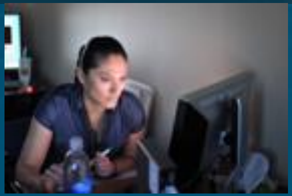
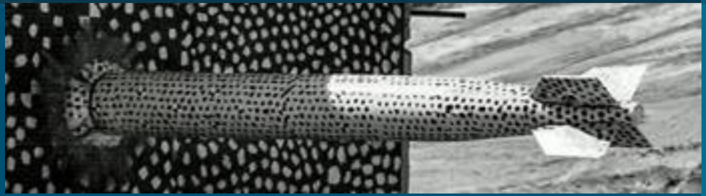
Towards Lightweight and Scalable Simulation of Large-Scale OpenSHMEM Applications





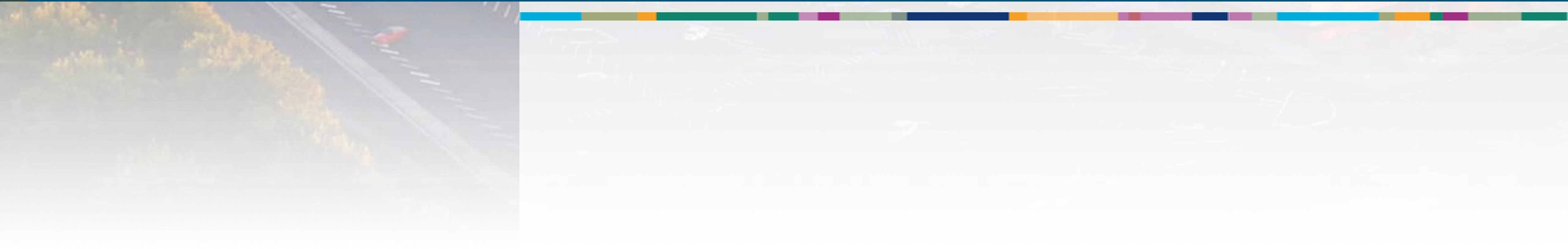
PRESENTED BY

M.J. Levenhagen, S.D. Hammond and K.S. Hemmert

Center for Computing Research, Sandia National Laboratories
wg-sst@sandia.gov

Talk Outline

Background and Context



Background and Context – Changing Application Requirements

High Performance Computing Workloads are changing quite significantly from historical trends

- Historically applications have used almost exclusively two-sided MPI and collective operations
- Typical use case for high-performance scientific applications (PDEs, FFTs, Linear Solvers etc..)

Growing interest in *newer* classes of workloads, typically as a component in a much larger workflow

- Driven partly by growing capabilities in machine learning and data analytics domains
- But also, improving hardware support for fine-grained communication, smaller more randomized accesses across the system

DOE is still trying to understand how to use algorithms like machine learning in its broader mission workloads

- Complicated by demands of reproducibility, traceability *etc* in high consequence modeling departments etc

Background and Context – Analyzing and Projecting Performance

Diminishing returns for near-term hardware is pushing DOE to think much harder about what systems are being purchased and what they will run

- Want to find more efficient matches between workloads and hardware solutions
- Tailor purchased systems so they are more tuned towards user requirements
- Significant focus within the DOE's Exascale Computing Project (ECP) and several DOE-led procurements

Requires high-performance, scalable and very flexible performance analysis/simulation capabilities

- Perform what-if analysis on various scenarios
- Deeper insight into *why* systems behave as they do (and therefore what we can do about it)

Combined with changing workload requirements – presents a challenging environment for hardware/system architects

Structural Simulation Toolkit (SST)

Architecture Analysis/Simulation Toolkit which is designed to provide high-performance, scalable simulation capabilities

- *Use supercomputers of today to design the supercomputers of tomorrow*
- MPI and C++11 Thread-based Parallel Discrete Event Simulation (PDES) core

Range of model fidelities – cycle-accurate through to analytical/coarse-grained solutions to enable users to pay for complexity when they need it

- On-node models typically cycle-based (e.g. processor, memory, NoC)
- Networking models can vary (e.g. cycle-based switch/router model, coarse-grained end-points etc)

Open-Source, Open API, easily extensible and modular

- Works with variety of other community simulators
- Vendor contributions, consistent with BSD license (Cray, Intel, ARM, NVIDIA, IBM, *etc)*

More Information: http://sst-simulator.org

Extending SST to Model OpenSHMEM Communication Patterns

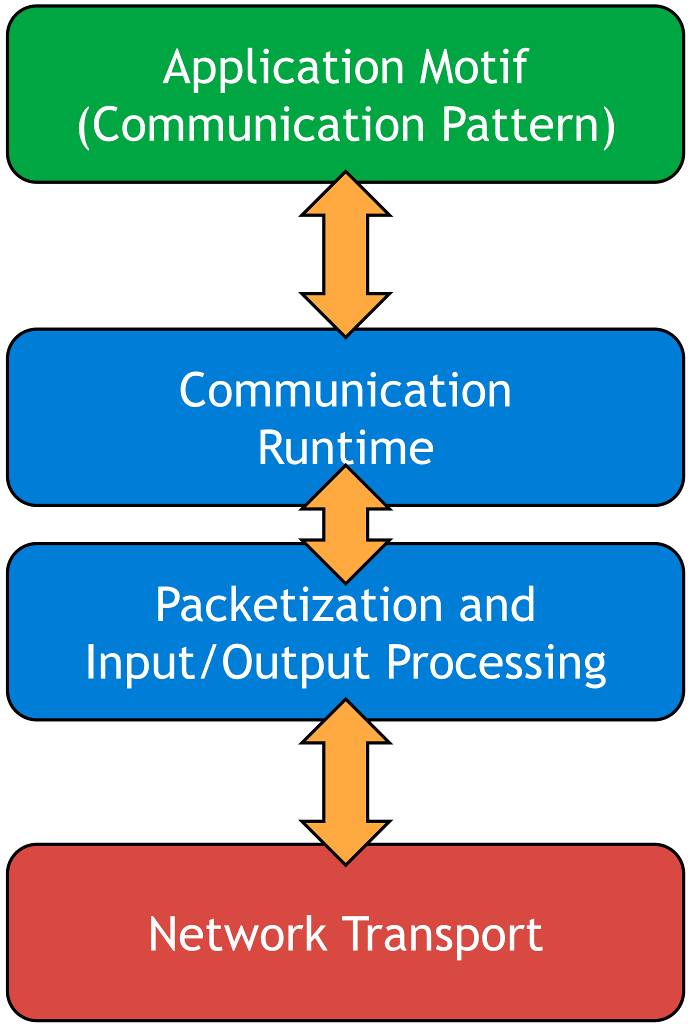
Modeling MPI Communication Patterns in SST

SST has well established models for communication patterns written to MPI two-sided/collective operations

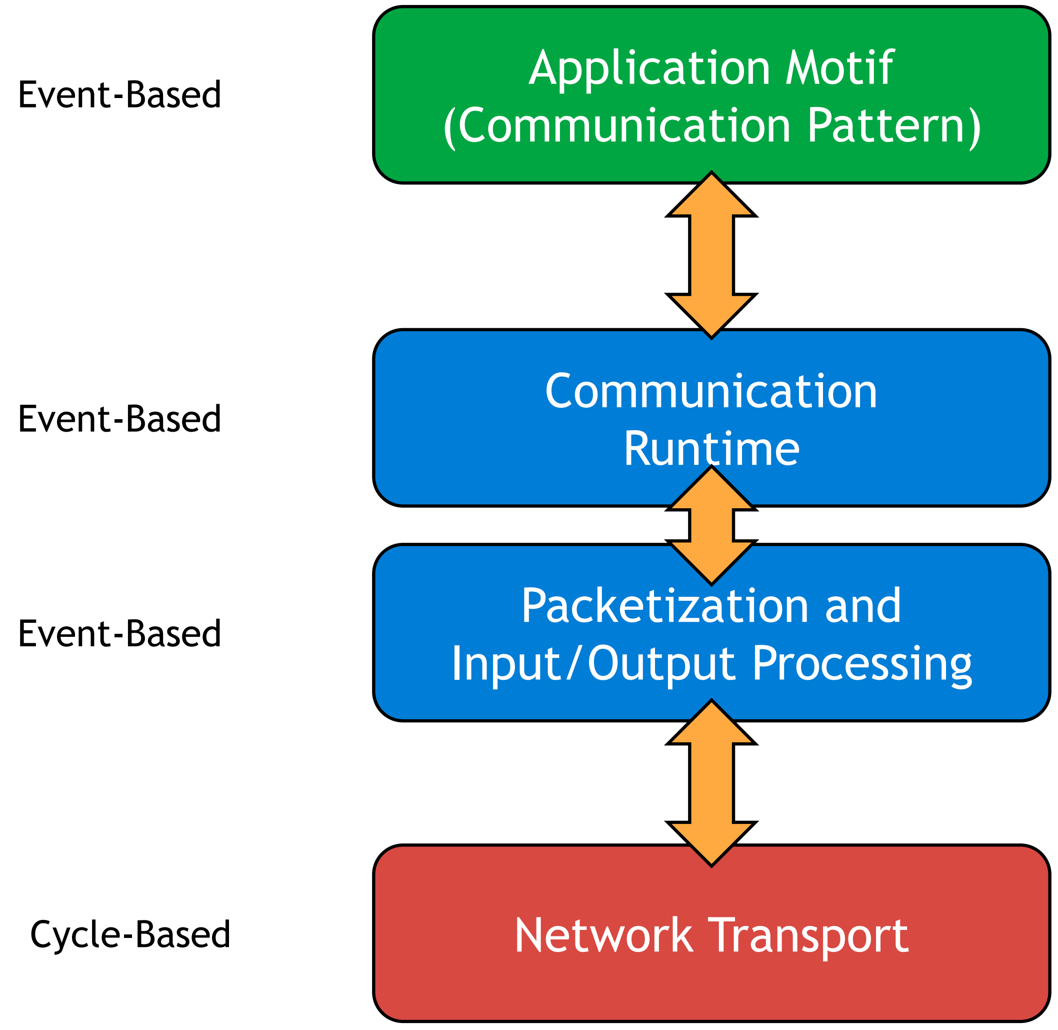
- Developed for modeling DOE systems during procurements
- Demonstrated using SST to over 1M simulated MPI ranks (beyond our largest systems)
- Provides access to a small collection of MPI functions (we do not implement the entire MPI standard)

Communication Patterns (”Motifs”) are small snippets of code which replicate communication behavior

- Very lightweight to execute, low overhead, small state to improve simulation performance
- Typically easy to write by application experts
- Can be arranged as a collection to replicate larger workflow/full application behaviors


Application Motif (Communication Pattern)
Communication Runtime
Packetization and Input/Output Processing
Network Transport

Components used for SST Communication Modeling


Event-Based
Application Motif (Communication Pattern)
Event-Based
Communication Runtime
Event-Based
Packetization and Input/Output Processing
Cycle-Based
Network Transport

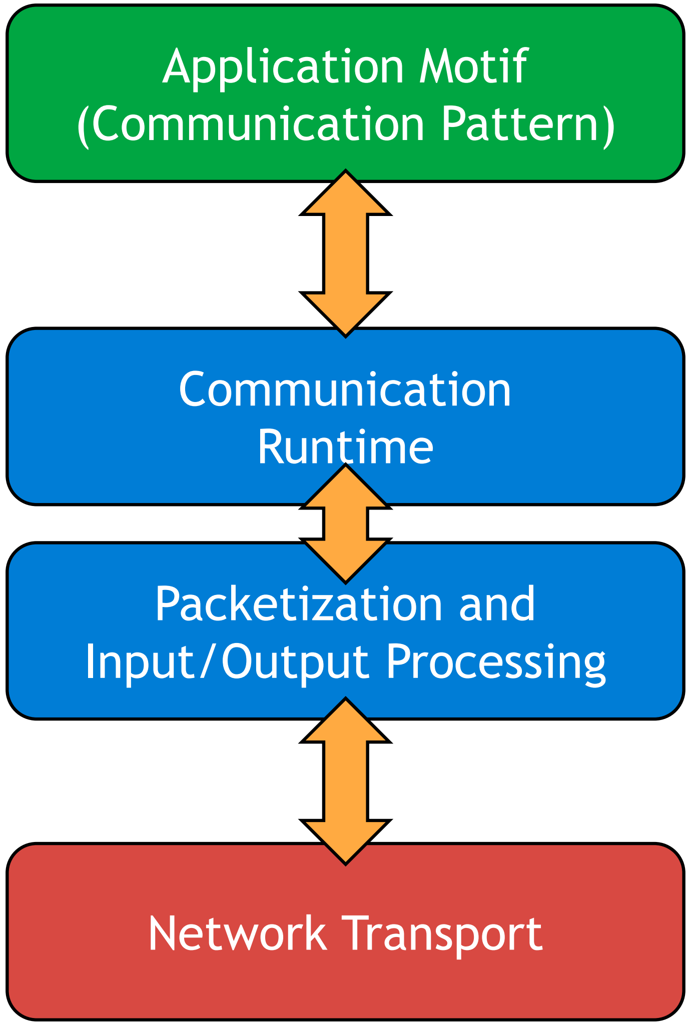
“Ember” – Collection of Communication Patterns (replicate common patterns, benchmark cases etc)

“Hades” – respond to events from the motif to implement high-level communication runtime operations (e.g. MPI_Send)

“FireFly” – respond to events from Hades to implement data (DMA) transfers with memory timing etc

“Merlin” – network transport, congestion, adaptivity modeling etc

Extending SST for OpenSHMEM



“Ember” – extended to include new communication patterns written to OpenSHMEM semantics and being more representative of operations we are being asked to model.

“Hades” – respond to OpenSHMEM-semantics

“FireFly” – significant increase in memory subsystem modeling requirements (caches, TLBs), atomic operation processing in the NIC (see later slides)

“Merlin” – no significant changes needed

MPI models typically are critical-path on tag/operation matching meaning overheads elsewhere in system require low fidelity models. Not true for OpenSHMEM.

Redesigned Node-Model for OpenSHMEM Modeling

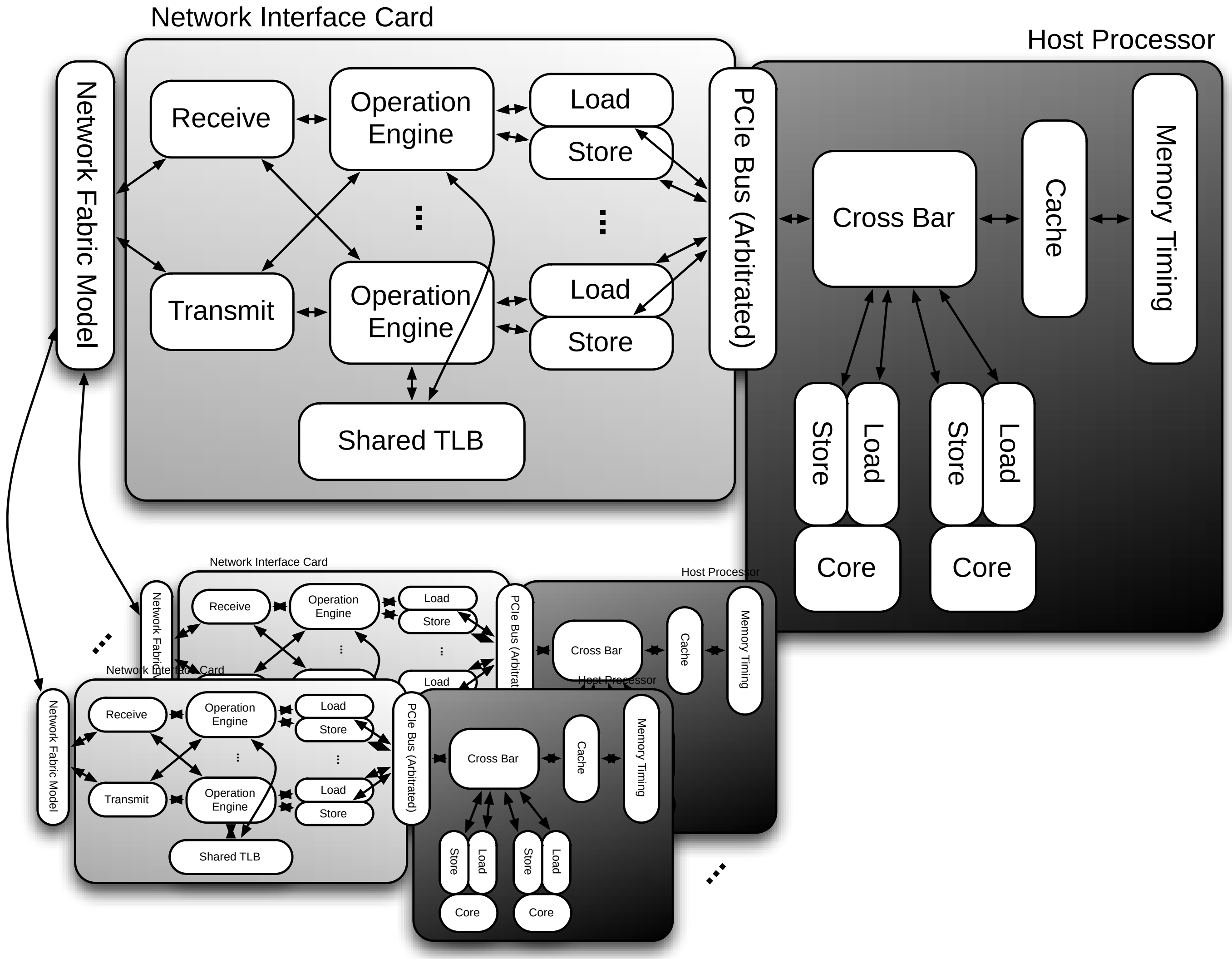
Significant increase in model complexity for on-node resources

- Driven by fine-grained nature of OpenSHMEM operations where smaller overheads quickly dominate
- Requires fidelity in memory system components because of workload data access patterns (e.g. random)

On-node Communication still by-passes NIC

- See cross-bar modeling on host-processor

Driven by push to validation Cray XC40/Aries hardware



Evaluating OpenSHMEM Communication Patterns

Benchmarks for Evaluating SST OpenSHMEM Implementation

Developed several example micro-benchmarks to stress OpenSHMEM models and overhead modeling

Randomized Remote Atomic Increment (“GUP/s”) – selects a remote PE from a uniform random distribution, selects an address in the remote memory space from uniform random distribution and performs a 32-bit atomic increment operation

Randomized Remote Atomic Increment with Hot-Spot (”Hot-Spot GUP/s”) – performs the same operations as standard GUP/s but weights PE selection towards a “victim” node to model the effect of high-degree vertices in a large-scale graph

Randomized Remote Increment with Value Return (“FINC-GUP/s”) – performs a GUP/s like operation but requires that the value is returned to the caller. Drives modeling of the return path timing.

We also perform some basic PUT/GET operation timing to ensure basic timing operations are understood.

Benchmarks are implemented/online at: https://github.com/sstsimulator/ember

Validation Activity – Comparing SST Models to Cray XC40 Hardware

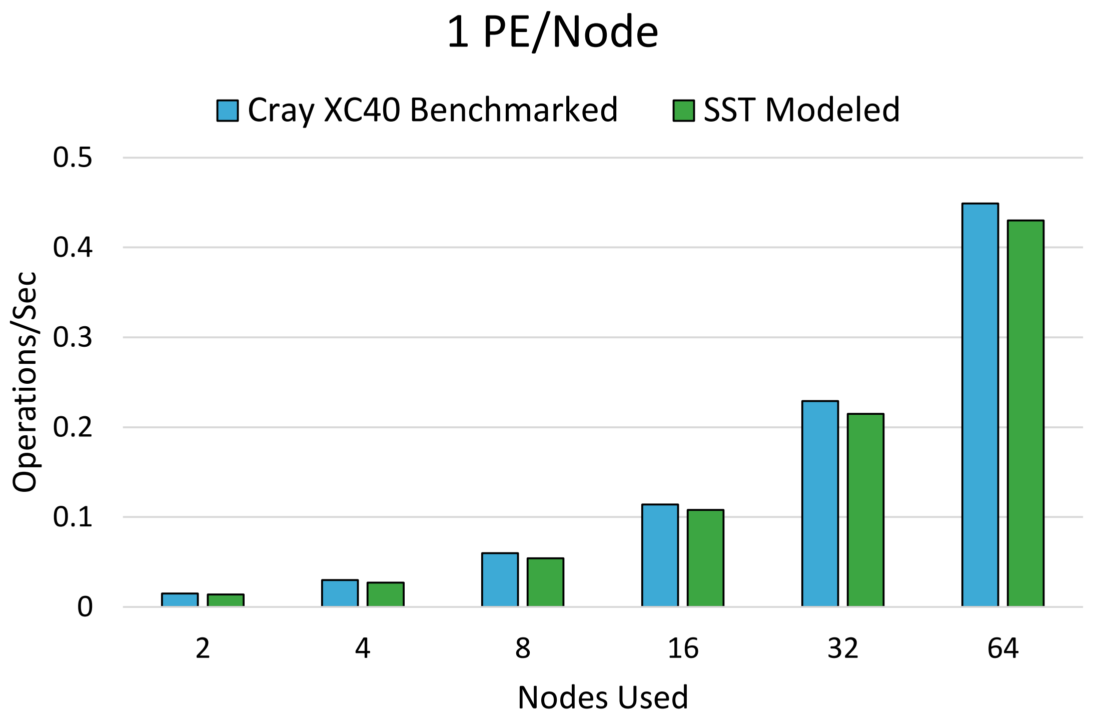
Compare predictions from SST to a Cray XC40 test machine at SNL, NM.

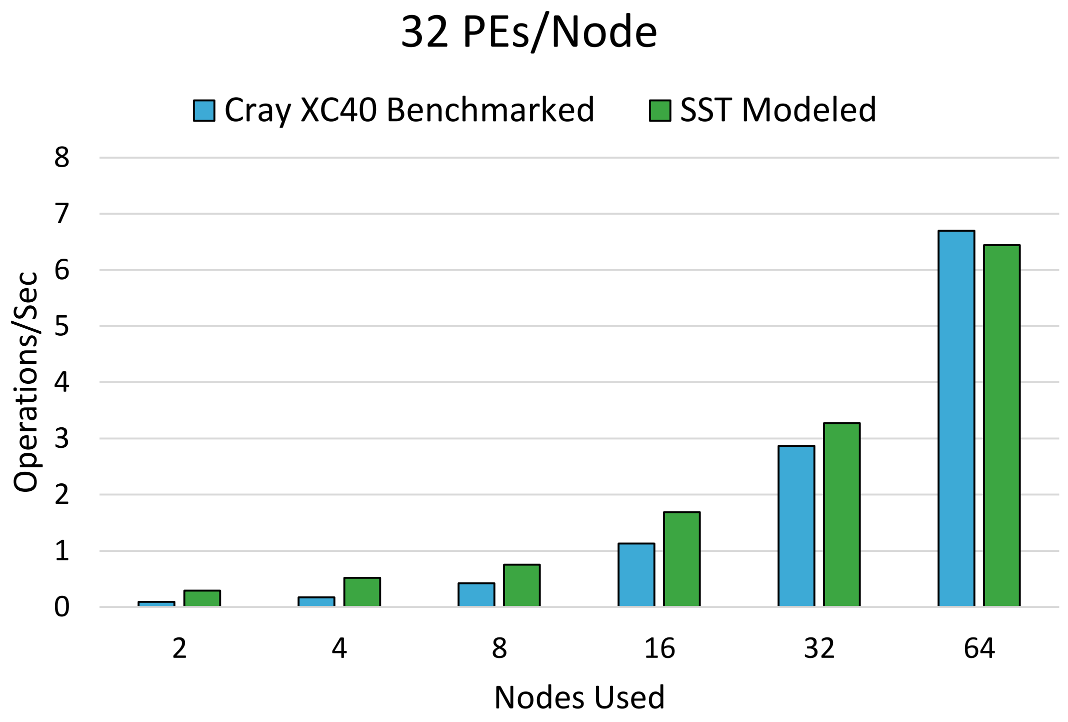
- Utilize dual-socket 16-core Haswell 2.3GHz Xeon processors
- Cray Aries XC40 Network
- Cray optimized SHMEM implementation
- Intel 17.4 Compiler Suite and Programming Environment
- ~100 dual-socket Haswell nodes (and 100 KNL which we do not use)
- Application Test System for the larger *Trinity* NNSA/ASC platform housed at Los Alamos National Laboratory, NM

SST Configuration:

- Merlin configured for Cray XC40 DragonFly network parameters
- Ember motif representations of each benchmark
- Memory timing taken from small micro-benchmarks run on the Cray system by performance engineering team
- See paper for detailed high-level model timing parameters

Validation of Basic PUT Operations

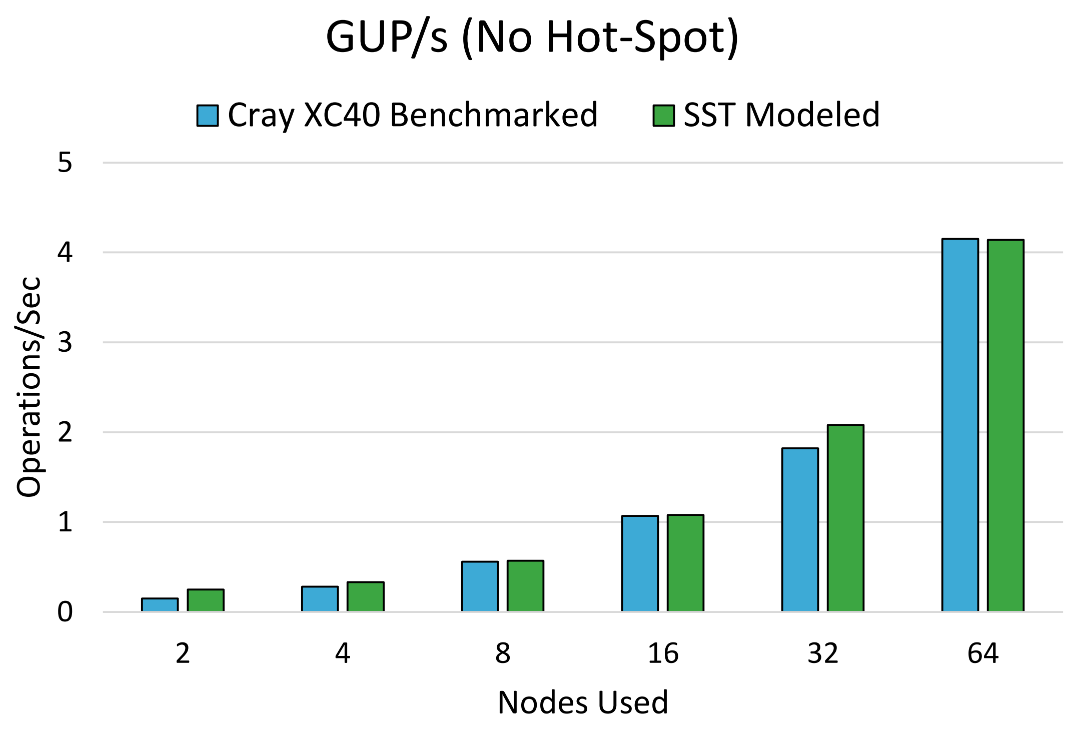


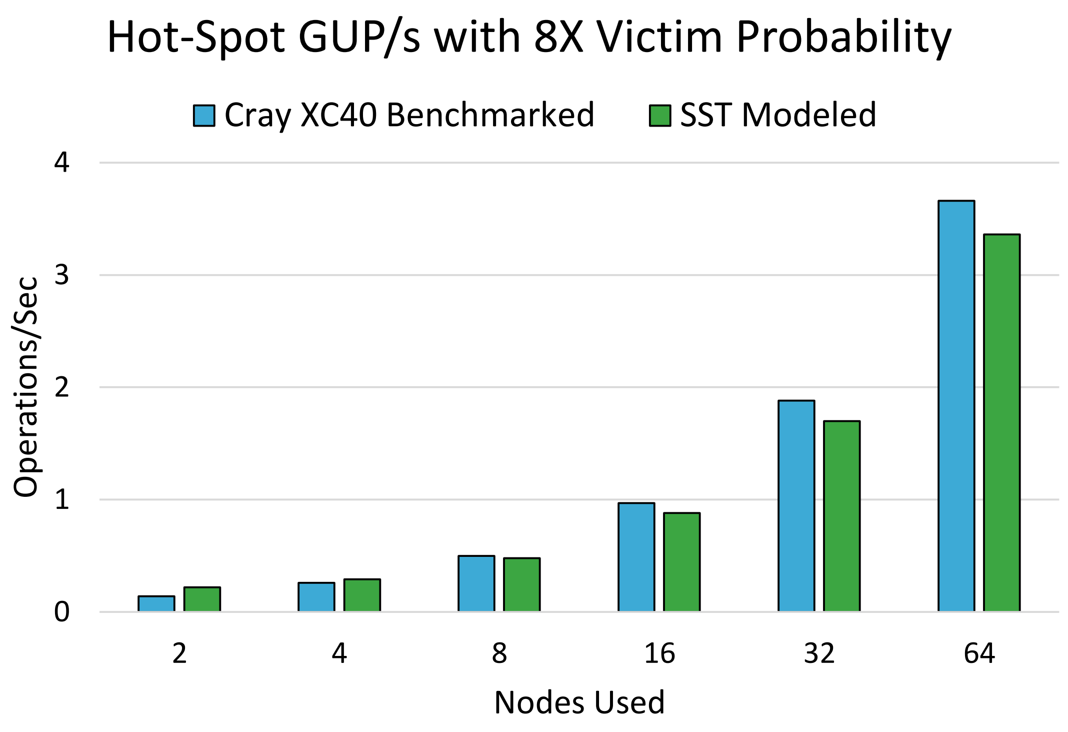


Compare Cray XC40 (Blue) to SST (Green) – strong qualitative trend. Predictive error is approx. 10% for 1 PE/node but up to 200% for very small node counts when using 32PE/node.

- Reflects difficulty in modeling high on-node and network contention for large PE/node counts
- Overall strong predictive trend as system size increases (more nodes are added)

Validation of GUP/s and Hot-Spot GUP/s Operations (32PE/Node)

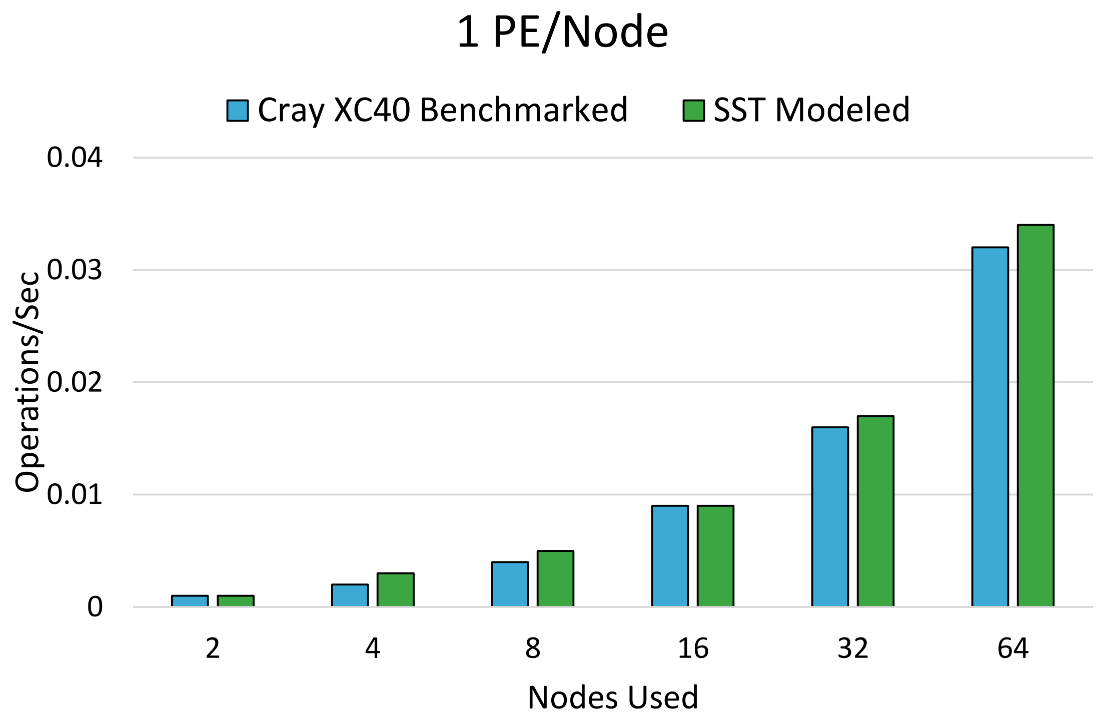


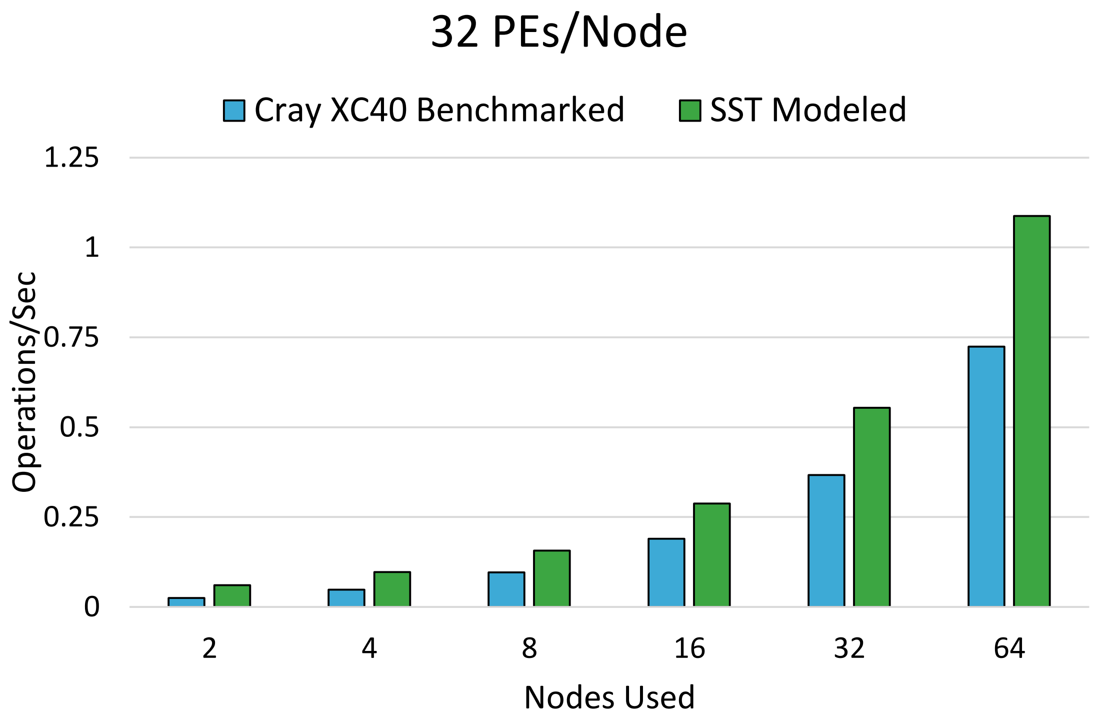


Compare Cray XC40 (Blue) to SST (Green), results are for 32 PEs per node. With no hot-spot, the GUP/s results show strong correlation with typically low (10-15%) predictive error. With an 8X increase in probability of selecting a victim node, predictive accuracy decreases at scale (still around 10-15%)

- Contention modeling is more challenging in victim node with higher number of PEs communicating with a single rank/node

Validation of FINC-GUP/s Operations

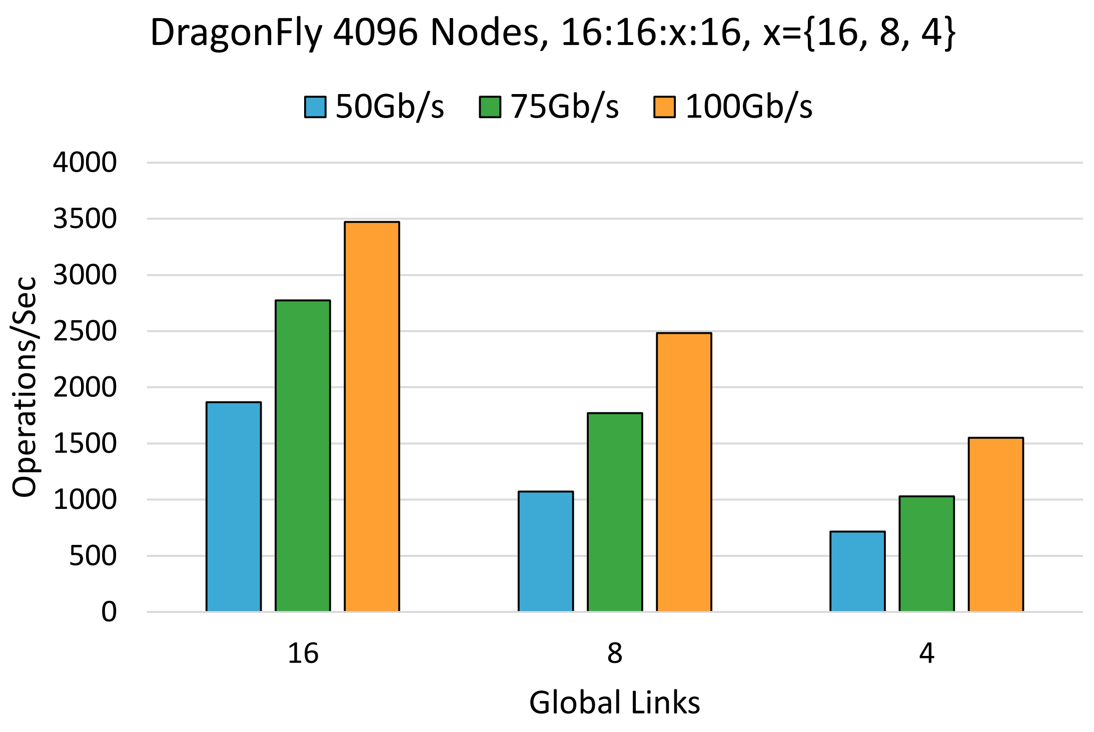


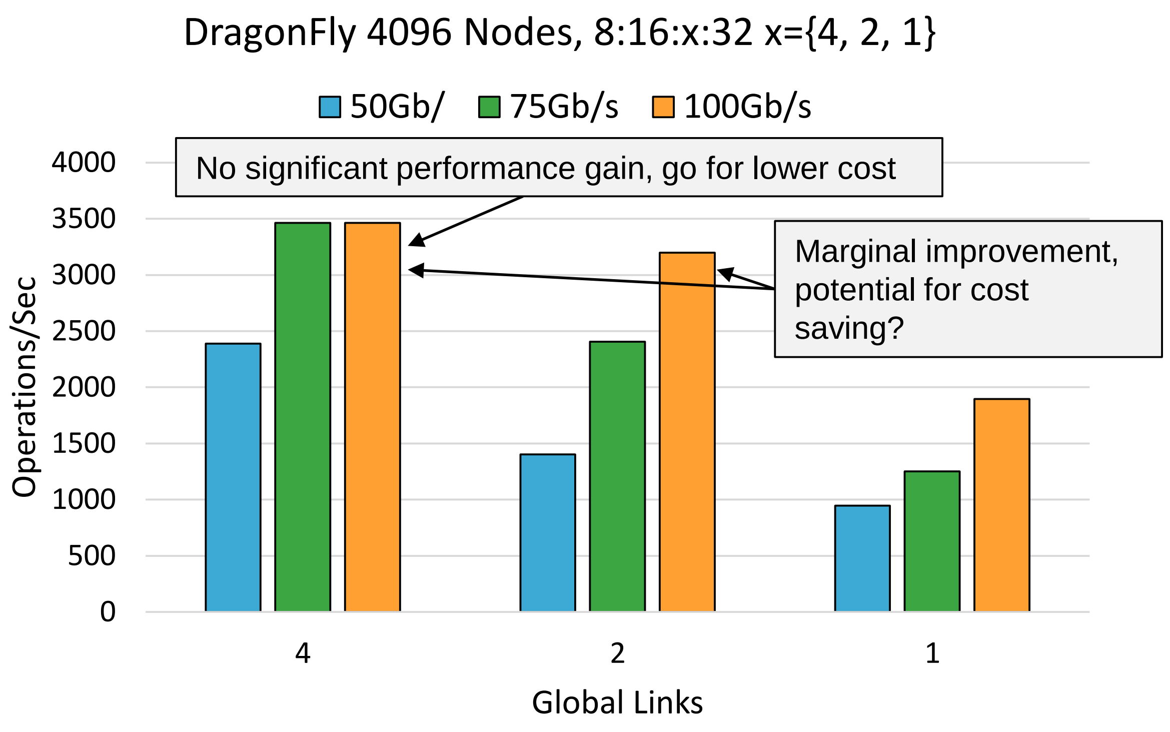


Compare Cray XC40 (Blue) to SST (Green) – models use of return path for atomics. Predictive accuracy decreases significantly over other models for 32 PEs per node to around ~50% error.

- Complex return paths are not well modeled yet and need to be improved (implies send side models are perhaps more accurate than receive side/response models)

Example – GUP/s Performance - What If for Network Design?





SST allows us to begin to perform What-If analyses on various network configurations and node designs

- 128 cores per node, 4096 nodes with different DragonFly topologies (~512K PEs), X Axis = Number of Global Links
- Graph on left is unlikely to be built, graph on right is more conventional design

Discussion and Conclusions

Discussion

OpenSHMEM is an interesting extension for our simulation and modeling capabilities

- Drives communication patterns which we are beginning to see as more common in our research activities
- Potential for these workloads to be important contributions to DOE scientific delivery in the future
- Implies that we (DOE and HPC community) need to understand performance issues

SST has historically been an MPI-focused tool reflecting DOE drivers

- Traditionally focused on critical path analysis around MPI tag/recv match timing in an offloaded NIC
- Heavily governs operation rates and other overheads act as second order

OpenSHMEM/fine-grained communication models have very different model characteristics

- Overheads may be small but add up when repeatedly performing many small operations (unlike MPI which is typically one larger monolithic operation)
- Pushes on models for caches, TLBs and on-node structure contention which are usually less important for MPI bulk data transfers

Required significant extensions and modifications to SST to support OpenSHMEM-like semantics

Conclusions

Work to develop OpenSHMEM support and higher fidelity models is not complete but is now well underway

- Overhaul of components and key performance pieces are now in-place
- Initial communication micro-benchmarks written

Predictive accuracy varies by node count and the number of PE/s used per node

- Higher density PE/node tends to drive higher contention and decrease model accuracy (10-15% error levels)
- Increase node counts push greater performance modeling into the network which SST has established models (tends to reduce model errors)

Our intention is to push OpenSHMEM models to full DOE system scale (e.g. >10K nodes)

- Argue that modeling large-scale OpenSHMEM patterns in SST should be reasonably accurate because of higher fidelity in network models that will be used at scale
- Will clearly vary by workload/primitive operations
- Micro-benchmarks used are somewhat worse-case because communication only and no compute/calculations are implemented

Sandia National Laboratories