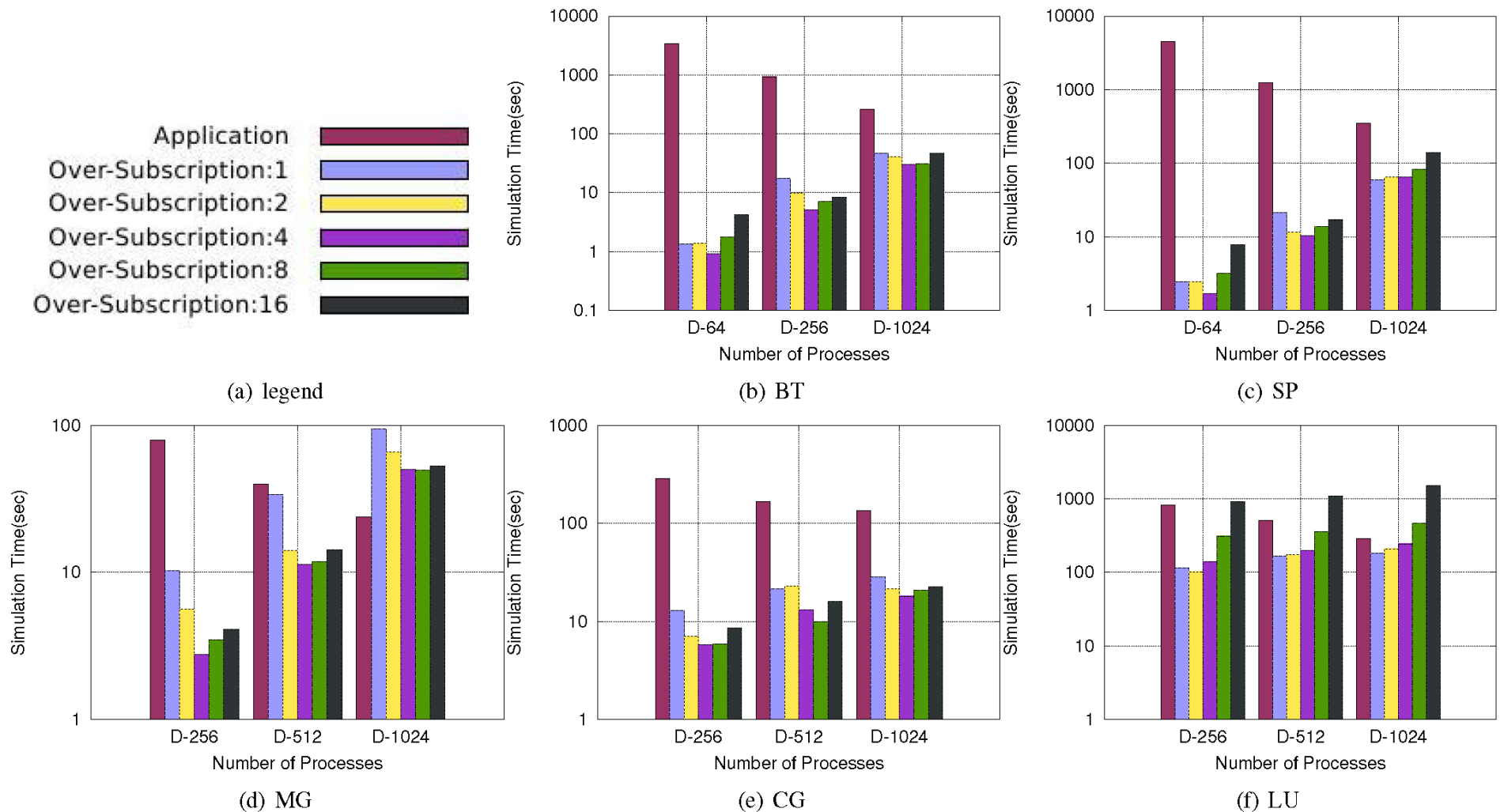


Fig. 7. Simulation Time for Different Over-Subscription Scenarios

xSim's MPI Message Data Amount

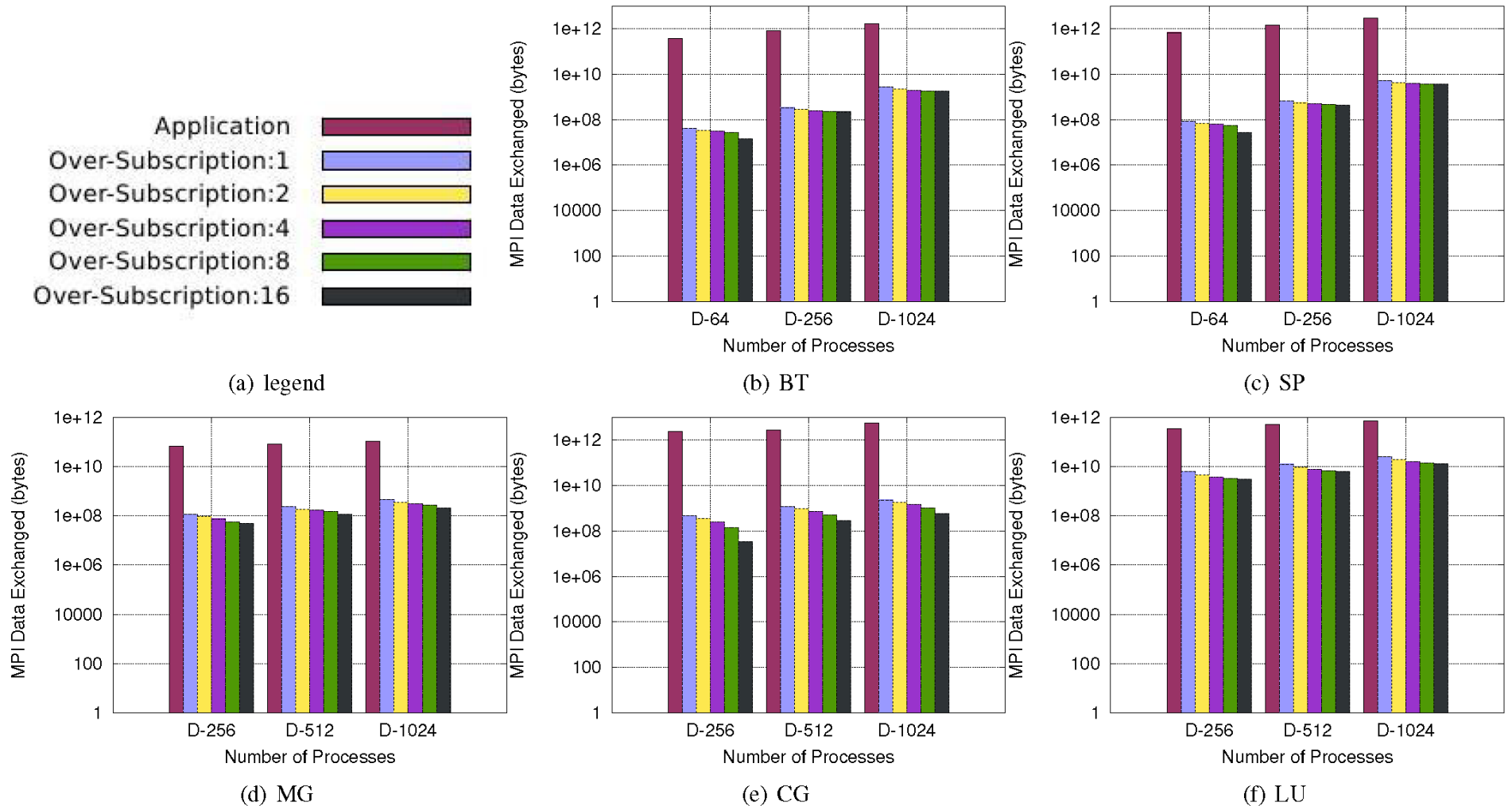


Fig. 5. MPI Data Exchanged for Different Over-Subscription Scenarios

xSim's MPI Message Count

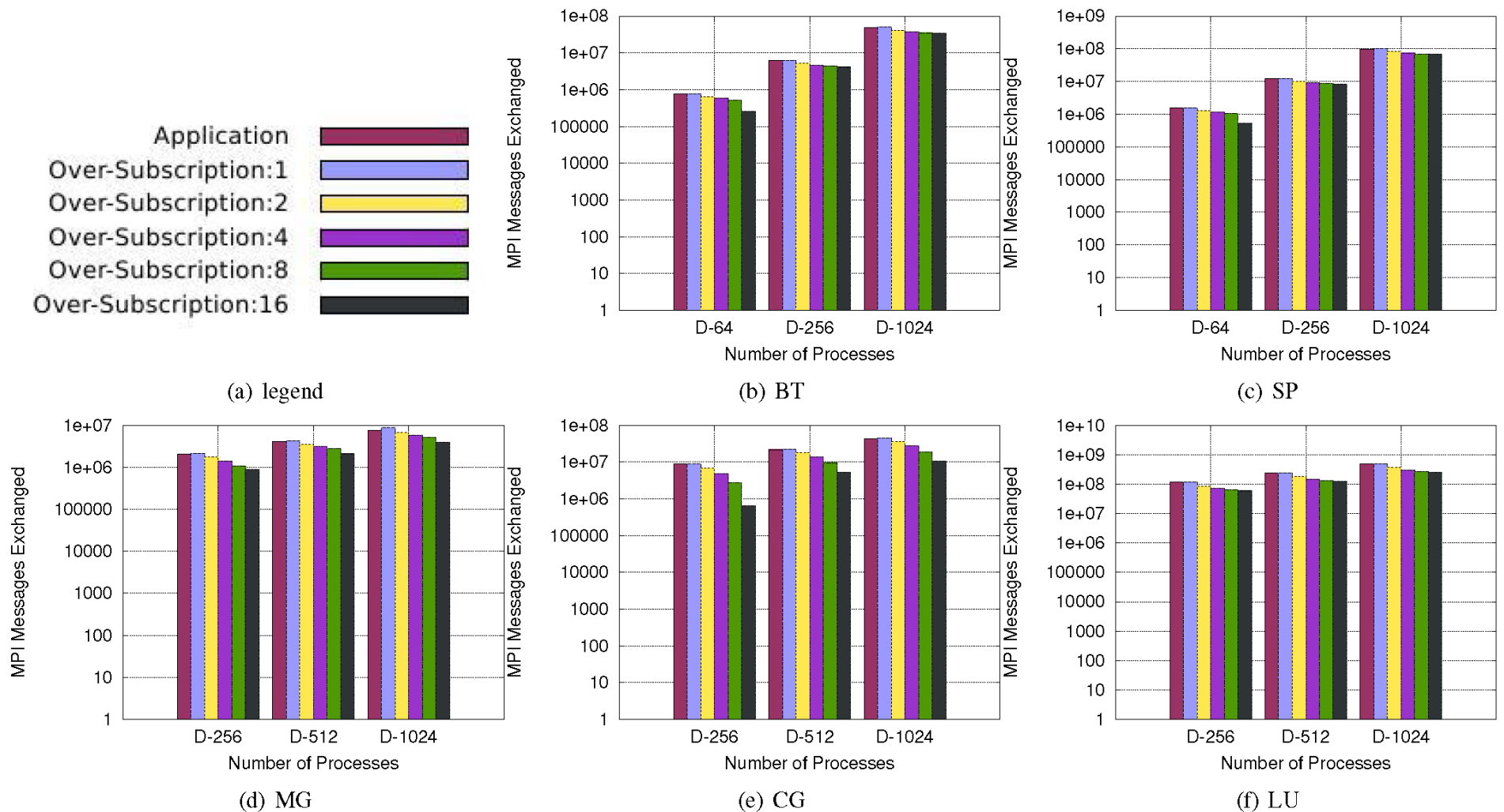


Fig. 6. MPI Messages Exchanged for Different Over-Subscription Scenarios

Conclusion & Future Work

- Demonstrated the capability to drive HPC co-design using simulation by combining ScalaTrace II, ScalaBenchGen II and xSim into a novel tracing/benchmarking/simulation framework
- ScalaTrace II produces a constant-size, near lossless, scalable and elastic communication trace
- ScalaBenchGen II produces a benchmark code from the trace, abstracting proprietary code
- xSim executes the benchmark on a simulated HPC system without the need for time consuming application execution or trace replay
- Future work could focus on fault tolerant MPI applications

Questions



M. Lagadapati, F. Mueller, C. Engelmann. Benchmark Generation and Simulation at Extreme Scale. DS-RT 2016.