The UHPC X-Caliber Project
Architecture, Design Space, & Codesign

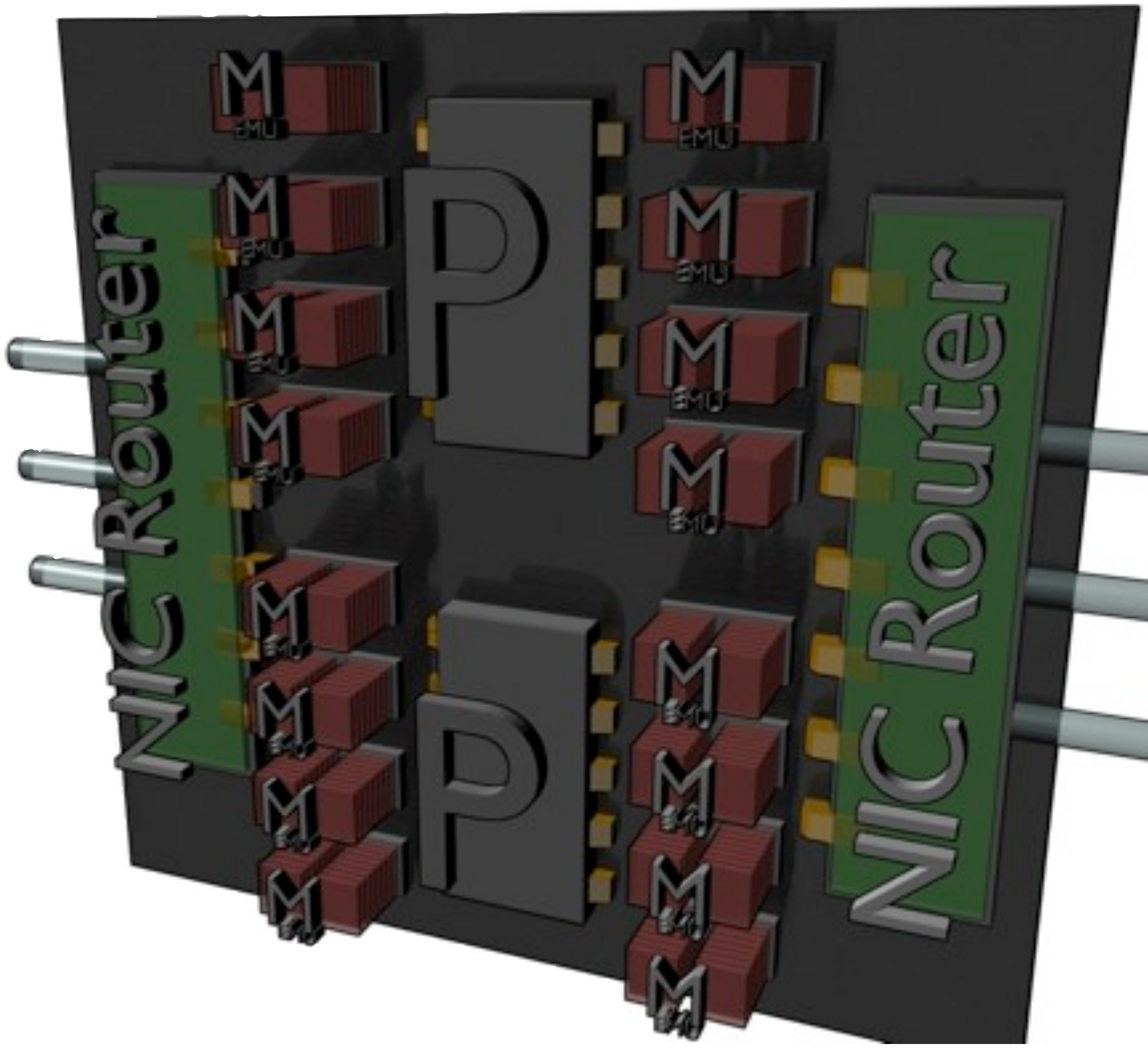
Arun Rodrigues

Sandia National Labs

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company,
for the United States Department of Energy's National Nuclear Security Administration
under contract DE-AC04-94AL85000.

DARPA UHPC Project

- Goal: Prototype Rack
  - 1 PetaFlop in 2018
  - 57 KW (inc. cooling & IO)
- Sandia led team with Micron, LexisNexis, 8 academic partners
- Major innovation required in packaging, network, memory, apps, and architecture
- Major Themes
  - Parallax programming Model
  - Stacked memory
  - Memory Processor
  - Optical Network
  - Codesign
- Codesign
  - Application driven
  - Iterative
  - Common simulation platform (SST) as "clearing house" for ideas



Exascale Design Study

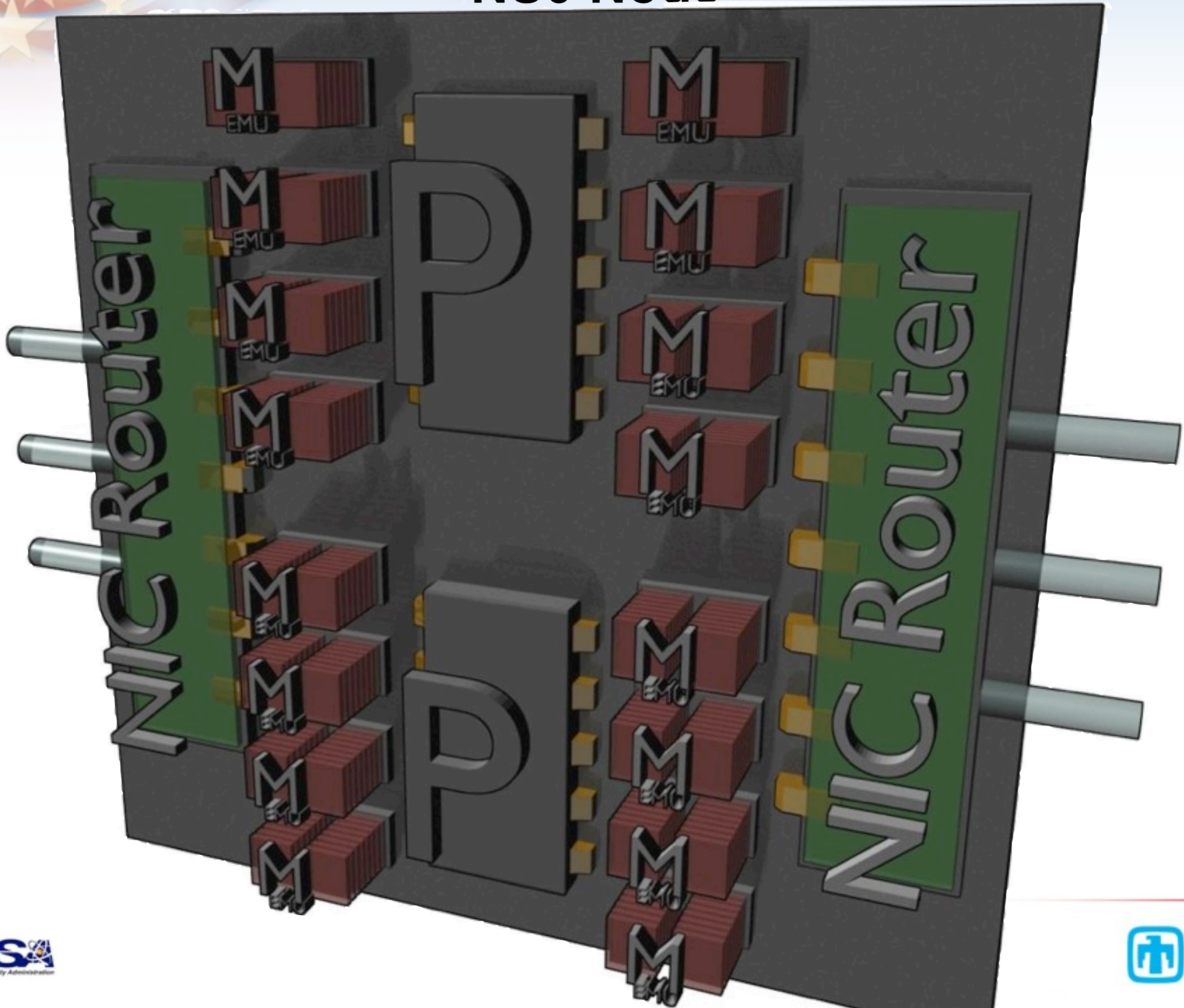
- 2018 Exascale Machine
  - 1 Exaop/sec
  - 500 petabyte/sec memory bandwidth
  - 500 petabyte/sec interconnect bandwidth
  - Assume advanced packaging may, or may not be available

- Consider power
  - 1 pJ * 1 Exa = 1 MW
  - 1 MW/year = $1 M
  - $100-200M / year power bill infeasible
- But, that's OK
  - Reliability, programmability, component cost, scalability, get you first

	Energy	Events/sec	Conventional	Adv. Pack	Improved
Processor	62.5 pJ/op	1E+18 ops	62.5 MW	40%	37.5 MW
Memory	31.25 pJ/bit	4E+18 bits	125 MW	45%	68.8 MW
Interconnect	6 pJ/bit	4E+18 bits	24 MW	44%	13.4 MW
Total			211.5 MW		119.7 MW

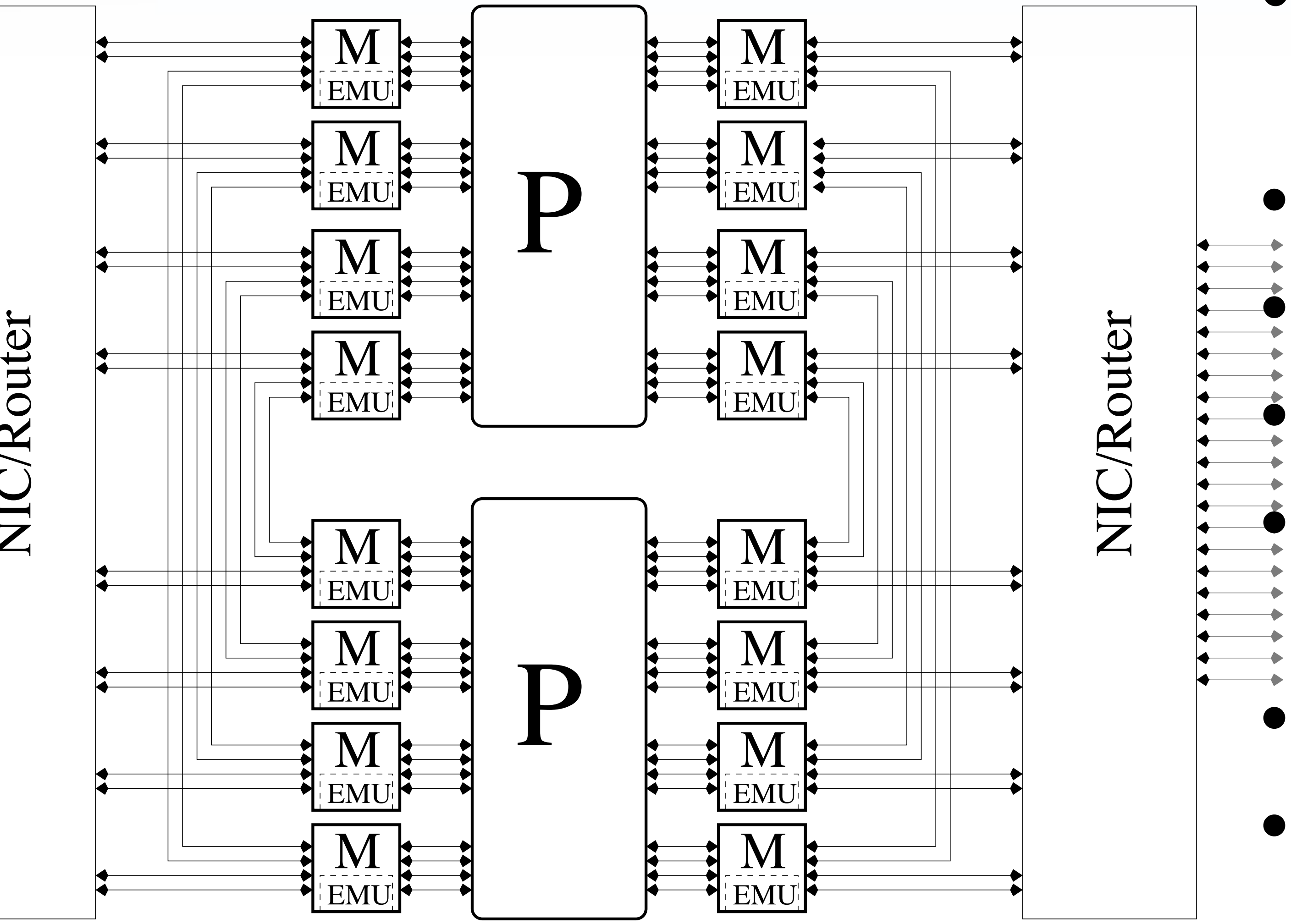
X-Caliber Architecture

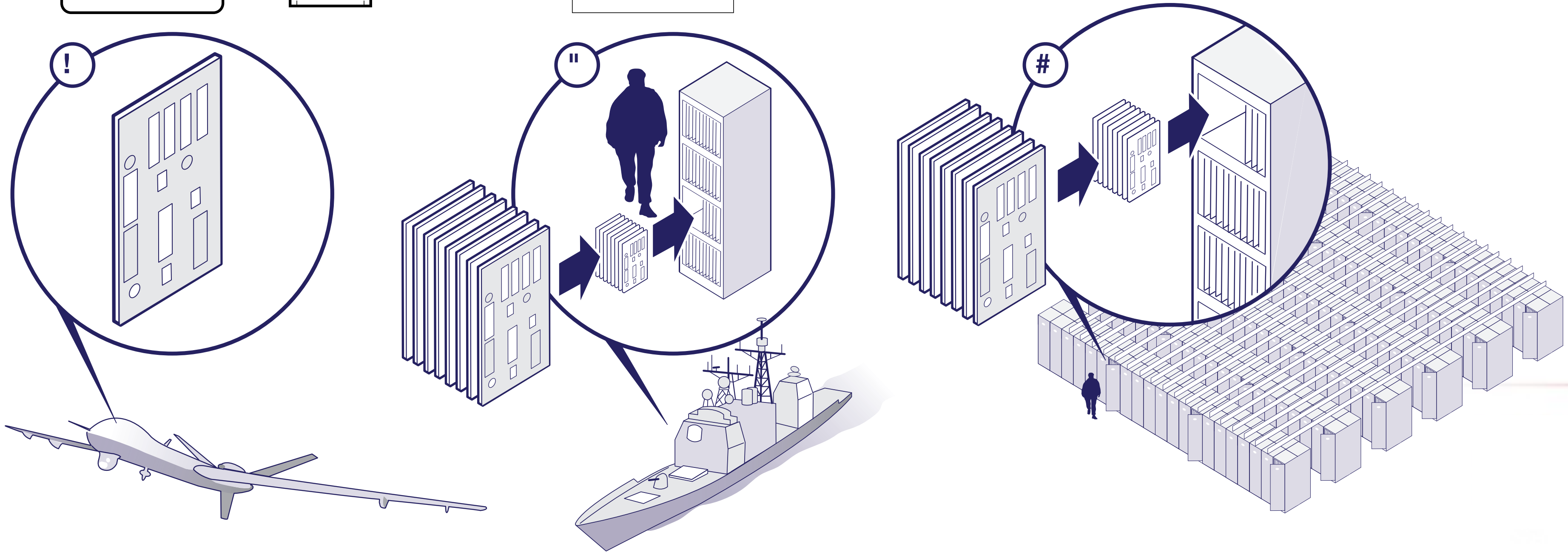
NS0 Node



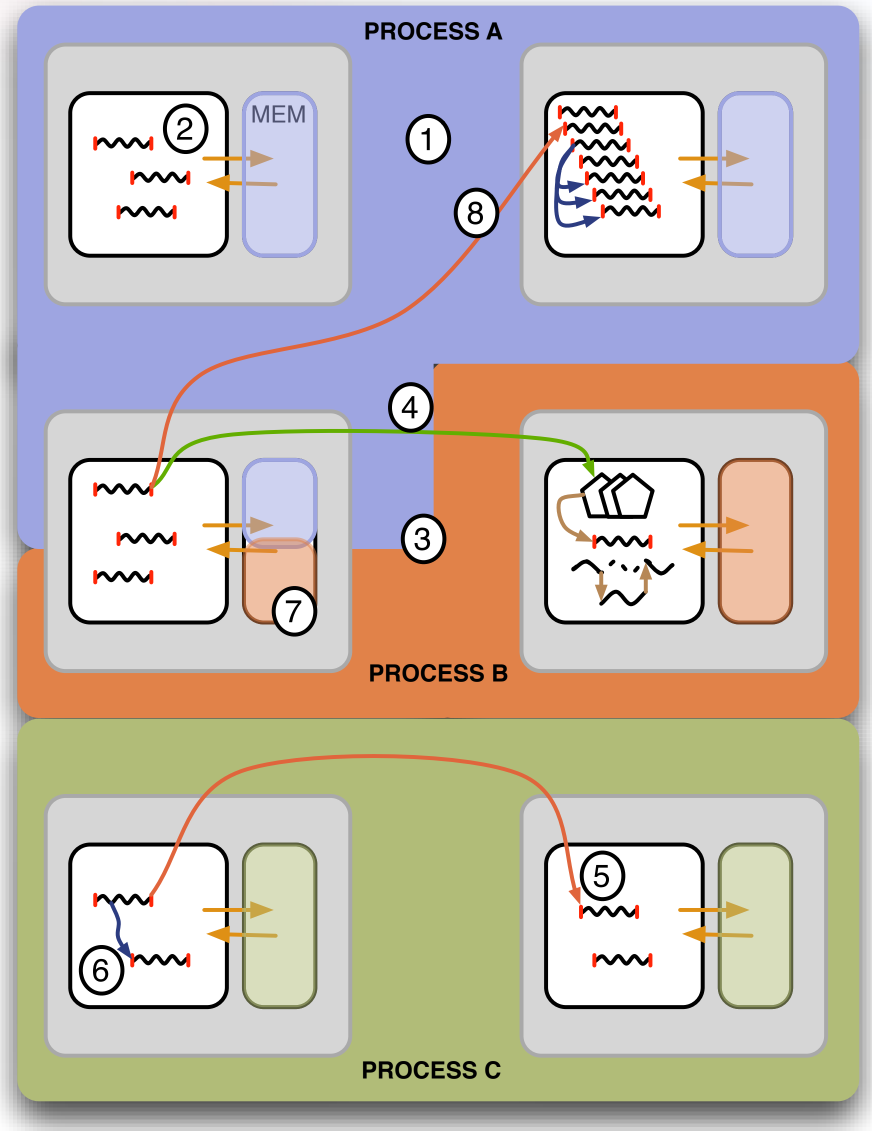
Architecture

- Movement = power
- Reduce amount & cost of data movement
- Stacked memory
- Memory Processor
- Optical Network
- Heavyweight multithreaded vector processor
- Sprint Modes
- Node building block





Parallax Execution Model



① Process spans multiple nodes

② Multiple user-level threads per node

③ Processes can share same nodes

④ Light weight synchronization through Local Control Objects

⑤ Message driven computation creates remote threads with parcels

⑥ Threads can spawn local threads

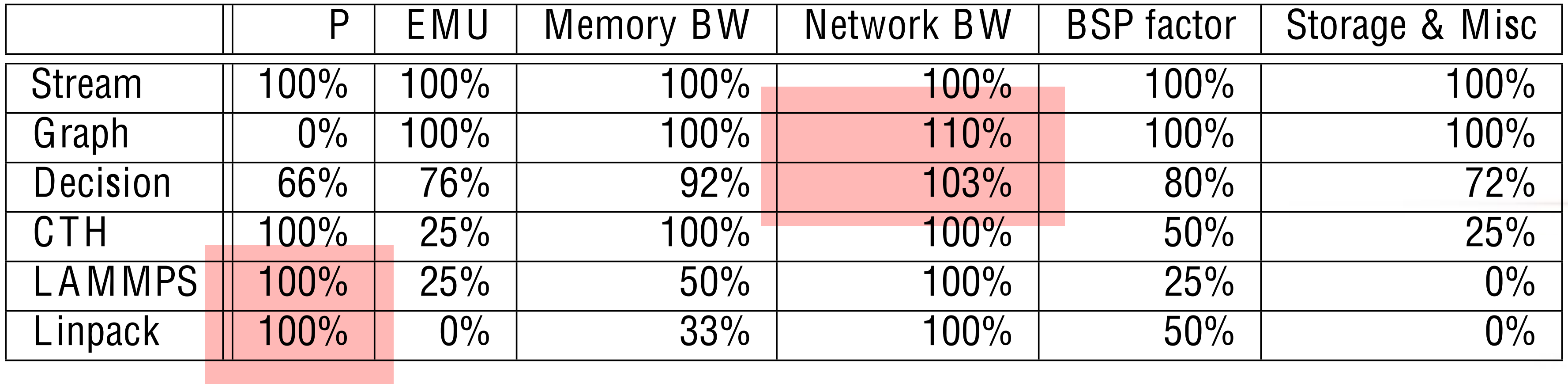
⑦ Distributed shared Memory

⑧ Global address space enables direct remote memory access

- Highly threaded
  - –Threads as addressable objects to enable introspection
  - –Low cost thread spawn
- Light weight Synchronization
  - –Hardware-assisted sync
  - –Enables dataflow computation
- Message driven computation
  - –Remote thread instantiation with parcels
- Highly Dynamic
  - –Thread migration, object migration
  - –Global namespace to coordinate object movement

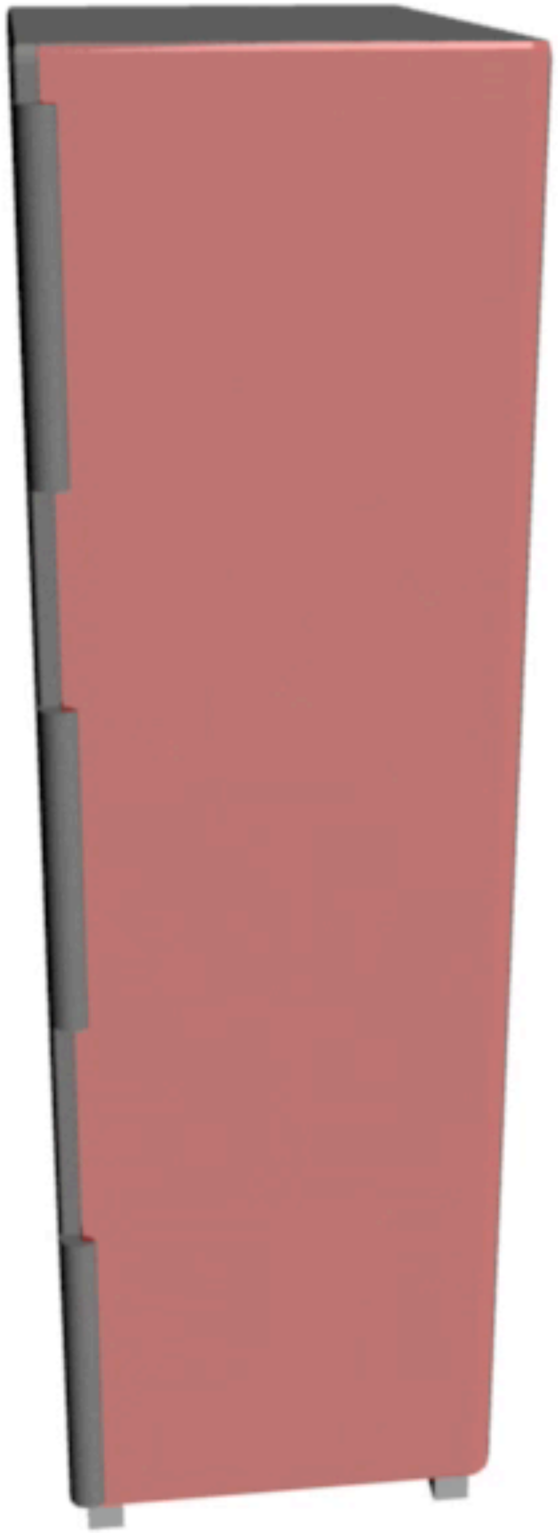
Sprint Modes

- **Processor**
  - –Normally 1.5GHz, can sprint to 2.5 GHz
    - All cores sprint for limited period of time
    - Some cores sprint indefinitely
  - –Useful for Amdahl regions of code
- **Network**
  - –512 GB/s normal injection BW, can sprint to 1024 GB/s
  - –Can activate additional links to increase BW
  - –Useful for smoothing bursty communication
- **Memory**
  - –Ability to boost clock of processing components
  - –Perform more processing in memory



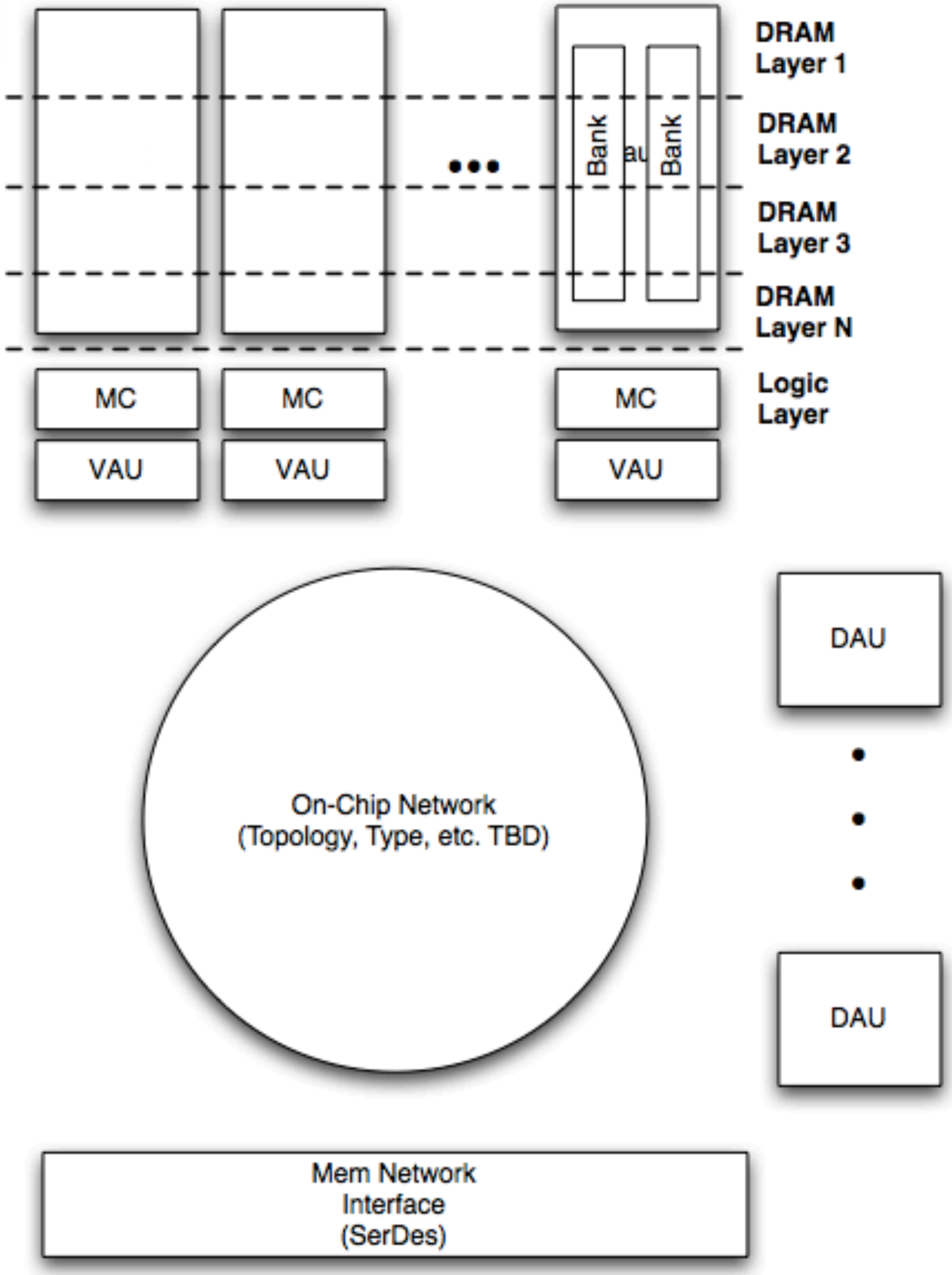
	P	EMU	Memory BW	Network BW	BSP factor	Storage & Misc
Stream	100%	100%	100%	100%	100%	100%
Graph	0%	100%	100%	110%	100%	100%
Decision	66%	76%	92%	103%	80%	72%
CTH	100%	25%	100%	100%	50%	25%
LAMMPS	100%	25%	50%	100%	25%	0%
Linpack	100%	0%	33%	100%	50%	0%

The Embedded Memory Processor



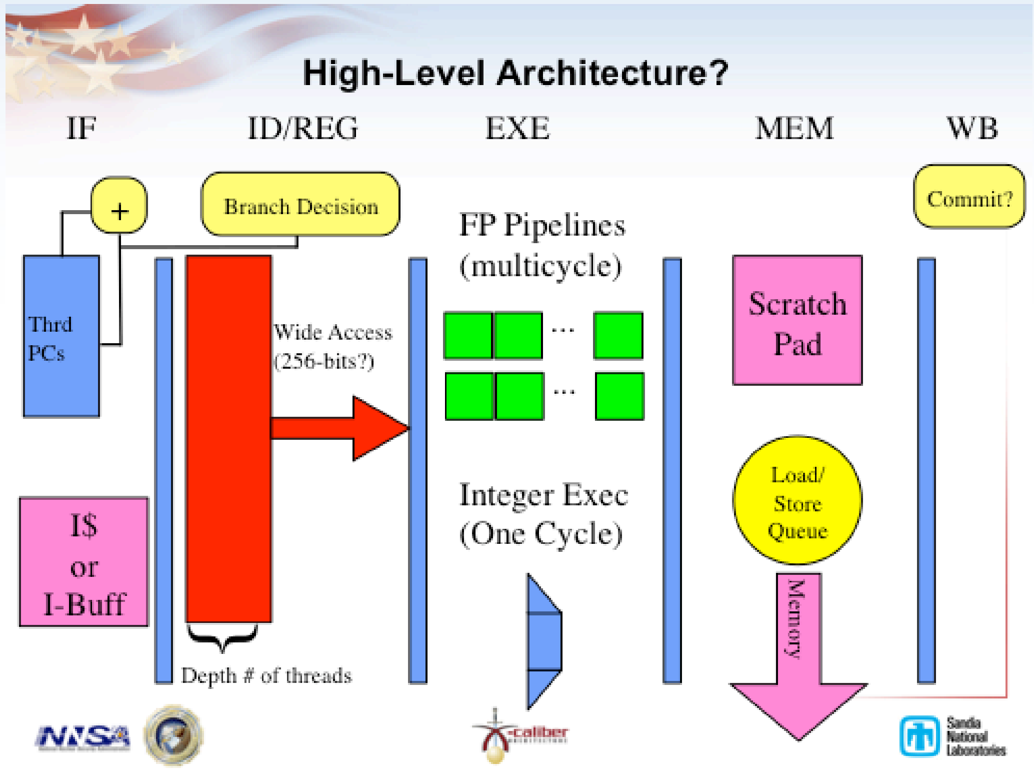
The EMP

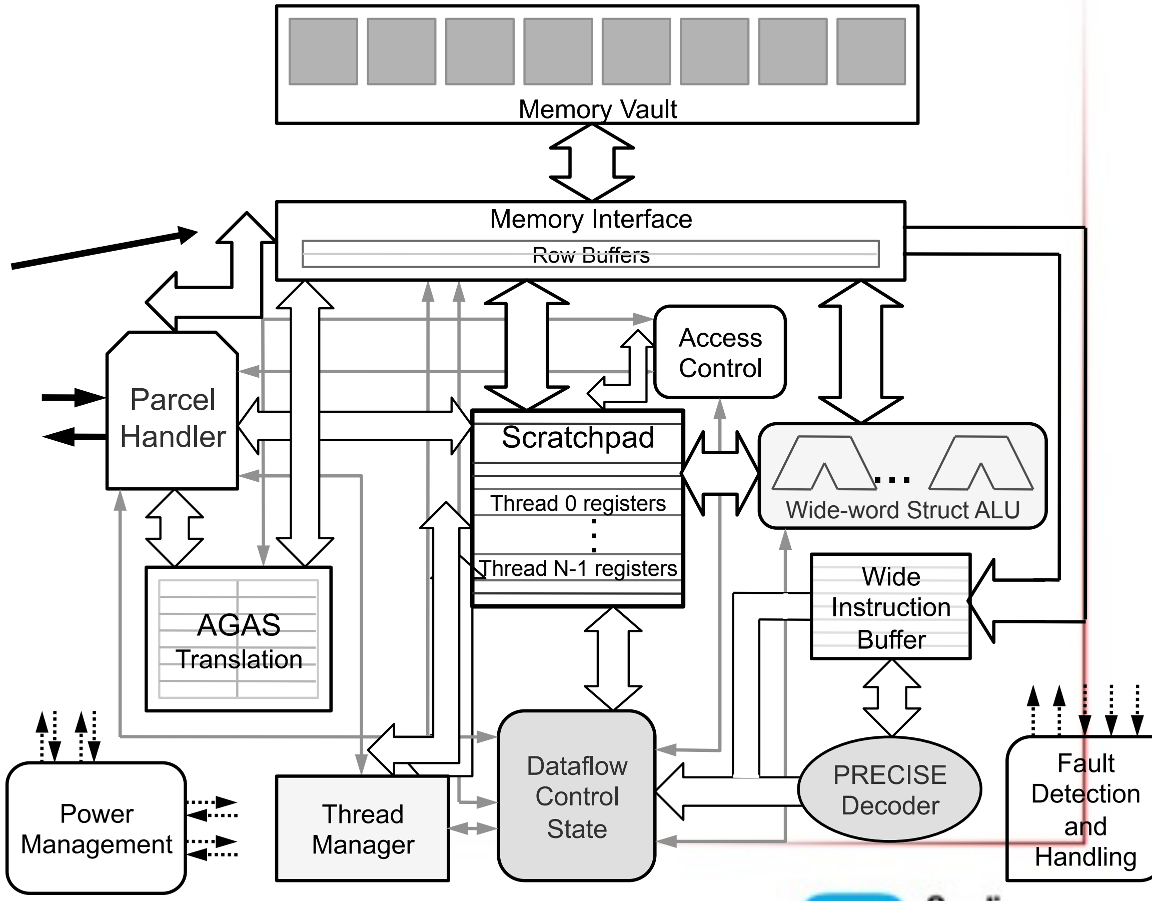
- 3D Stack: DRAM & Logic
- Memory Controllers (read/ write→RAS/CAS)
- On Chip network
- Off-chip communication
- Multiple Processing elements
  - ‘DAUs’: Close to memory controller
  - ‘VAUs’ : Closer
- Design space:
  - # of VAUs/DAUs/MCs, bandwidths, topologies, etc...



Inside the DAUs

- Simple pipeline of some sort
  - –Wide access(?)
  - –Multithreaded
- Memory/NoC access
- Interesting bits...
  - –Scratchpad: vs cache. Shared w/ registers? globally addressable?
  - –Instruction encoding: Compressed? Contains dataflow state?
  - –Global address space
  - –Integration w/ network (parcel handling)
  - –Integration w/ memory



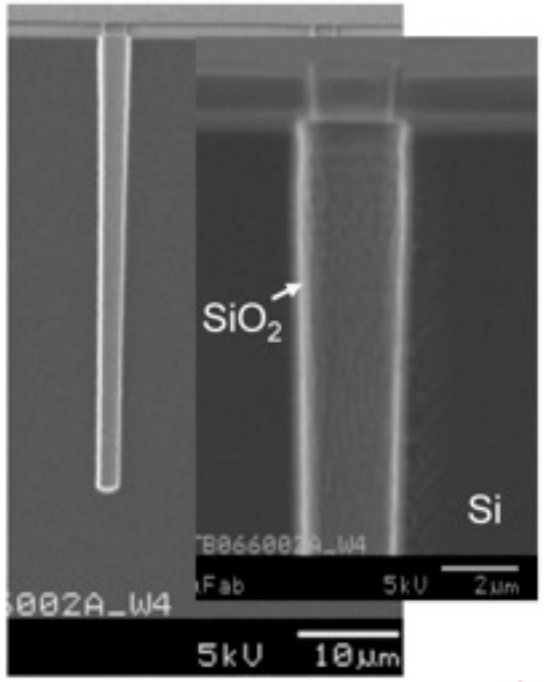


A Few

^Key Design Space Explorations

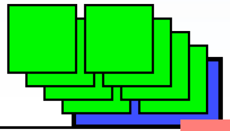
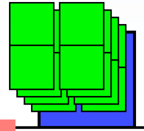
Advanced Packaging Technology

- Enables
  - Huge amounts of bandwidth at low power and latency
  - Real integration of processing and memory without prohibitive fabrication problems
    - Moving processing into memory makes percolation, possible
    - Opens new realms for message-based computation



- Limits
  - Temperature more constrained
  - May increase cost
- Need cost, power, energy, and thermal models
- Tradeoffs:
  - Depth of stack, #of stacks
  - Density of TSVs (logic vs. communication)
  - Composition of stack (optics? memory? logic?)

2015 Capability Machine Cost Study

	Adv. Packaging		Conventional System	
			24 Core	96 Core
Peak PF	124	124	303.5	286.3
Memory (PB)	20.6	20.2	20.1	20.1
GB/Core	1.7	1.6	2.2	2.7
Link BW (B/flop)	4.0		0.20	0.03
Power (MW)	14.3	13.7	96.4 !!!	27.8 !!
Cost ($M)	$161.4	$155.3	$567.3	$258.8
Memory ($M)	$101.90	$101.1	146.59	$146.59
Processor ($M)	$18.58	$24.47	$55.88	$34.13
Network ($M)	$20.69	$13.41	$256.30	$46.99
Power, RAS, racks ($M)	$20.20	$16.35	$108.58	$31.13

Requires lower peak

More wasted flops

Optics allows massive BW

Stacking decreases power

Lower Cost

“Dumber” Memory is cheaper

Amortize Packaging

1000s of pins

Power density allows more nodes/board

Global Address/Name Space

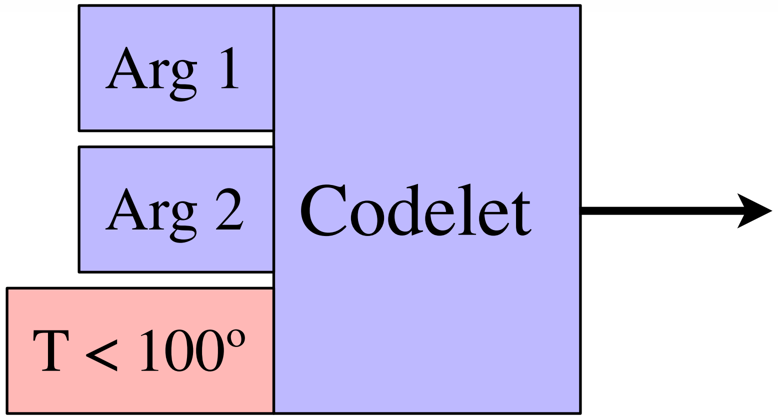
- What hardware support for lookups
  - Without hardware = too slow?
  - With hardware support = too much power?
- Which objects are in the global name space?
  - Do we need to limit the number?
- How do we partition lookups between the NIC and EMP?
- How do we partition between SW&HW lookup

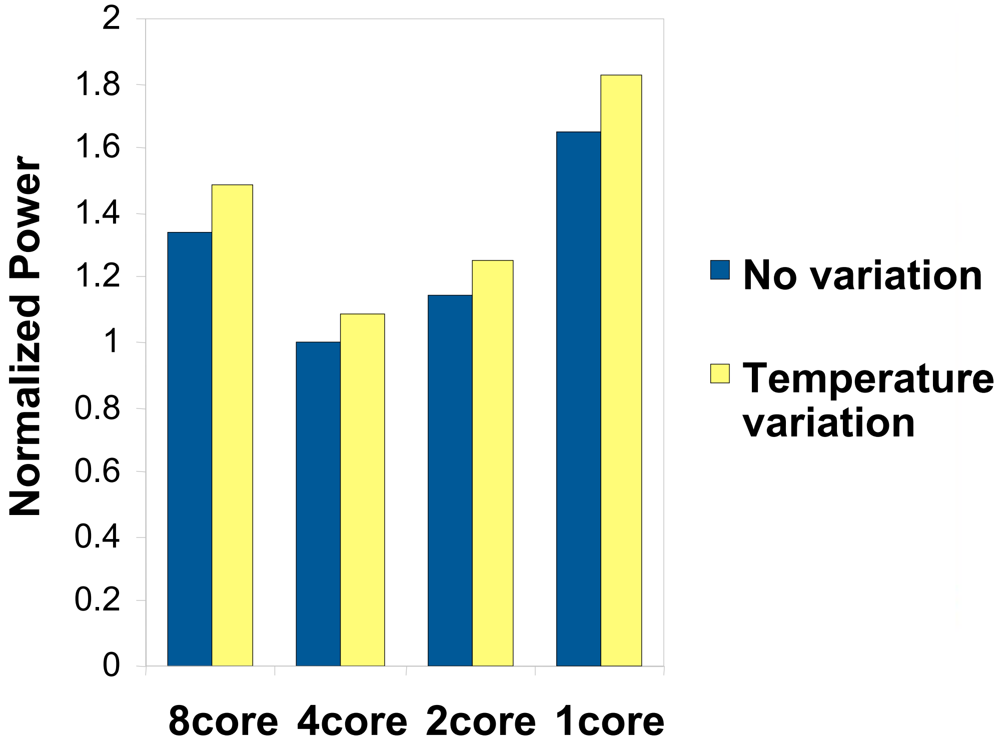
Entries	Avg Obj Size	pJ/ Access	Area mm^2	Energy (W)	Energy Budget	Area cost
8k	512KB	96	1.9	96	0.17%	0.95%
64k	64KB	229	10.5	229	0.40%	5.25%
512k	8KB	615	88.4	615	1.08%	44.20%

peta-scale rack, 57kW budget, 8B entries, 4GB stack, 22nm
1 tera-access/sec, 200 mm^2 logic part

Power/Energy Feedback / Hooks

- Feedback
- Thermal Migration
  - Move computation around to keep chip within thermal bounds
  - Possibly decrease overall energy consumption by reducing leakage current at higher temperatures
- Thermal Scheduling
  - Only do certain work when temperature/power usage is low
  - "Computational Siesta"
  - Fits in with Codelet/static dataflow model



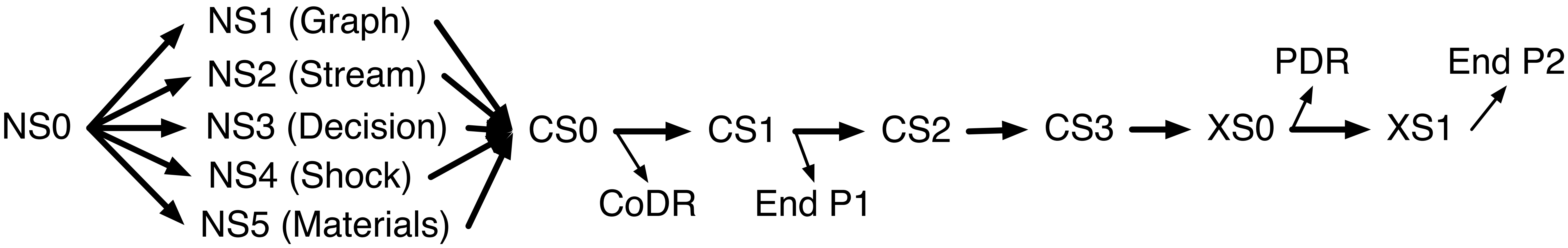


Design Space: In-memory Operations

- Functionality (PIM & AMO) & supported AMOs
- Ratios of compute in memory vs. cpu
- How do the CPU & PIM share the memory channel how (priority, master/slave, etc...)
- OS support, ISA support, runtime & compiler support
  - Protection
- What state is shared (TLB, memory, process status word, LSQ etc...)? what read/writeable?
- How does PIM to PIM communication work (thorough CPU? direct to other PIM? through NIC?)
- Do PIMs talk virtual or physical addresses?
- CPU/PIM protocol: RPC like?
- Impact on CPU pipeline (do acks effect commit of instructions, or is at a software level)?
- Separate AMO & PIM unit, or just PIM, or just AMO, or neither
- Separate units for gather/scather?

Co-Design Philosophy

Co Design Process



Paper Designs Optimized for Each Challenge Problem

Iterative Codesign of a Unified System (Integrating all NS Designs)

Prototype Proposal (for Phase 3-4)

- **NS0 Starting point**
- **Diversify into application-specific versions**
- **Merge into Conceptual System 0**
- **Iterate and refine towards prototype**

Co-Design Process

- Key Metrics
  - Energy/Power, Performance, Cost, Area
- Iterative
  - Need early results to guide design
  - Lack complete understanding of execution model, architecture, technology, applications...
  - Initial experiments will use conventional components & application implementations, before novel models/ implementations are available
  - Carefully avoid over-constraining problem, while still guiding
- Early design space exploration
  - Analytical models
  - Technology models
  - Execution-based simulation with SST

SST Simulation Project Overview

Goals

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

Status

- Current Release (2.1) at code.google.com/p/sst-simulator/
- Includes parallel simulation core, configuration, power models, basic network and processor models, and interface to detailed memory model

Technical Approach

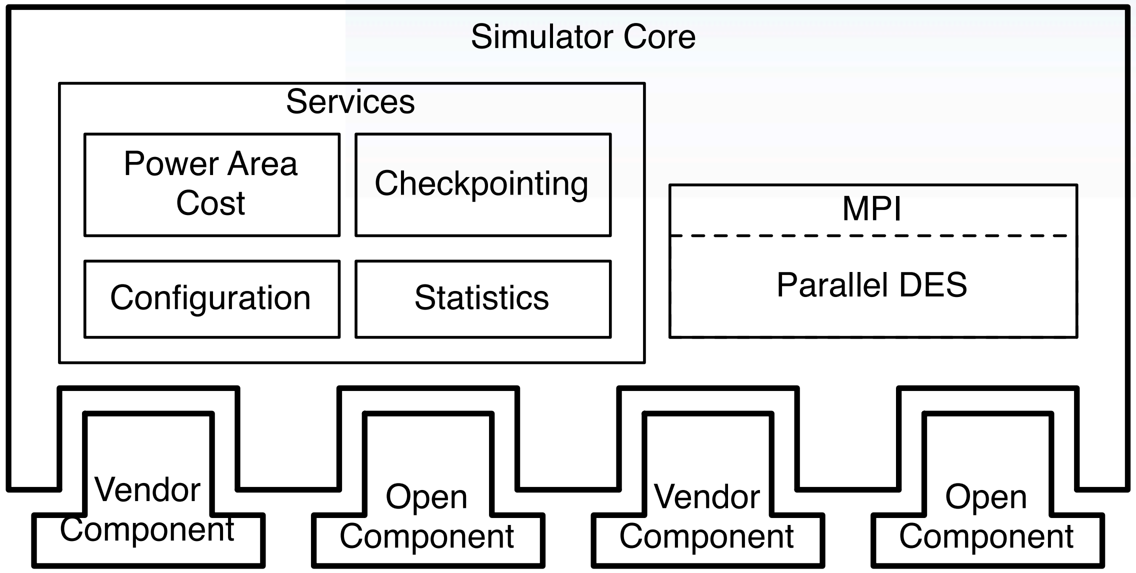
- Parallel
  - Parallel Discrete Event core with conservative optimization over MPI
- Holistic
  - Integrated Tech. Models for power
  - McPAT, Sim-Panalyzer
- Multiscale
  - Detailed and simple models for processor, network, and memory
- Open
  - Open Core, non viral, modular

Consortium

- "Best of Breed" simulation suite
- Combine Lab, academic, & industry

Component Library

- Parallel Core v2
  - Parallel DES layered on MPI
  - Partitioning & Load Balancing
  - Configuration & Checkpointing
  - Power modeling
- Technology Models
  - McPAT, Sim-Panalyzer, IntSim, Orion and custom power/energy models
  - HotSpot Thermal model
  - Working on reliability models
- Components
  - Processor: Macro Applications, Macro Network, NMSU, genericProc, state-machine, Zesto, GeM5, GPGPU
  - Network: Red Storm, simpleRouter, GeM5
  - Memory: DRAMSim II, Adv. Memory, Flash, SSD, DiskSim
  - Other: Allocation Node, IO Node, QSim



Simulator Core

Services

Power Area Cost

Checkpointing

Configuration

Statistics

MPI

Parallel DES

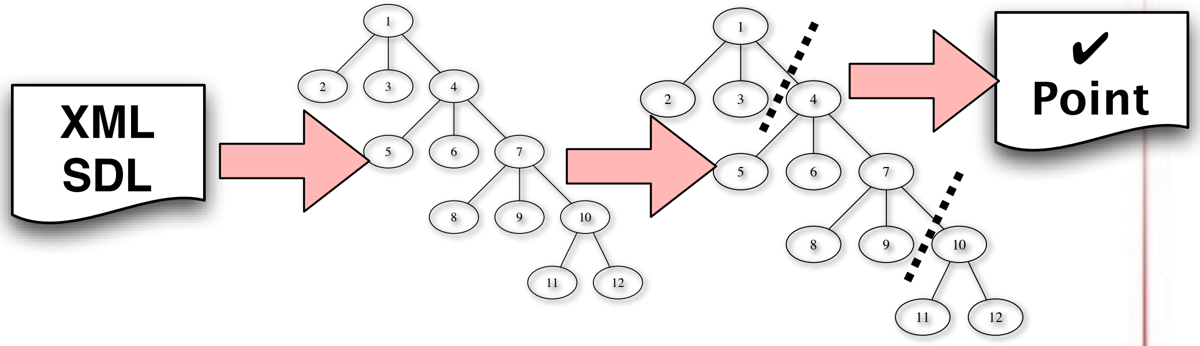
Vendor Component

Open Component

Vendor Component

Open Component

SST Simulator Core



XML SDL

✔ Point

SST Workflow

SST in UHPC

- Clearing house for new ideas
- Testbed
  - Early results quickly
  - Progressively add detail
- Bring in simulation models from others
- Provide holistic feedback (power, area, etc...)

Summary & Collaborations

- Aggressive architecture focusing on the data movement problem
- Vast design space
- Iterative application-driven codesign process

- Applications
  - Are there areas we are not looking at?
- Simulation
  - New technologies
  - Existing ‘baseline’ models
- Programming Models & Runtimes
  - How do we adapt?
  - What feedback is needed from the HW to the runtime?

Bonus slides

Mantevo MiniApp Goals

- Goal: “Best approximation of X in O(1000) lines or fewer

- Predict performance of real applications in new situations.
- Aid computer systems design decisions.
- Foster communication between applications, libraries and computer systems developers.
- Guide application and library developers in algorithm and software design choices for new systems.
- Provide open source software to promote informed algorithm, application and architecture decisions in the HPC community.

- Co-Design!

ParalleX vs. Today's Dominant Model

Element	Parallex Mechanism	Stylized Communicating Sequential Processes
Concurrency	Lightweight Threads/ Codelets	MPI Ranks/Processes
Coordination	Lightweight Control Objects (LCOs) (fine-grained)	Bulk Synchronous (or maybe by teams and messages) (coarse-grained)
Movement	of Work: Parcels of Data: PGAS and Bulk	of Work: None of Data: Bulk
Naming	Global Name Space Global Address Space	Coarse, rank/node names
Introspection	System Knowledge Graph (enables dynamic/adaptive)	Not specified by the model, in practice out-of-bands RAS network

Xcal

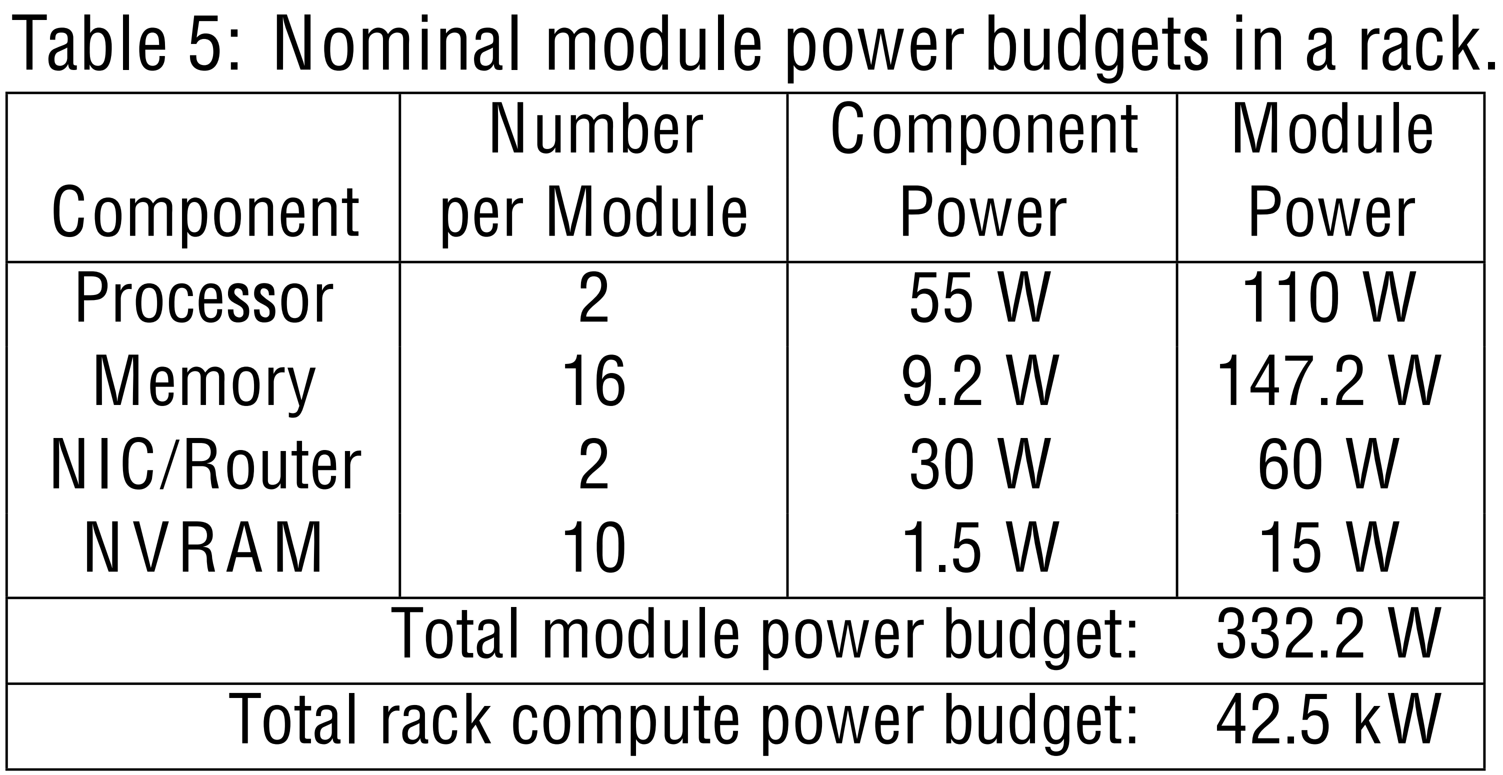
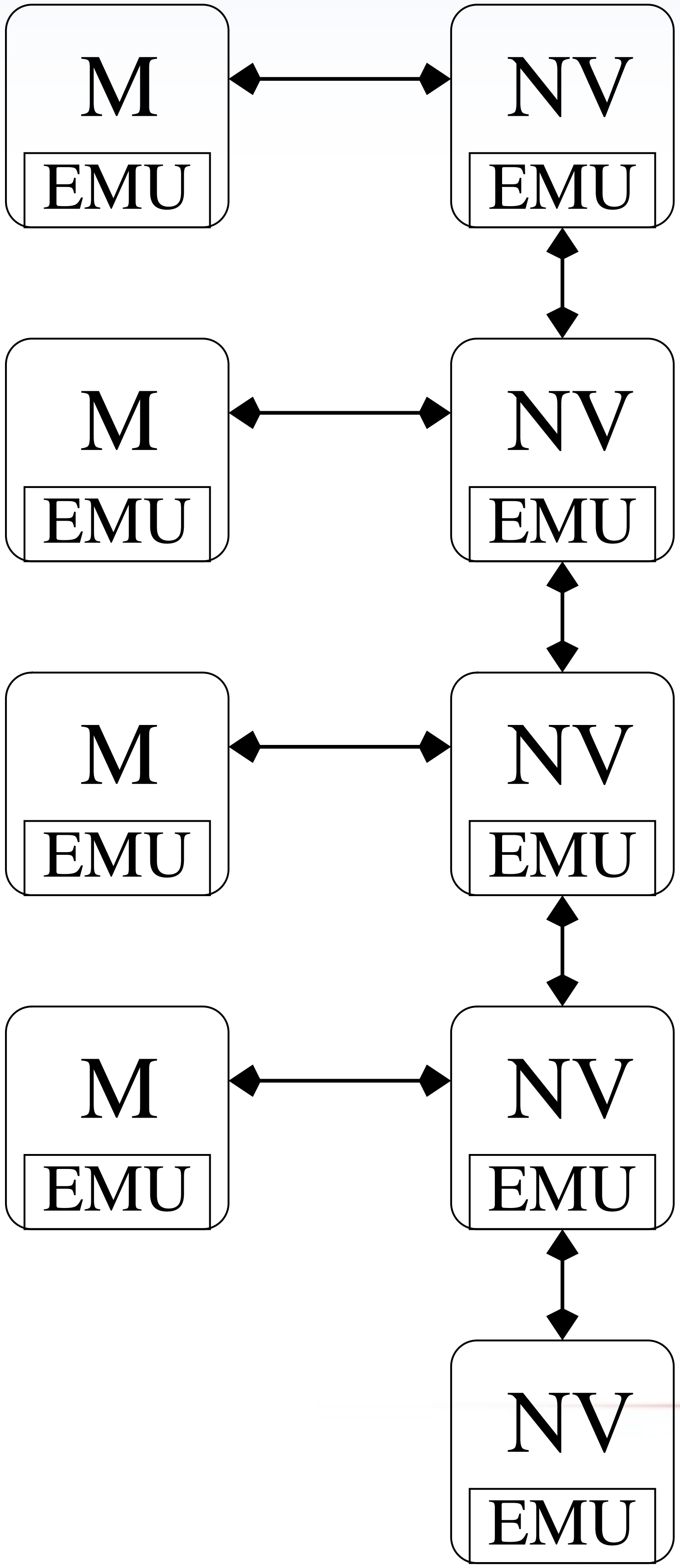


Table 5: Nominal module power budgets in a rack.

Component	Number per Module	Component Power	Module Power
Processor	2	55 W	110 W
Memory	16	9.2 W	147.2 W
NIC/Router	2	30 W	60 W
NVRAM	10	1.5 W	15 W
		Total module power budget:	332.2 W
		Total rack compute power budget:	42.5 kW



Sandia's Mantevo Project

- Mantevo is a collection of small and agile mini-applications to predict performance and explore new system architectures
  - Potential rewrites allow exploring programming models
  - http://software.sandia.gov/mantevo
  - Code collection is Open Source

- Currently in the Mantevo suite:
  - HPCCG (conjugate gradient)
  - MiniMD (MD force calcs, e.g. LAMMPS)
  - MiniFE (unstructured implicit finite element)
  - MiniXyce (in progress) (circuit modeling)
  - LANL is exploring developing a mini-IMC code for radiation transport