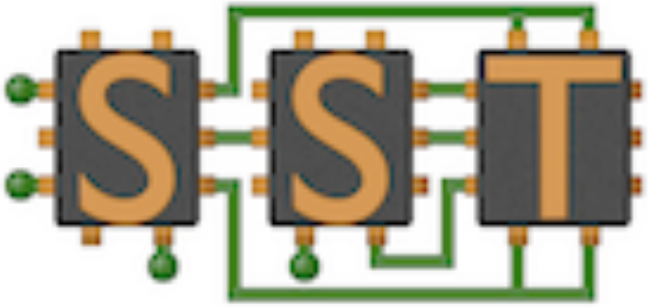
Exceptional service in the national interest

Sandia National Laboratories

Photos placed in horizontal position with even amount of white space between photos and header

Structural Simulation Toolkit (SST)



Modsim2017

SST Team

U.S. DEPARTMENT OF ENERGY

NNSA National Nuclear Security Administration

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. SAND NO. 2015-4701C

Why SST?

- Problem: Simulation is slow
  - Tradeoff between accuracy and time to simulate
  - Many simulators are serial, unable to simulate very large systems
- Problem: Lack of simulator flexibility
  - Tightly-coupled simulations: Difficult to modify
  - Difficult to simulate at different levels of accuracy

The Structural Simulation Toolkit:
A parallel, discrete-event simulation framework
for scalability and flexibility

What is SST?

Sandia National Laboratories

Goals

- Become the standard architectural simulation framework for HPC
- Be able to evaluate future systems on DOE/DOD workloads
- Use supercomputers to design supercomputers

Status

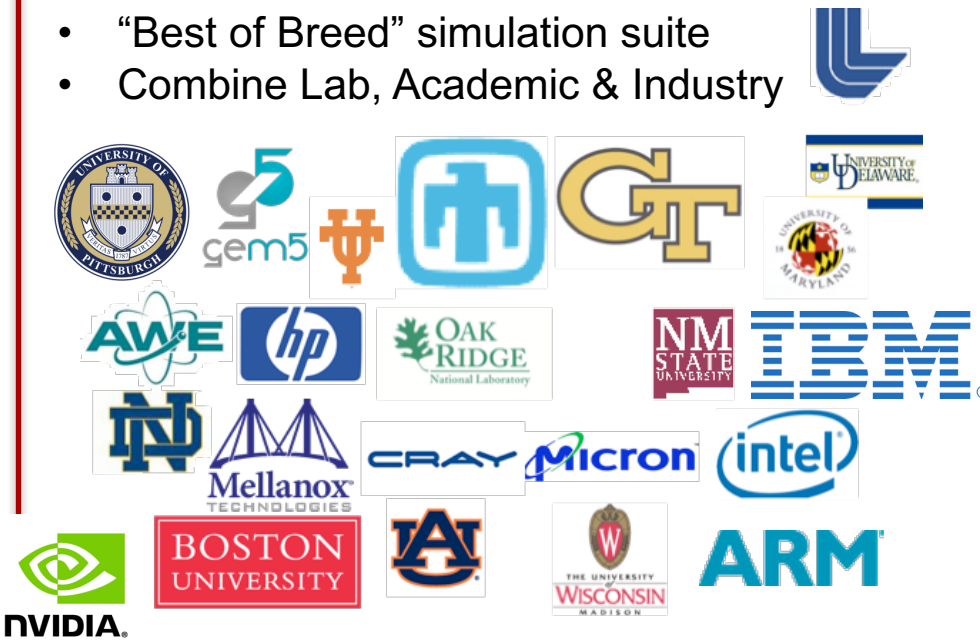
- Parallel Core, basic components
- Current Release (7.1)
  - Improved components
  - Modular core/elements
  - More Internal documentation

Technical Approach

- Parallel
  - Parallel Discrete Event core with conservative optimization over MPI/Threads
- Multiscale
  - Detailed and simple models for processor, network, & memory
- Interoperability
  - DRAMSim, ,memory models
  - routers, NICs, schedulers
- Open
  - Open Core, non-viral, modular

Consortium

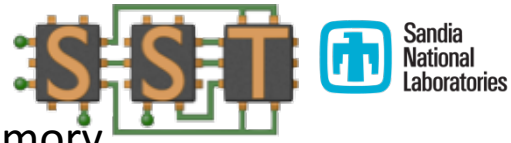
- "Best of Breed" simulation suite
- Combine Lab, Academic & Industry



Key Capabilities

- **Parallel**
  - Built from the ground up to be scalable
  - Demonstrated scaling to 512+ host processors
  - Conservative, Distance-based Optimization
  - MPI + Threads
- **Flexible**
  - Enables “mix and match” of simulation components
  - Custom architectures
  - Multiscale tradeoff between accuracy and simulation time
    - E.g., cycle-accurate network with trace-driven endpoints
- **Open API**
  - Easily extensible with new models
  - Modular framework
  - Open-source core

Existing SST Element Libraries



Detailed Memory Models

- memHierarchy - Cache and Memory
- cassini - Cache prefetchers
- DRAMSim2 - DDR
- NVDIMMSim - Emerging Memories
- Goblin - HMC

Dynamic Trace-based Processor Model

- ariel - PIN-based Tracing

Cycle-based Processor Model

- MacSim - GPGPU
- m5C - Gem5 integration layer

High-level Program Communication Models

- ember - State-machine Message generation
- firefly - Communication Protocols
- hermes - MPI-like interface

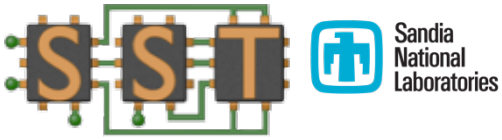
Cycle-based Network Model

- merlin - Network router model and NIC

High-level System Workflow Model

- scheduler - Job-scheduler simulation models

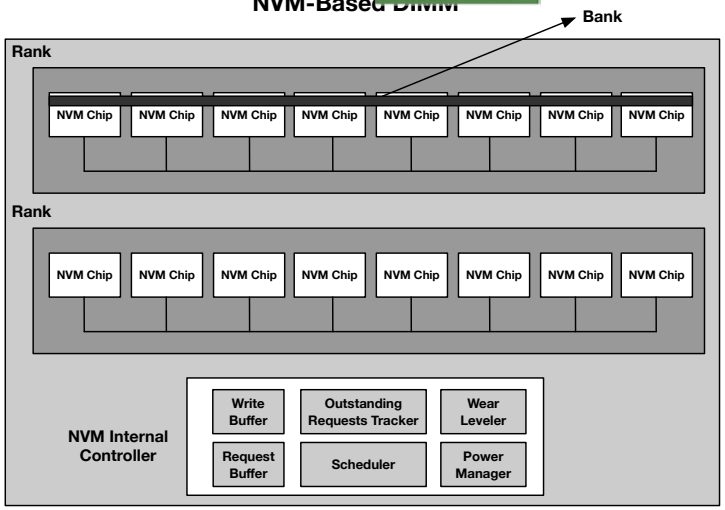
New Additions to SST: Core

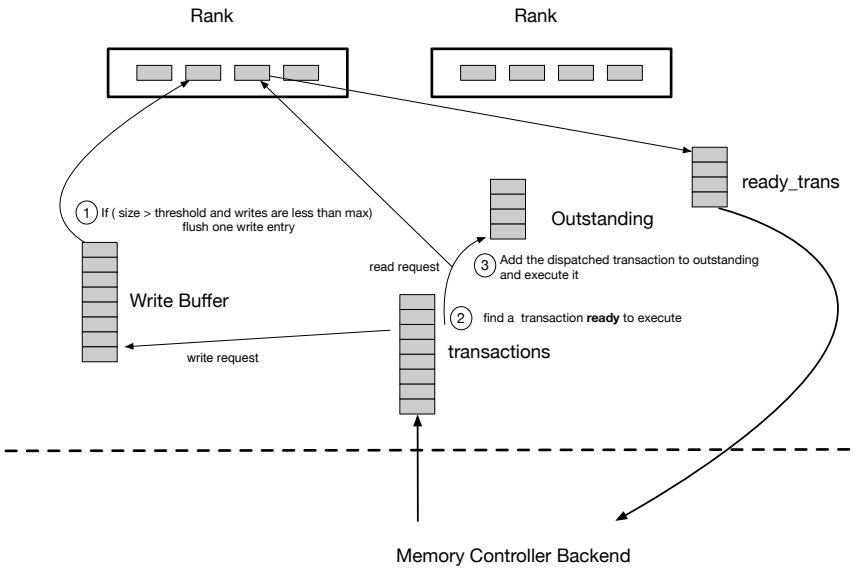


- Overhaul of the SST element information description system
- Improved external component support (much easier to support than external development community)
- HDF5 & JSON support for statistics output
- Improved thread scaling
- Easier Builds (removed Boost Dependency)
- Early support for running SST on IBM POWER
- Implementation of a “stop at” wall time feature for working within scheduled cluster environments
- Support for describing co-ordinates of components in Python configuration (for visualizations)

New Additions : Messier

- Messier: NV Memory model
- Focus on NV-DIMMs e.g.:
  - # Banks, Latencies
  - Row buffers, write buffers
  - policies, outstanding requests, ordering
  - Address mapping

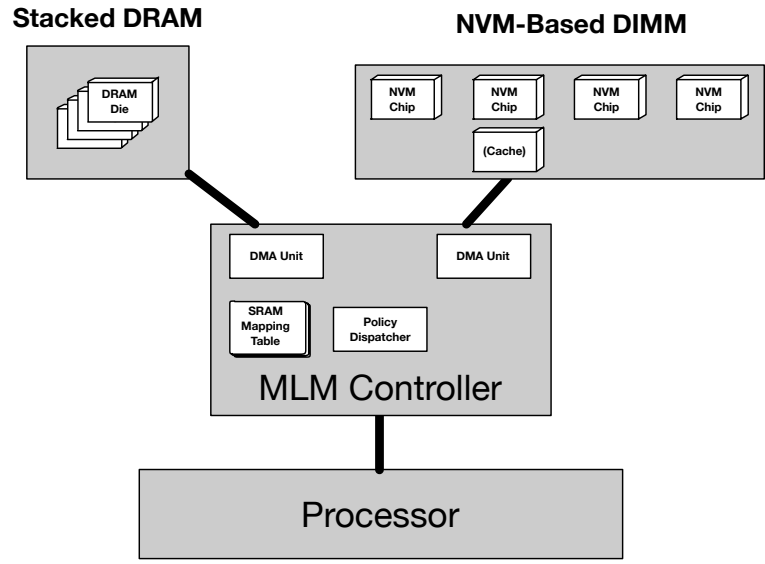




- Report: SAND2017-1830

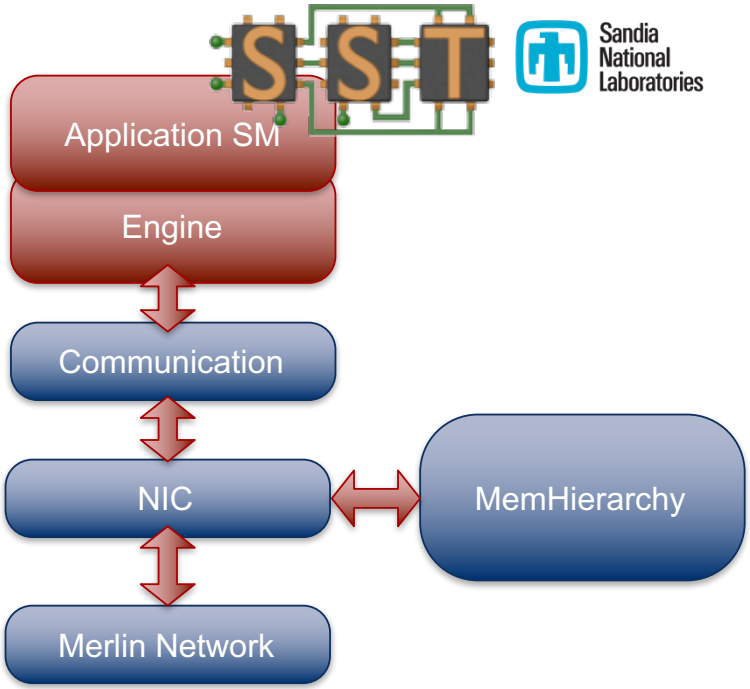
New Additions : Memory

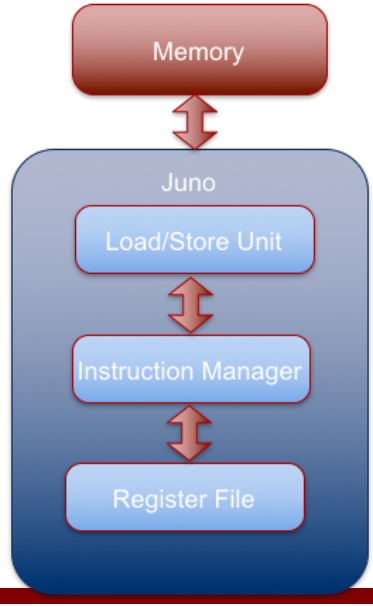
- Improvements to IBM CramSim for enable threaded simulations
- Improved multi-level memory models
- Performance and scaling improvements (event-driven (clock-less) memory models)
- Scratchpad support
- New TLB model



New Additions to SST

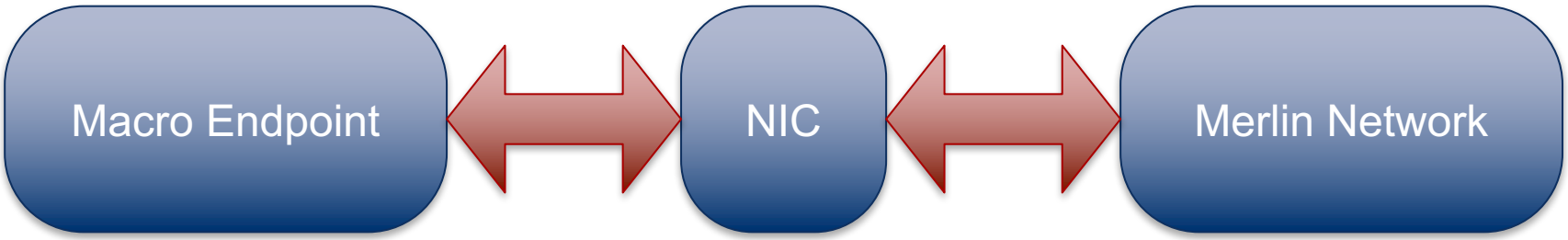
- Support for memory modeling in large-scale network analysis
- Early support for some SHMEM based communication models (in progress)
- Juno Processor Model
  - Simplified processor model
  - Designed for extensibility
  - Uses: Tutorial, Correctness checking





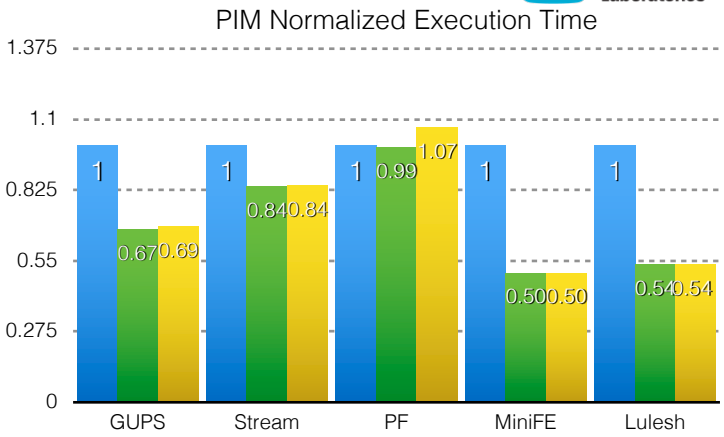
New Additions SST/Macro

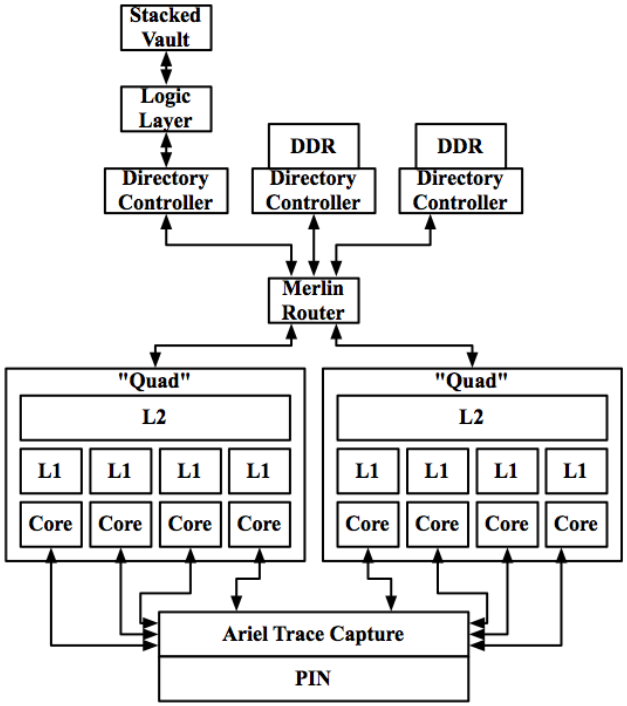
- Macro / Merlin Integration
- Beta release of OTF2 trace replay skeleton
- Beta release of Clang-based auto-skeletonization source-to-source tools
- Integrated job launcher components for simulating PBS or SLURM-like batch systems

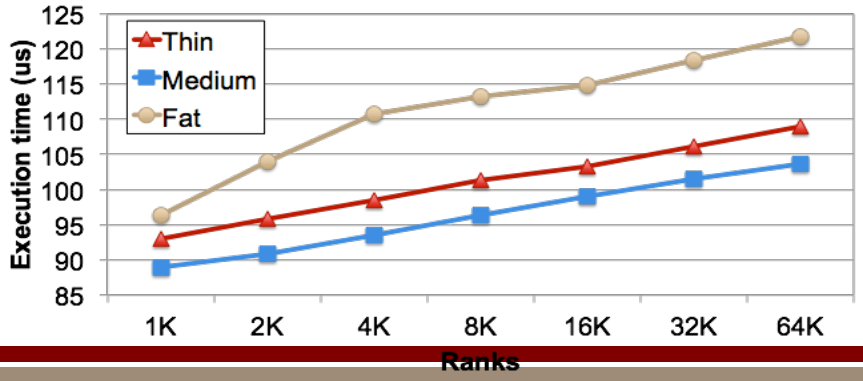


Use Cases

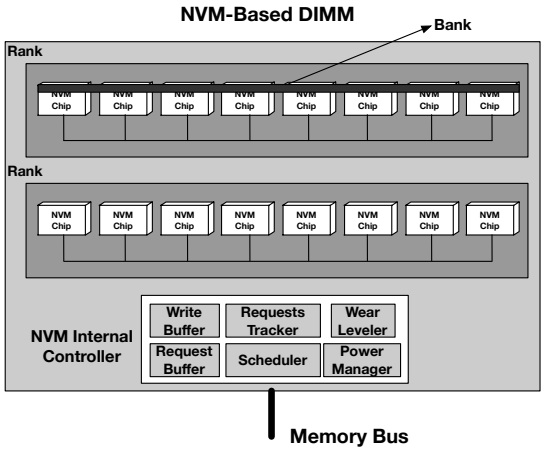
- Processing-in-memory
- Multi-Level Memory
  - HW Tradeoffs: capacity ratios,
  - SW Tradeoffs: application, runtime, OS, HW control
- Scalable Network Studies
  - Network on Chip
  - Coherent system interconnect NIC
- Scheduling



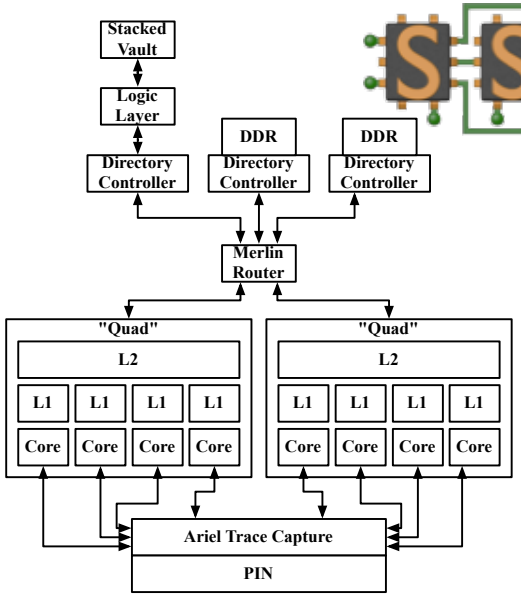




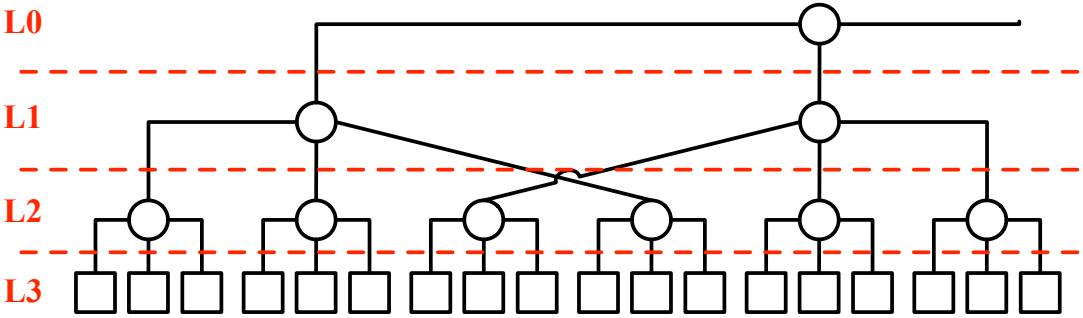
More SST Use Cases



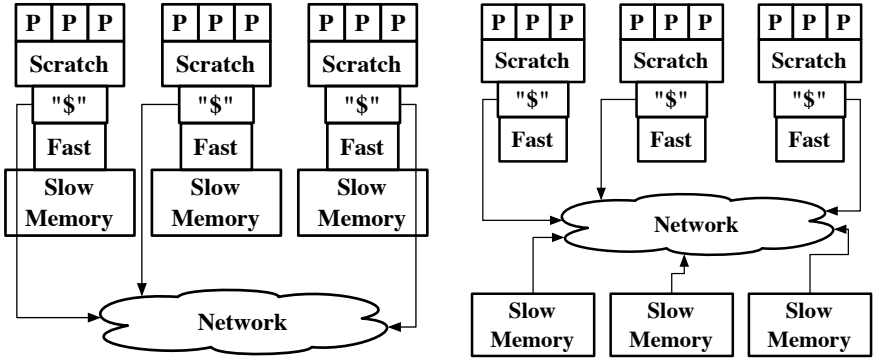
Emerging NV Memory Technologies



Multi-Level Memory (HBM+DDR+NV)



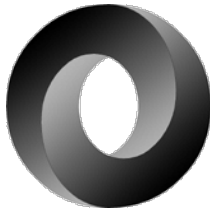
Photonic Network Topology & Routing



Dissaggregated Memory

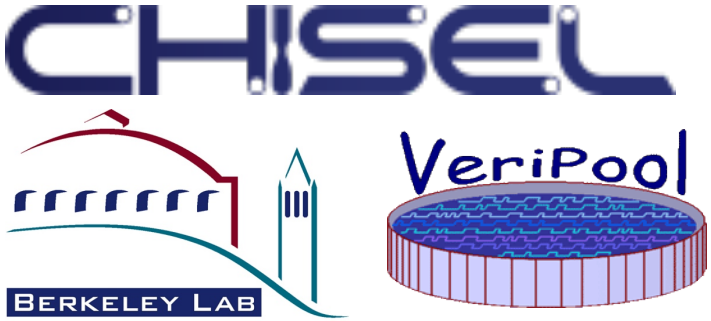
OCCAM & SST

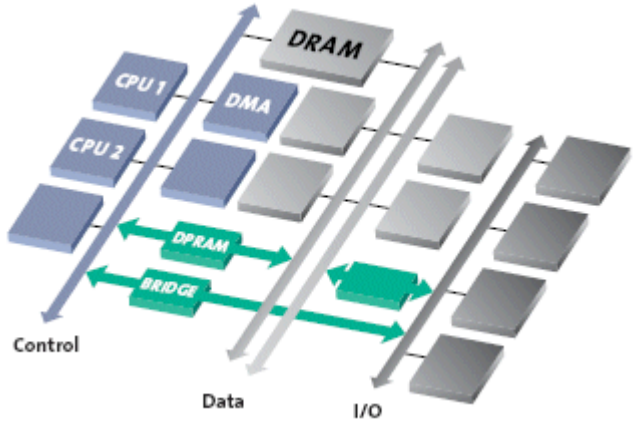
- Occam
  - Provides a framework for SST
- Configuration
- Experiment Organization
- Output Visualization
- Curation
- Sharing Results



- SST
  - Provides **a** simulator for Occam

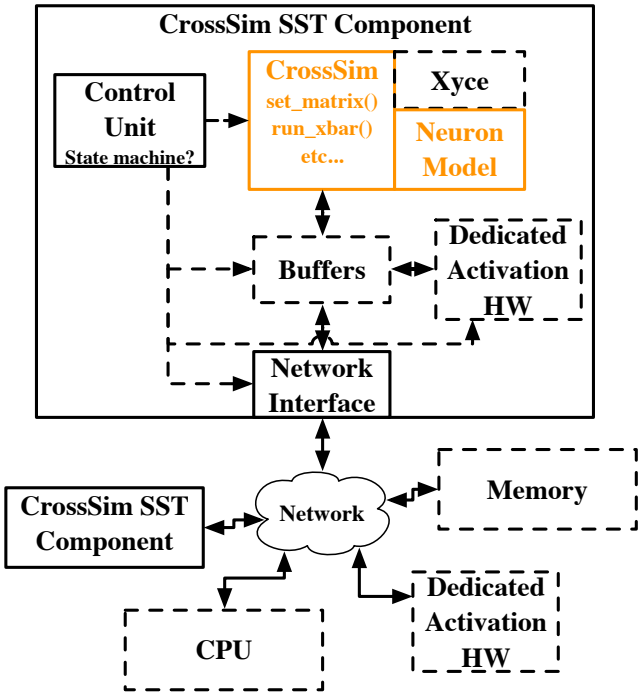
Future Directions

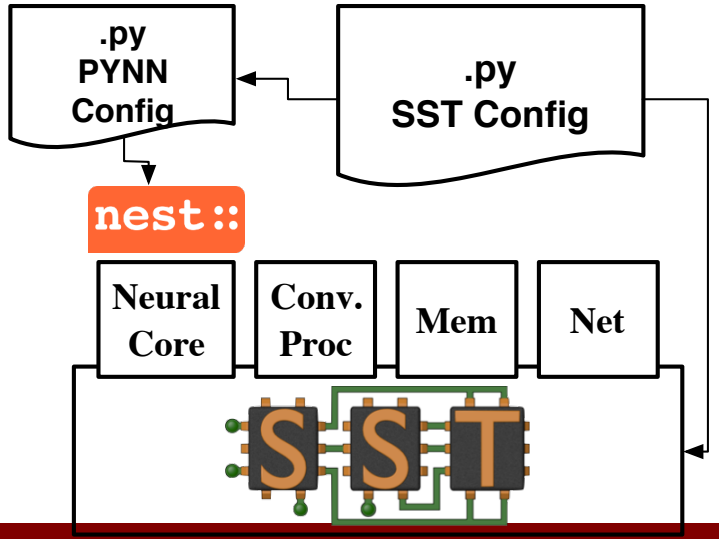




- HDL Simulation via Verilator & Chisel
  - Low-level hardware design
  - Path to tape-out (Chisel)
- New Processor Models
  - RISC V
  - Juno
- Improved NoC Models
  - Faster Performance
  - NoC QoS
  - Optical Circuit Routing

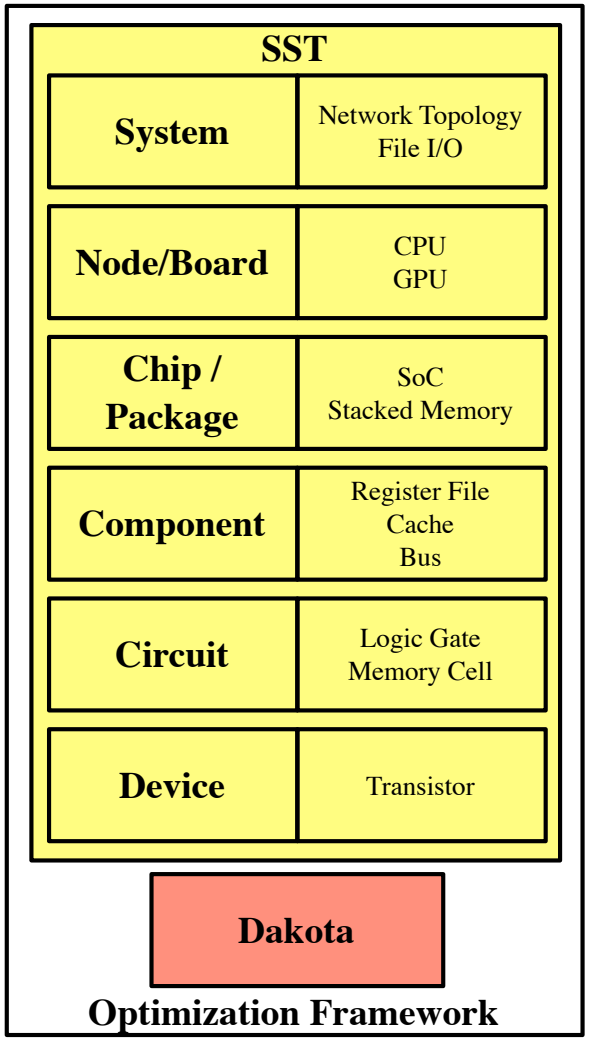
Future Directions: Neural Inspired





- Custom processors / accelerators
- Generic models (e.g. NEST, N2A)
- Explore
  - System Integration Issues
  - Architectural Bottlenecks
  - Programmability
- Allows…
  - Exploration of conventional / Neural interface (e.g. different message routing protocols)
  - Allows Neural “cores” to connect to conventional processors, network, memory, etc… (explore ’speeds and feeds’)
  - Scalability: SST can utilize thread- and MPI-level parallelism

Future Directions: More Framework Integration



- Beyond Moore Computing
  - New Architectures (e.g. Neuromorphic)
  - New Devices (e.g. Memristor)
  - New Programming Models / Algorithms
- Requires Cross-Stack Optimization
  - Device to System Level
  - Use of Dakota / INDRA to automate design space exploration
- Requires Inter-disciplinary approach
  - SST Simulator as “Clearinghouse of Ideas”
  - Common language of exploration

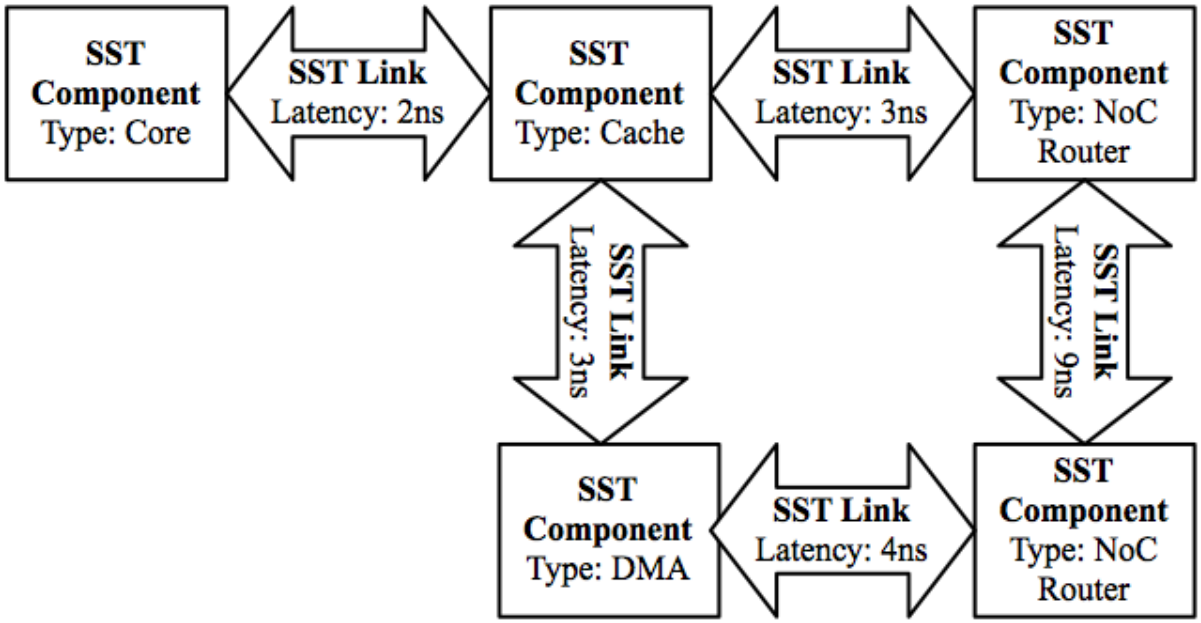
Conclusion

- SST is a Parallel, Flexible, Open architectural simulator
- Large library of Memory, Processor, Network, and other models
- 7.1 Release
  - Usability enhancements in the Core
  - New Memory, Network, Processor models
- Future SST
  - More & better components (RISC-V, Network, etc…)
  - Chisel, Occam integration
  - Beyond Moore Framework
  - **<Your Use Case Here>**

HOW CAN SST WORK FOR YOU?

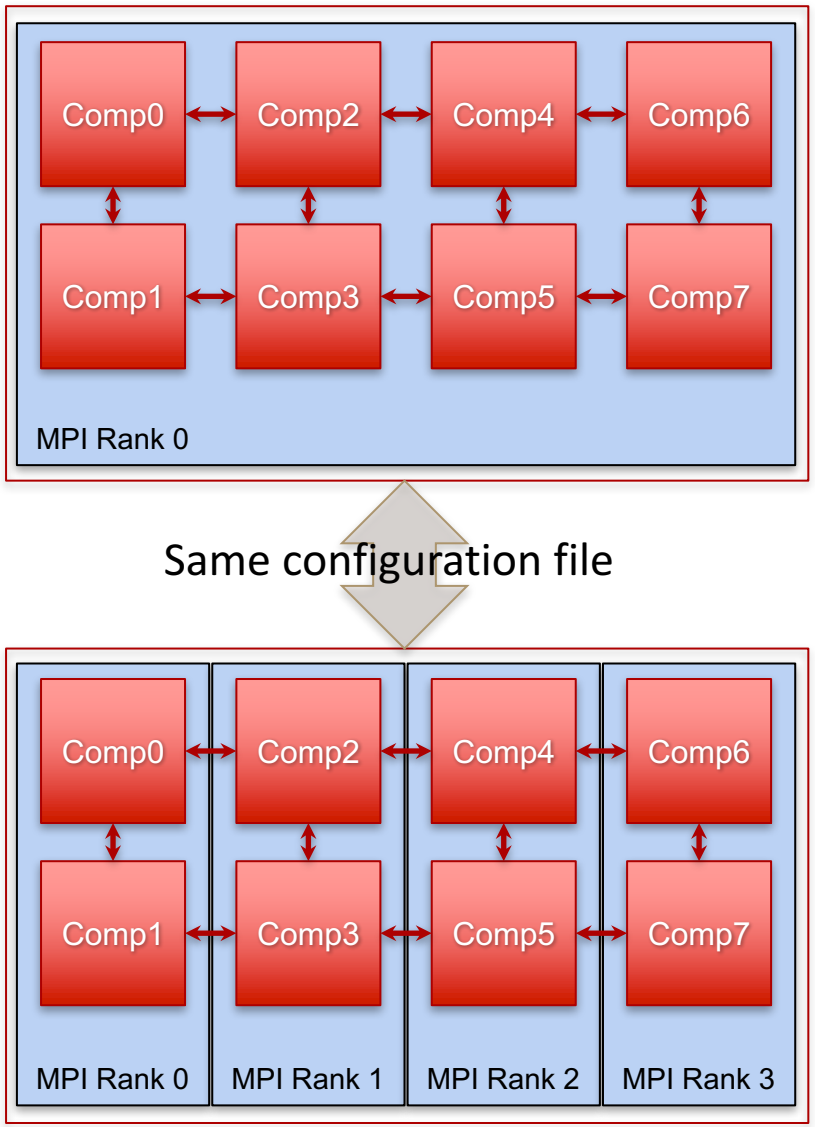
SST Discrete Event Algorithm

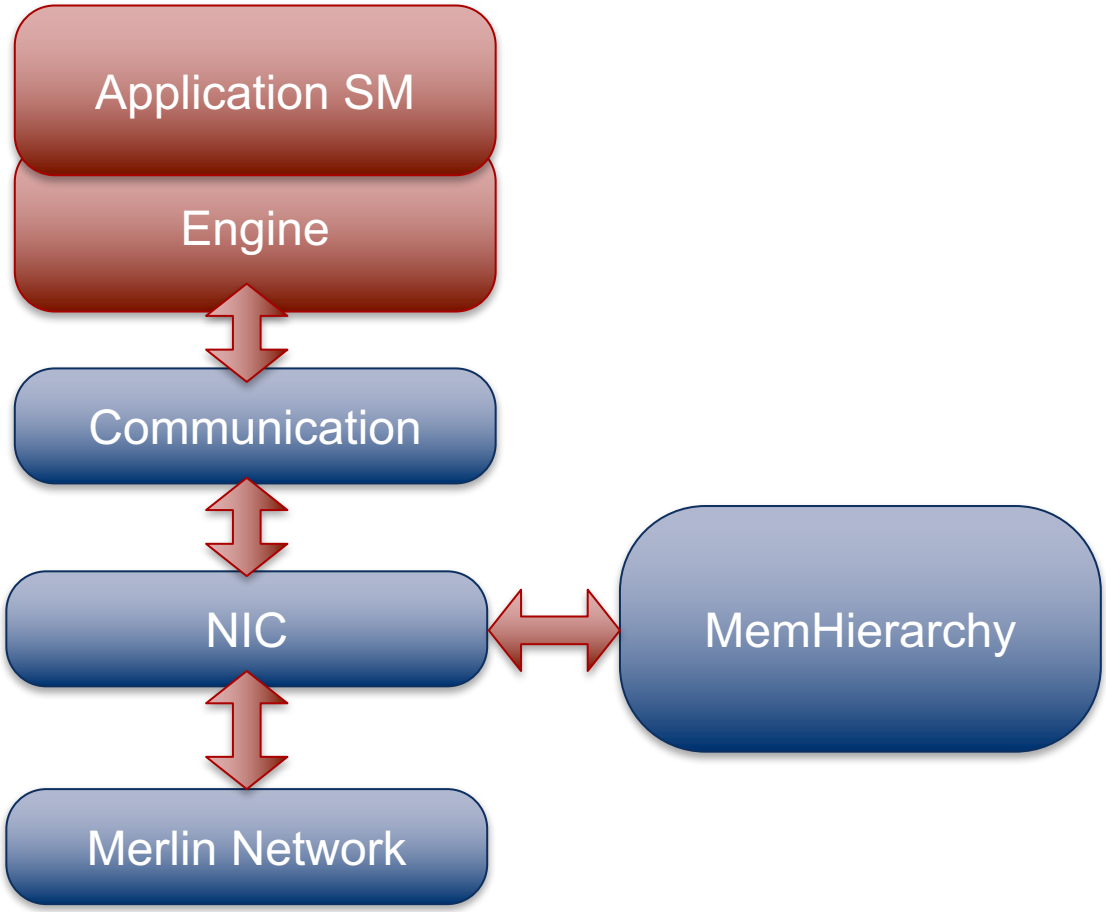
- SST Simulation are comprised of components connected by links
- Components interact by ending events over links
- Each link has a minimum latency (specified in SI time units)



SST in parallel

- SST was designed from the ground up to enable scalable, parallel simulations
- Components are distributed among MPI ranks & threads
- Links allow parallelism
  - Hence, components should communicate via links only
  - Transparently handle any MPI & Inter-thread communication
  - Specified link-latency determines MPI synchronization rate




Application SM
Engine
Communication
NIC
MemHierarchy
Merlin Network