SAND2009-3825P

Sandia Simulation and Networking

Arun Rodrigues

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.

SST Simulator

View of the Simulation Problem

Scale.....

Many
Cores
+
Memory

X

Many
Many
Nodes

X

Many
Many
Many
Threads

Multiple Audiences.....

Network
Processor
System

X

Application writers
purchasers
designers

X

system procurement
algorithm co-design
architecture research
language research

X

present systems
future systems

Complexity.....

Multi-Physics Apps
Informatics Apps

X

Communication Libraries
Run-Times
OS Effects

X

Existing Languages
New Languages

Constraints.....

Performance
Cost

Power
Reliability

Cooling
Usability

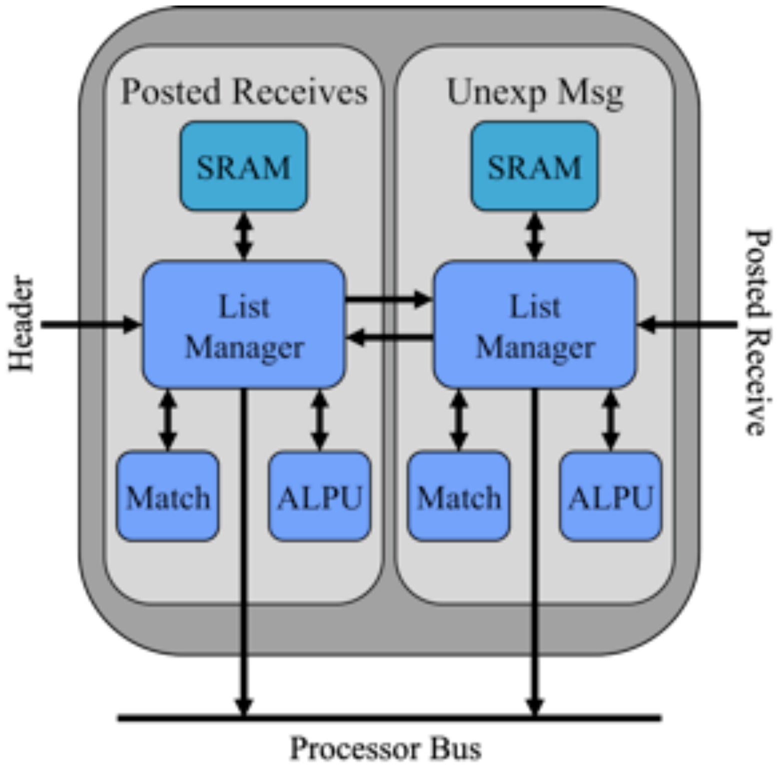
Risk
Size

HPC Simulation: More Challenges

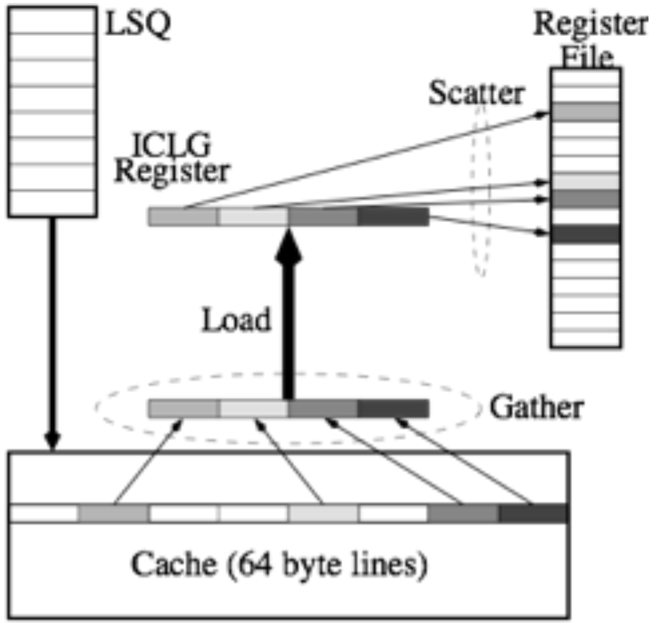
- **Network/Application Feedback**: A static trace or simple statistical model will not capture the causal relationships between messages.
- **Scalability**: Many network effects only become apparent at hundreds or thousands of nodes.
- **Variable Processor/Memory/Network Systems**: Local interactions can have global performance implications.
- **Ability to Model Message Overheads:** Overheads in the network (e.g. packetization, protocol overhead) and messaging library (e.g. MPI matching, message assembly) can have a major effect on performance.
- **Ability to Explore Programming Models**: Novel hardware will require novel programming techniques and capabilities.
- **Power and Economic Effects:** Power and cost are the key limiting factors on system design. Any system model must be able provide feedback on the power and cost implications of new architectural features.

SST Fundamental to Several Projects

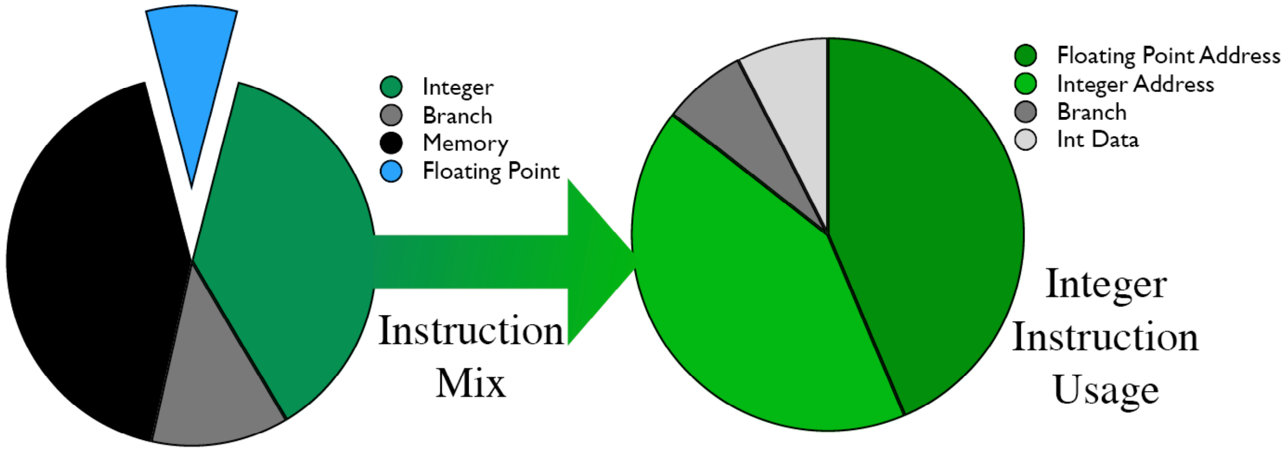
- Microarchitecture
  - Inter CacheLine Gather (ICGL)
  - Recon. FU (Wisc./SNL)
  - FP Aggregates
  - In-Memory Ops
- Application analysis
  - Memory Footprint
  - Instruction Usage
- Network/MPI
  - NIC Tradeoffs
  - MPI Acceleration
- Programming Models
  - PIM Compiler work (SNL/Rice)
  - ParallX (LSU/SNL)(FastOS)
  - Transactional Memory (ORNL)
  - QThreads
- Processor-In-Memory (LDRD)



MPI Accelerator Tested with SST



Inter CacheLine Scatter/ Gather Microarchitecture



Instruction Mix & Usage for Sandia Applications

Simulation Project

Goals

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

Technical Approach

- Multiscale
  - Cycle-accurate to analytic
  - Instruction-based to message-based
- Parallel
  - 1000s of simulated nodes on 100s of real nodes
- Holistic
  - Integrated Tech. Models

Consortium

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

Parallel Simulation

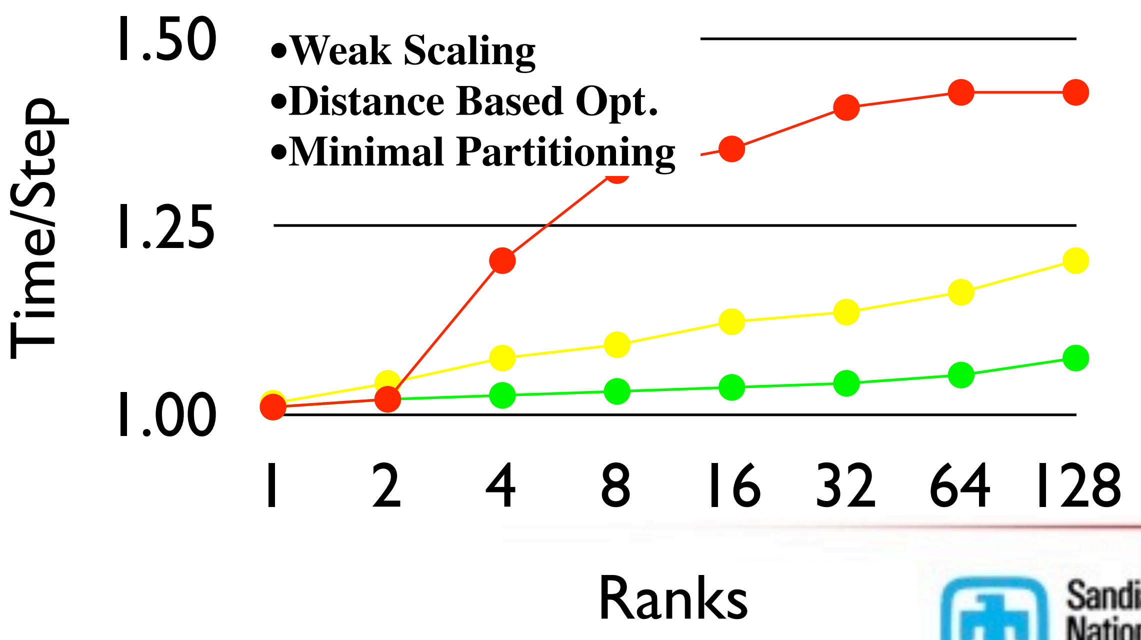
```
while(event = getNextEvent(queues)) {
  cycle = event->time;
  if (event->isClockEvent) {
    event->component->preTic();
  } else if (event->exchange) {
    startSends();
    recv();
    finishSends();
  } else if (event->checkpoint) {
    checkpoint();
  } else {
    event->component->handleEvent();
  }
}
```

Parallel Core Pseudocode

- **Requirements**
  - –High speed, Parallel, Scalable
  - –Multiple clock domains
  - –Checkpointing
- **Implementation**
  - –Conservative distance-based DES optimization
  - –Multi-criteria partitioning
  - –Built on MPI
  - –Future: FPGA acceleration

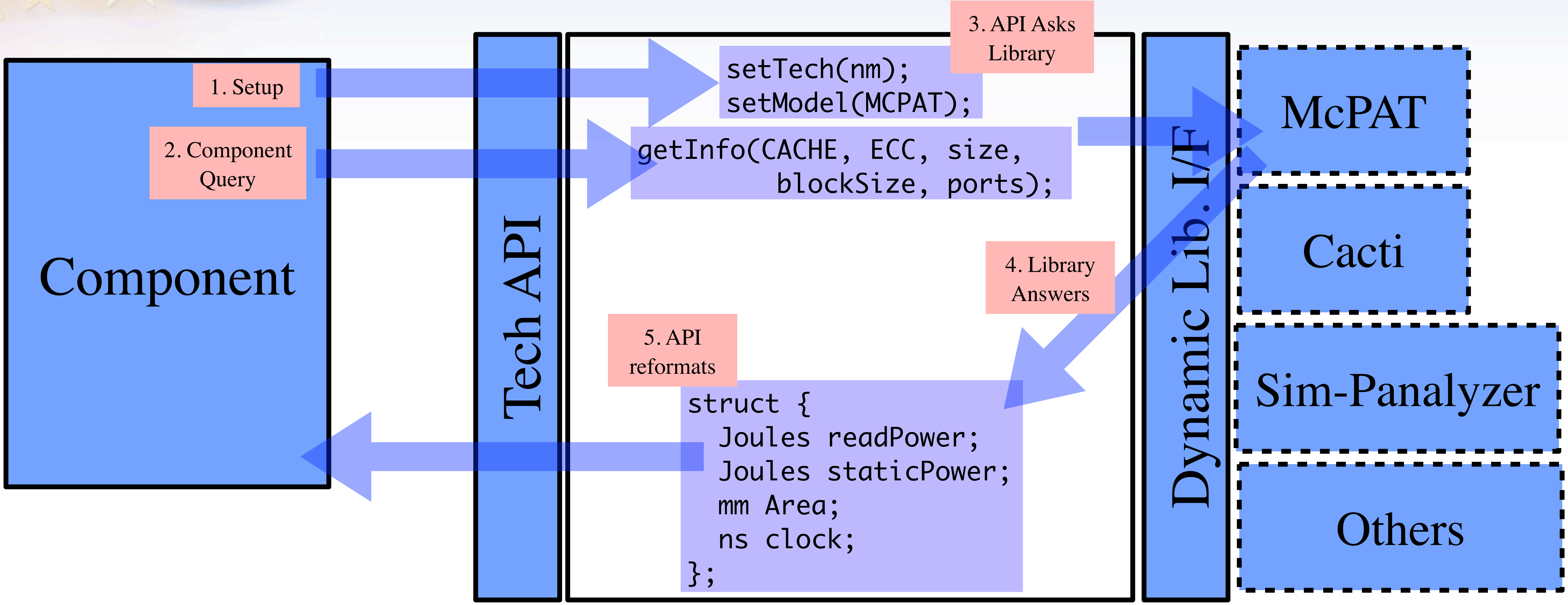
Usage Model	Message Traces, Symbolic Workload Descriptions	Execution Based	Execution w/ FPGA Acceleration
Real Nodes	100s-1000s	100-1000s	1-10
Simulated Nodes	10000s-100000s	100s-1000s	1-10
Goal	System Scaling Behavior	Cycle-level system performance	Co-design

Usage Models



Distance-based DES Scaling

Holistic Simulation



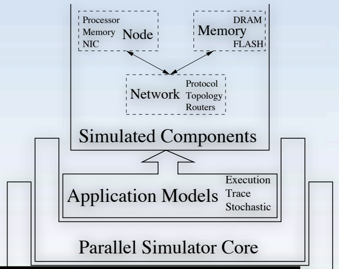
```
setTech(nm);
setModel(MCPAT);
```

```
getInfo(CACHE, ECC, size,
         blockSize, ports);
```

```
struct {
  Joules readPower;
  Joules staticPower;
  mm Area;
  ns clock;
};
```

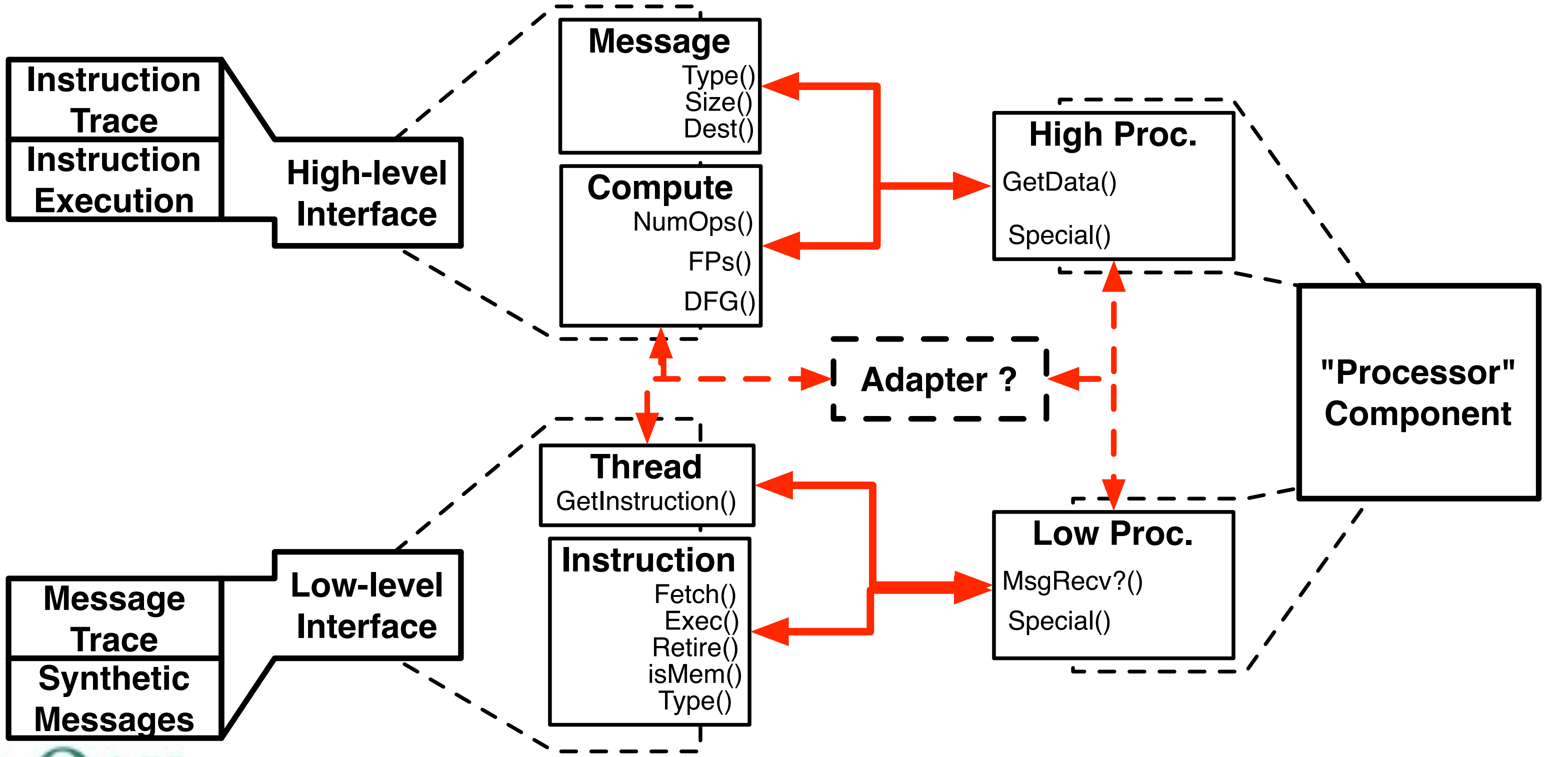
- **Create interface to multiple technology libraries**
  - **–Power/Energy**
  - **–Reliability**
  - **–Area/Timing estimation**
- **Make it easier for components to model technology parameters**

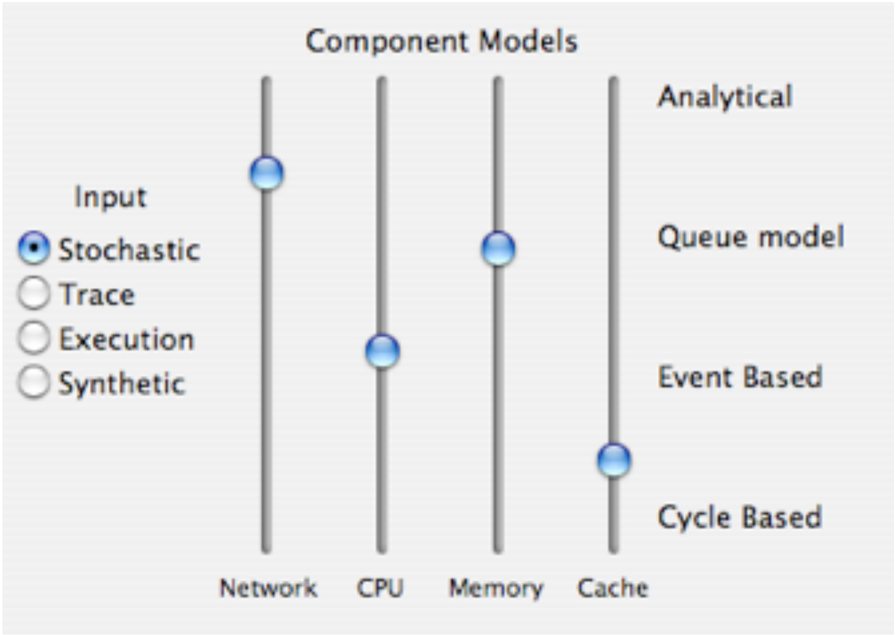
Multi-Scale Simulation



- Goal: Interface component reuse at different scales
- High- & Low-level interfaces (more?)
  - Allows multiple input types
  - Allows multiple input sources
    - Traces, stochastic, state-machines, execution...
  - Adapter objects to translate?

	High-Level	Low-Level
Detail	Message	Instruction
Fundamental Objects	Message, Compute block, Process	Instruction, Thread
Static Generation	MPI Traces, MA Traces	Instruction Trace
Dynamic Generation	State Machine	Execution

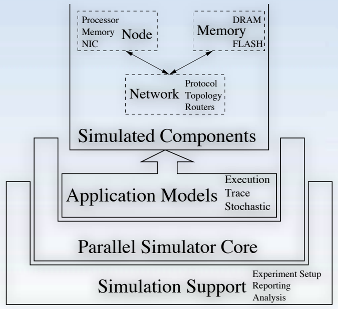


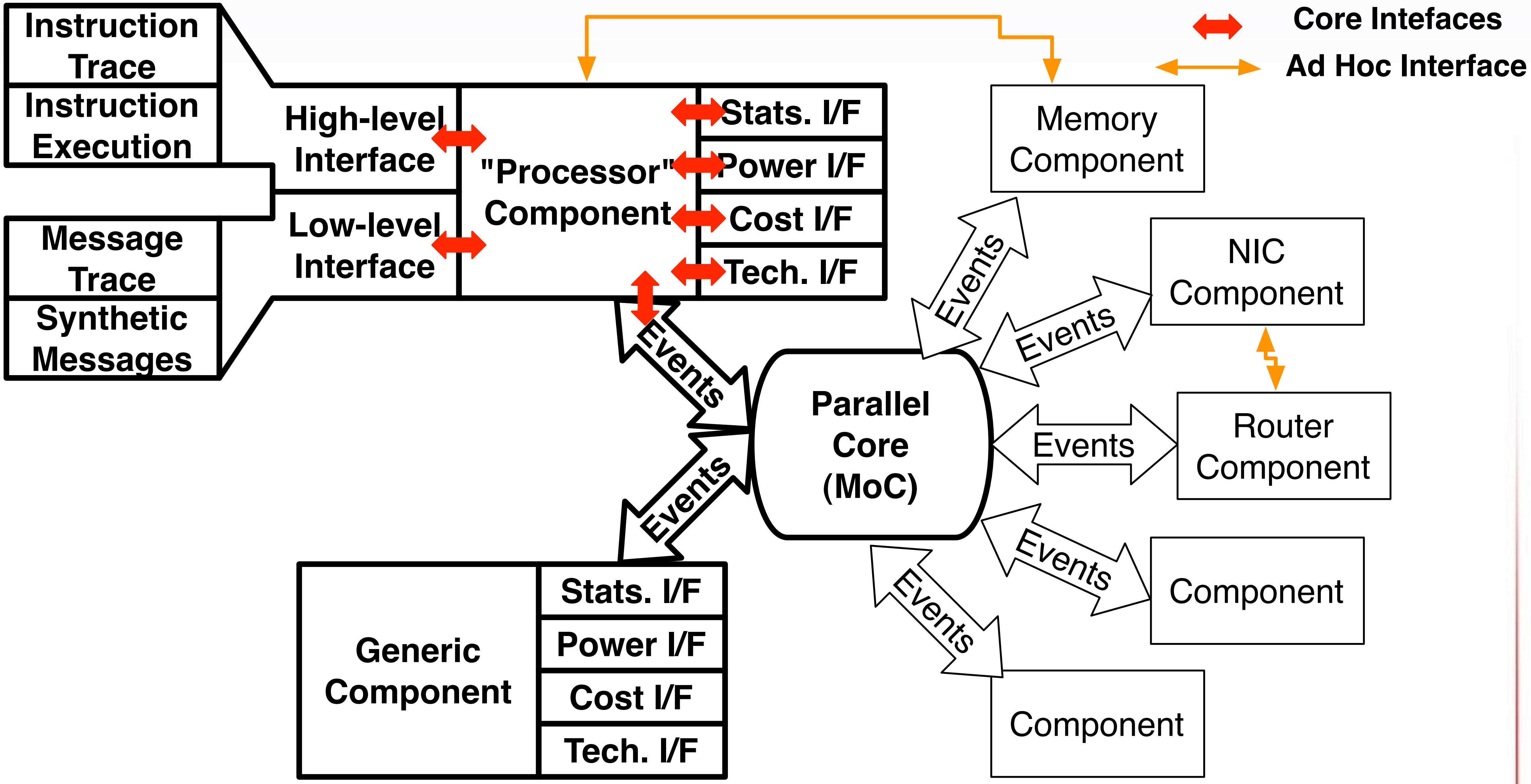

Component Models

Input
● Stochastic
○ Trace
○ Execution
○ Synthetic

Multiscale Parameters

Proposed Structure




Core Intefaces

Ad Hoc Interface

Instruction Trace

Instruction Execution

Message Trace

Synthetic Messages

High-level Interface

Low-level Interface

"Processor" Component

Stats. I/F

Power I/F

Cost I/F

Tech. I/F

Memory Component

NIC Component

Router Component

Component

Component

Parallel Core (MoC)

Events

Events

Events

Events

Events

Events

Events

Generic Component

Stats. I/F

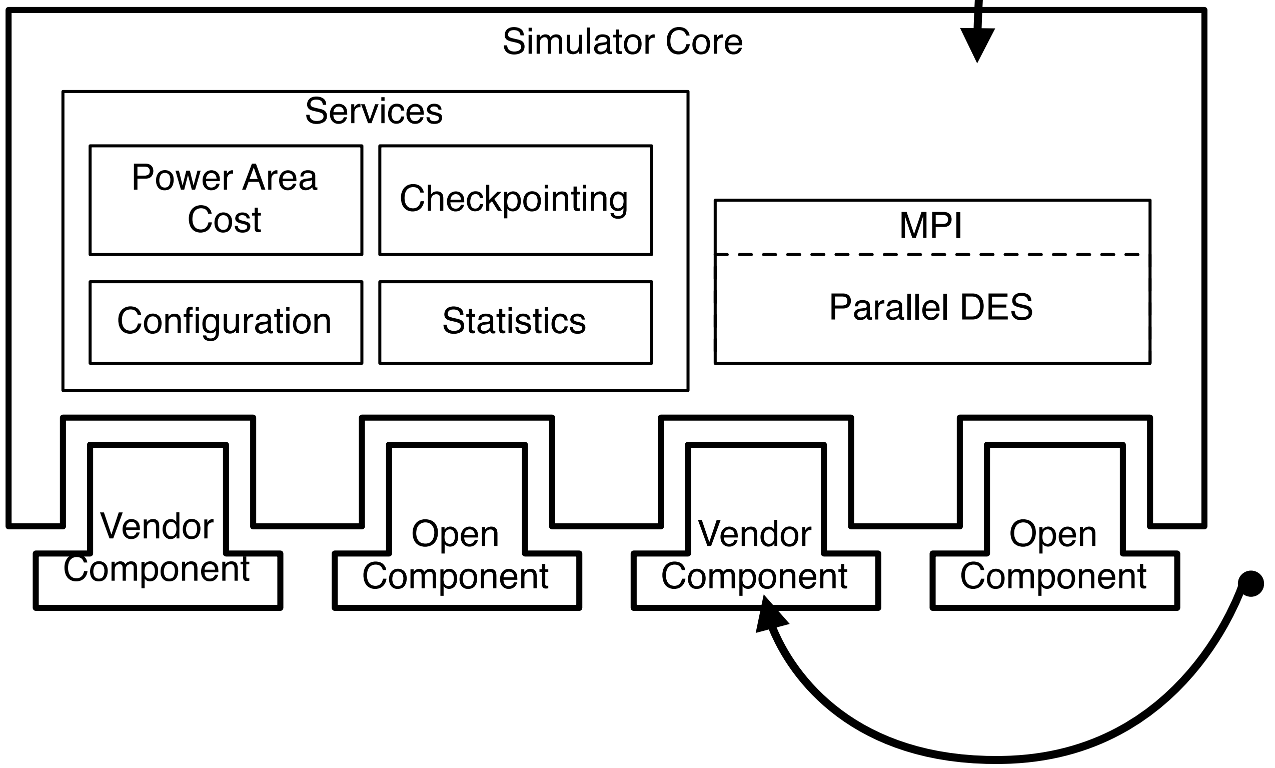
Power I/F

Cost I/F

Tech. I/F

- Separate Software/Front-End from Hardware/Timing/Back-End
- Standard interfaces for power, area, cost?

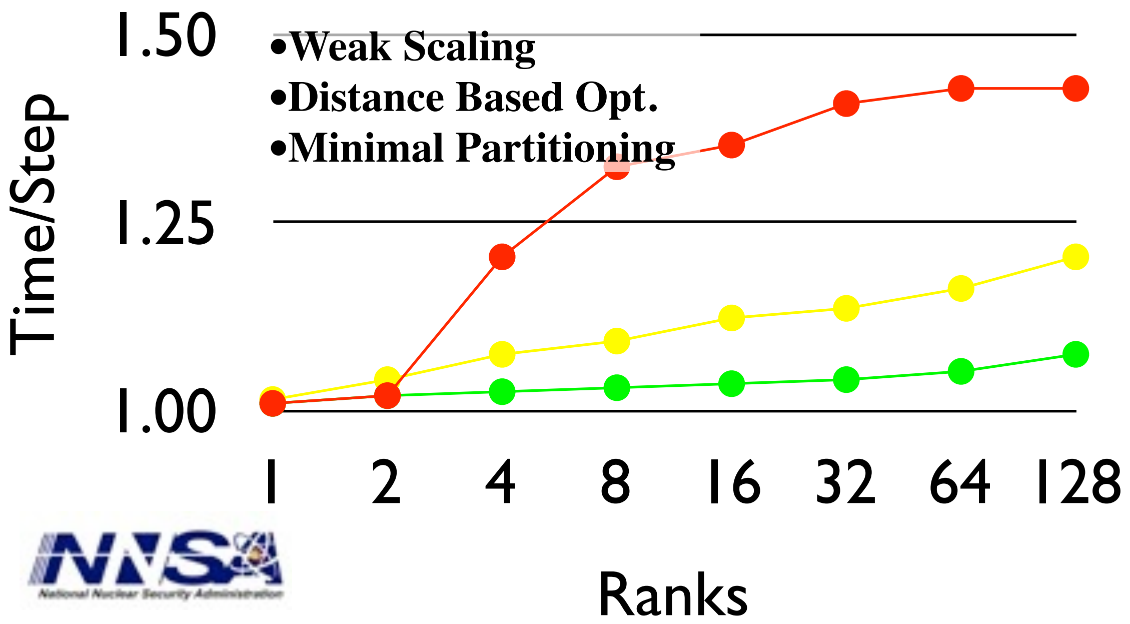
Simulator Framework Model

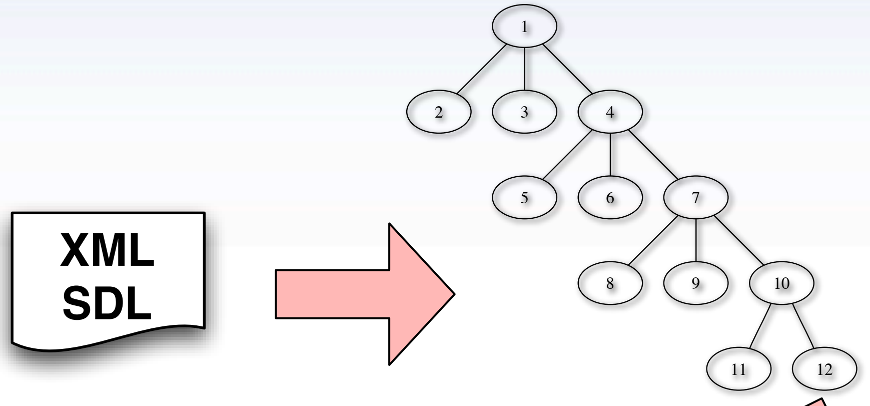


- Simulator Core will provide...
  - Power, Area, Cost modeling
  - Checkpointing
  - Configuration
  - Statistics gathering
  - Parallel Component-Based Discrete Event Simulation
    - MPI hidden from user
    - Multiple clocks
- Components
  - Ships with basic set of open components
  - Industry can plug in their own models
    - Under no obligation to share

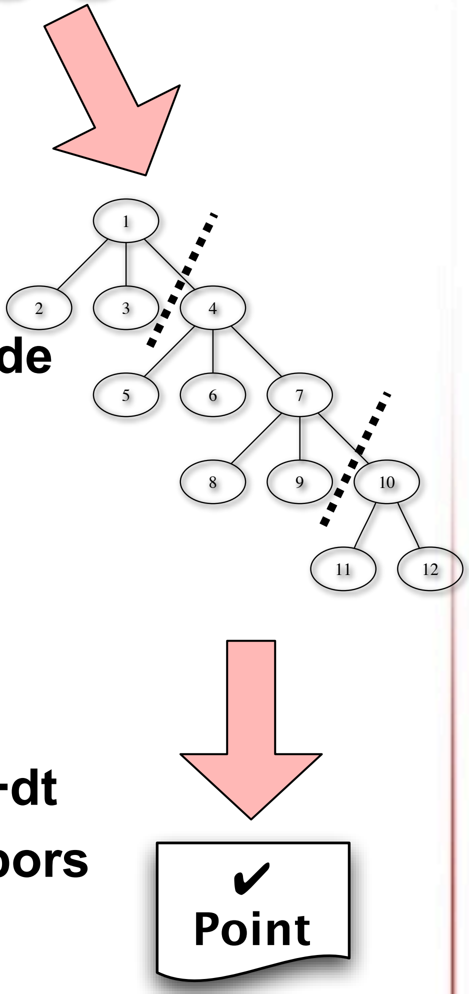
Parallel SST Core

- Strawman
  - “API Testbed”
  - < 1000 lines of code
  - Demonstrates basic functionality
    - Sim startup
    - Component partitioning
    - Checkpointing
    - Event passing
- Current work
  - System Description Language
  - Refining Parallel DES
  - Event interfaces
  - Scaling





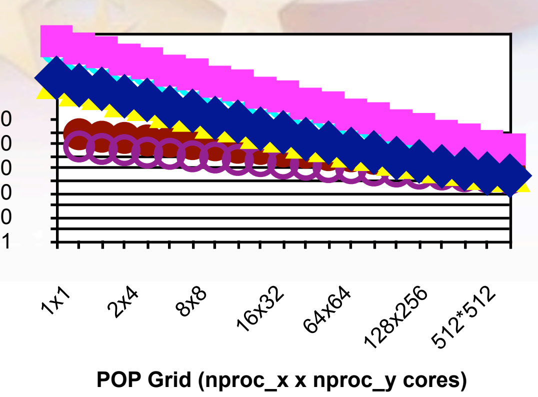
- Initial/Setup Mode:
  - 1. Load config file(s)
  - 2. Generate component graph
  - 3. Partition graph
  - 4. Instantiate components on each node
  - 5. Dump initial checkpoint
- Run Mode:
  - 1. Read checkpoint from disk
  - 2. Apply Edits
  - 3. Run Loop
    - a. advance components upto time+dt
    - b. exchange messages with neighbors
    - c. goto (3a)



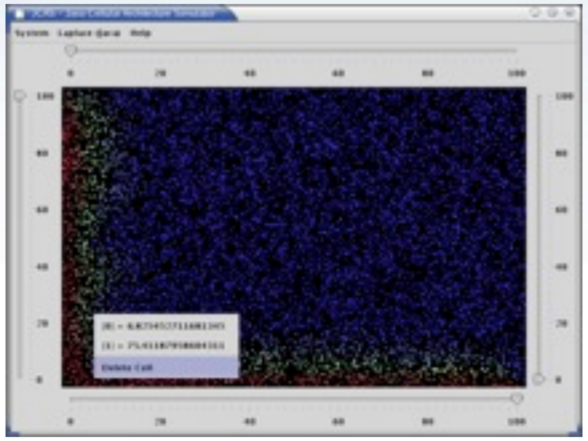
Consortium

- IAA Simulation effort is a community effort
- Seeking more partners…

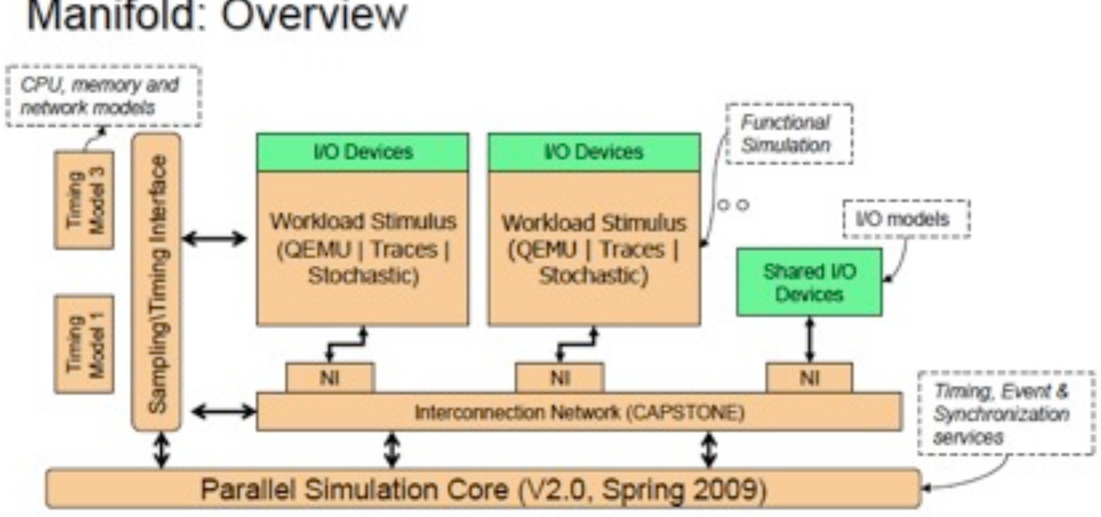
- Current consortium
  - Sandia (Structural Simulation Toolkit)
  - ORNL (Scalable application models)
  - U. Maryland (DRAMSim II)
  - U.Texas-Austin (FAST)
  - Georgia Tech (CAPSTONE/Manifold)
  - JCAS (ORNL)
  - Seshat (SNL)

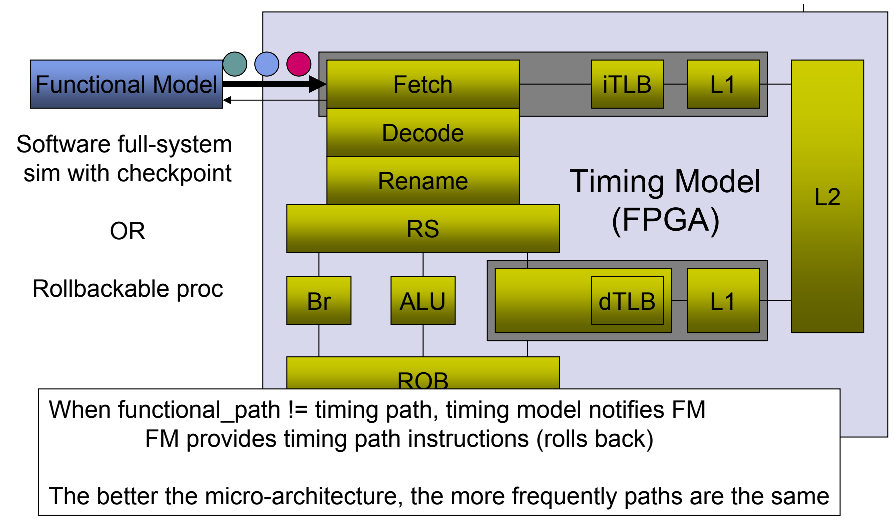


Modeling Assertions



JCAS Vizualizer

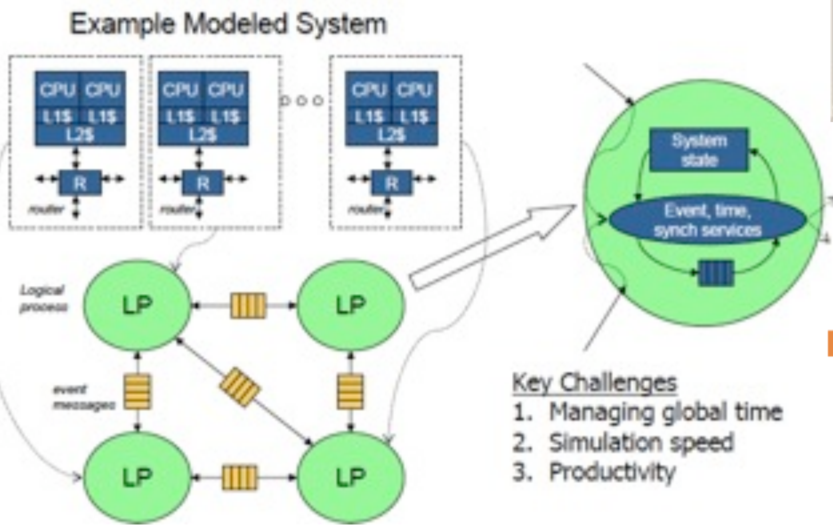

Manifold: Overview
CPU, memory and network models
Timing Model 3
Timing Model 1
Sampling\Timing Interface
I/O Devices
Workload Stimulus (QEMU | Traces | Stochastic)
I/O Devices
Workload Stimulus (QEMU | Traces | Stochastic)
Functional Simulation
I/O models
Shared I/O Devices
NI
NI
NI
Interconnection Network (CAPSTONE)
Timing, Event & Synchronization services
Parallel Simulation Core (V2.0, Spring 2009)



When functional_path != timing path, timing model notifies FM
FM provides timing path instructions (rolls back)

The better the micro-architecture, the more frequently paths are the same

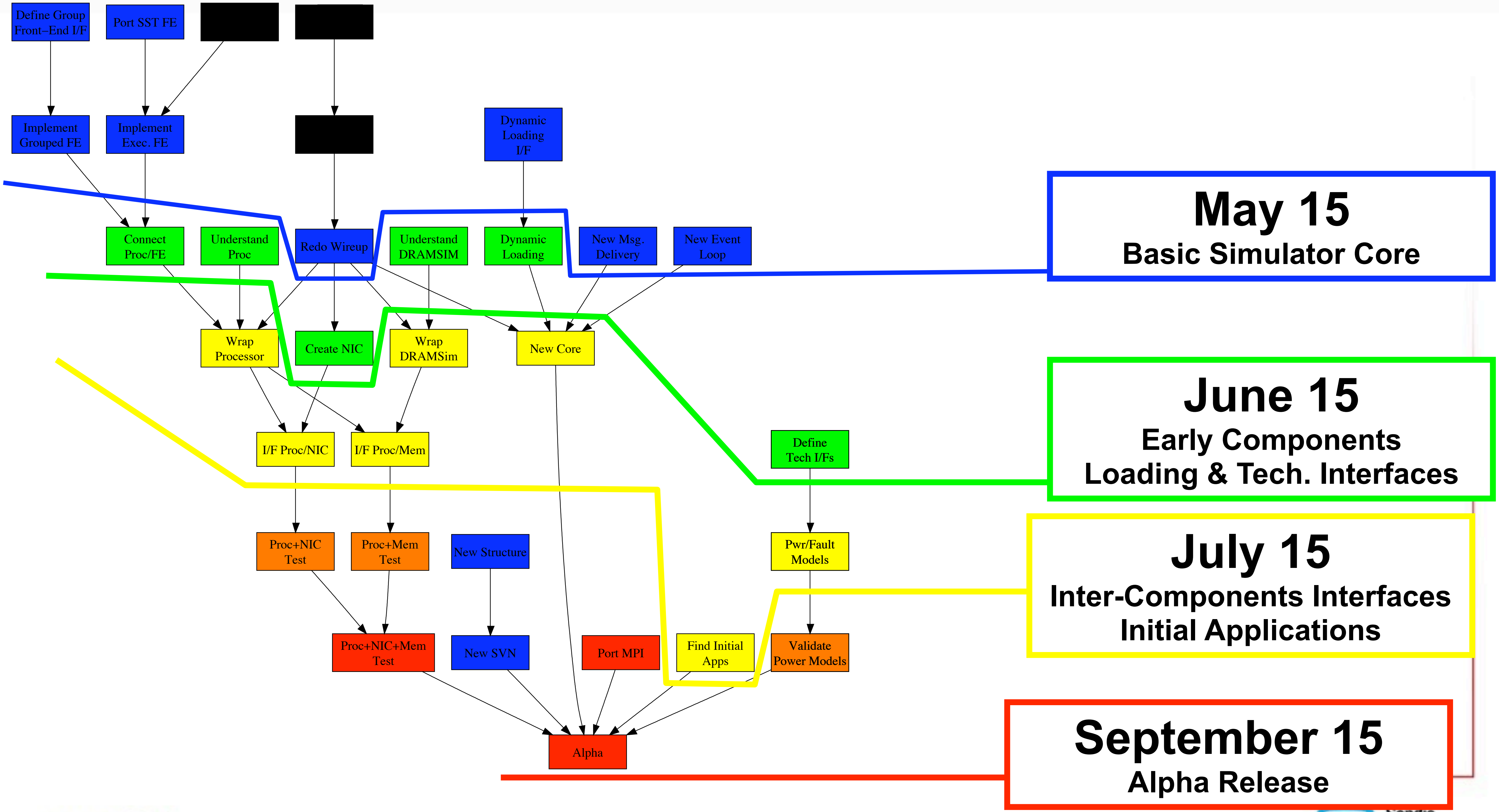
FAST



Key Challenges
1. Managing global time
2. Simulation speed
3. Productivity

DRAMSim II

Initial Project Plan



Define Group Front–End I/F

Port SST FE

[redacted]

[redacted]

Implement Grouped FE

Implement Exec. FE

[redacted]

Dynamic Loading I/F

Connect Proc/FE

Understand Proc

Redo Wireup

Understand DRAMSIM

Dynamic Loading

New Msg. Delivery

New Event Loop

Wrap Processor

Create NIC

Wrap DRAMSim

New Core

I/F Proc/NIC

I/F Proc/Mem

Define Tech I/Fs

Proc+NIC Test

Proc+Mem Test

New Structure

Pwr/Fault Models

Proc+NIC+Mem Test

New SVN

Port MPI

Find Initial Apps

Validate Power Models

Alpha

May 15
Basic Simulator Core

June 15
Early Components
Loading & Tech. Interfaces

July 15
Inter-Components Interfaces
Initial Applications

September 15
Alpha Release

Of FLITS and FLOPS: Balancing Energy and Interconnect Performance

Scott Hemmert

Scalable Computer Architectures
Sandia National Laboratories

SAND2009-2588C

Sandia is a Multiprogram Laboratory Operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy Under Contract DE-ACO4-94AL85000.

Contributors

- **Sandia**
  - Jim Ang
  - Brian Barrett
  - Ron Brightwell
  - Kurt Ferreira
  - Sue Kelly
  - Jim Laros
  - Kevin Pedretti
  - Courtenay Vaughan

- **Indiana University**
  - Torsten Hoefler

System-level Interconnect and Energy

- Interconnect performance is the key factor in determining how well many applications scale

- With increasing bandwidths, interconnect power is becoming a real concern
  - Serdes don't turn off well (OK, they turn off fine, they just don't turn back on quickly, due to channel initialization times)
    - Uses power whether valid data is moving through the network or not

- A lot of discussion lately on minimizing picojoules/bit

- However, interconnects are not used in isolation and a system view is vital to maximizing energy efficiency
  - NIC and router architectures, topologies and MPI implementations all play an important role

Application Case Study: CTH

- CTH is a multi-material, large deformation, strong shock wave, solid mechanics code developed at Sandia National Laboratories. CTH has models for multi-phase, elastic viscoplastic, porous and explosive materials.

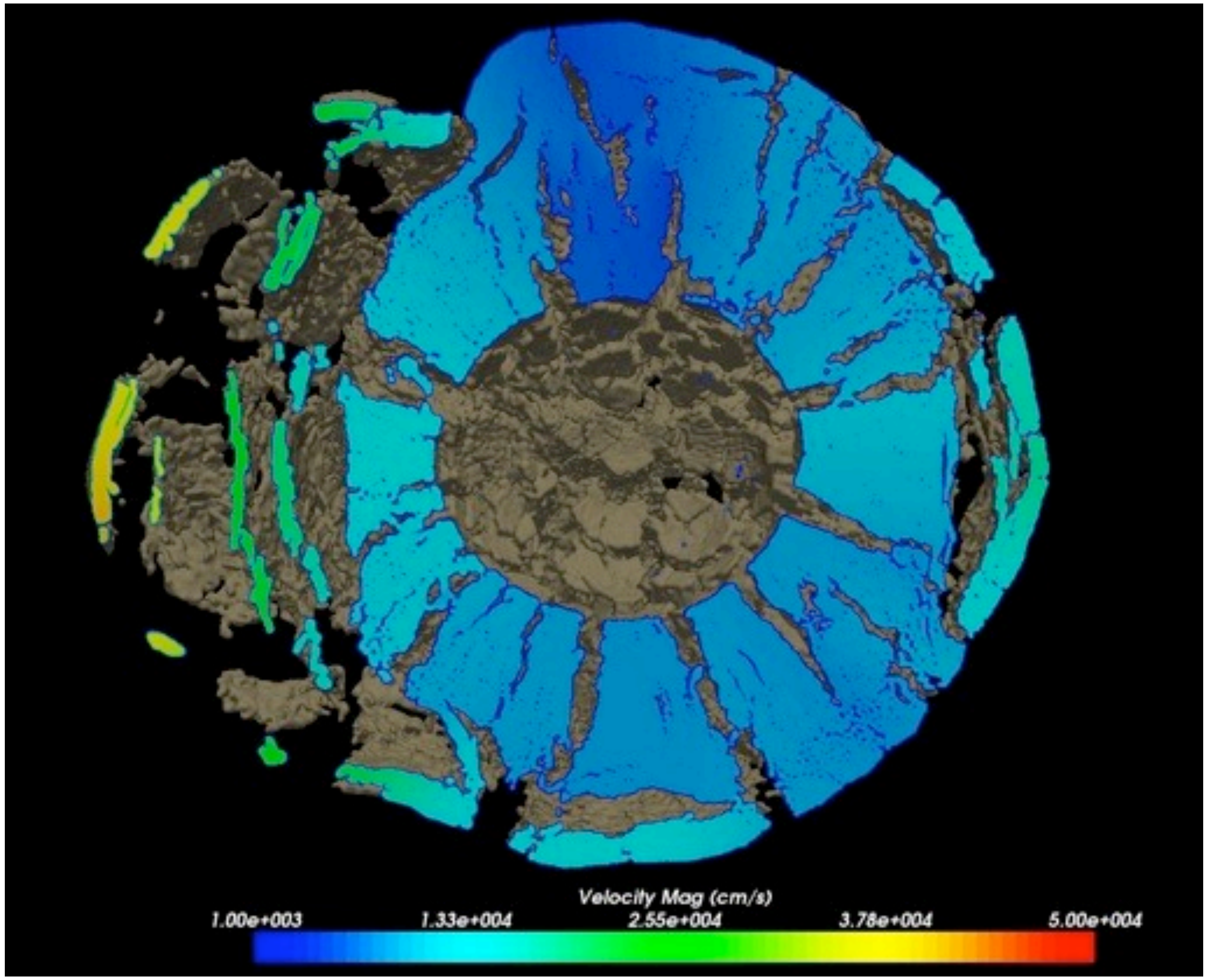
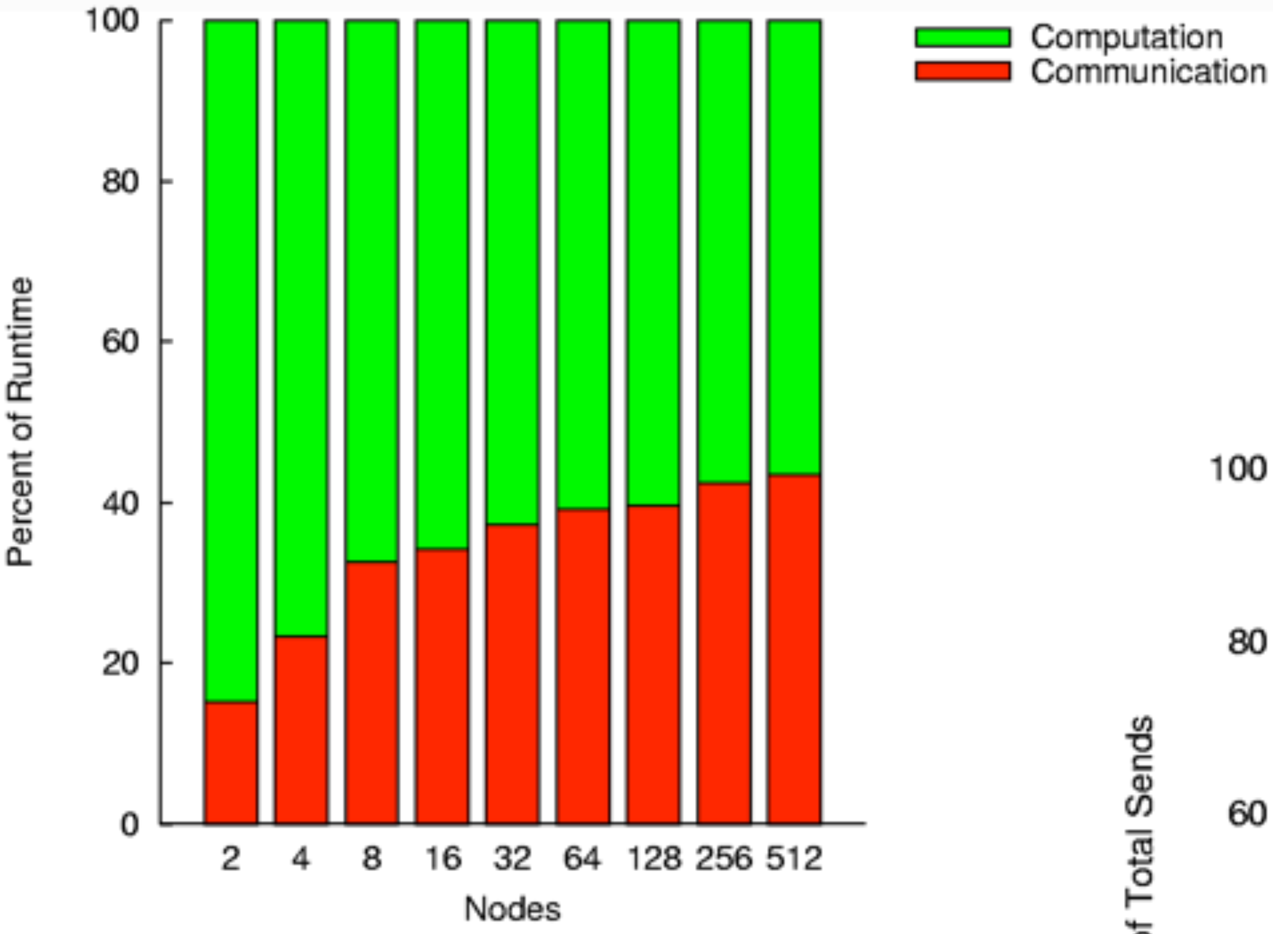


Image courtesy of ASC

Asteroid Golevka measures about 500 x 600 x 700 meters. In this CTH shock physics simulation, a 10 Megaton explosion was initiated at the center of mass. The simulation ran for about 15 hours on 7200 nodes of Red Storm and provided approximately 0.65 second of simulated time.

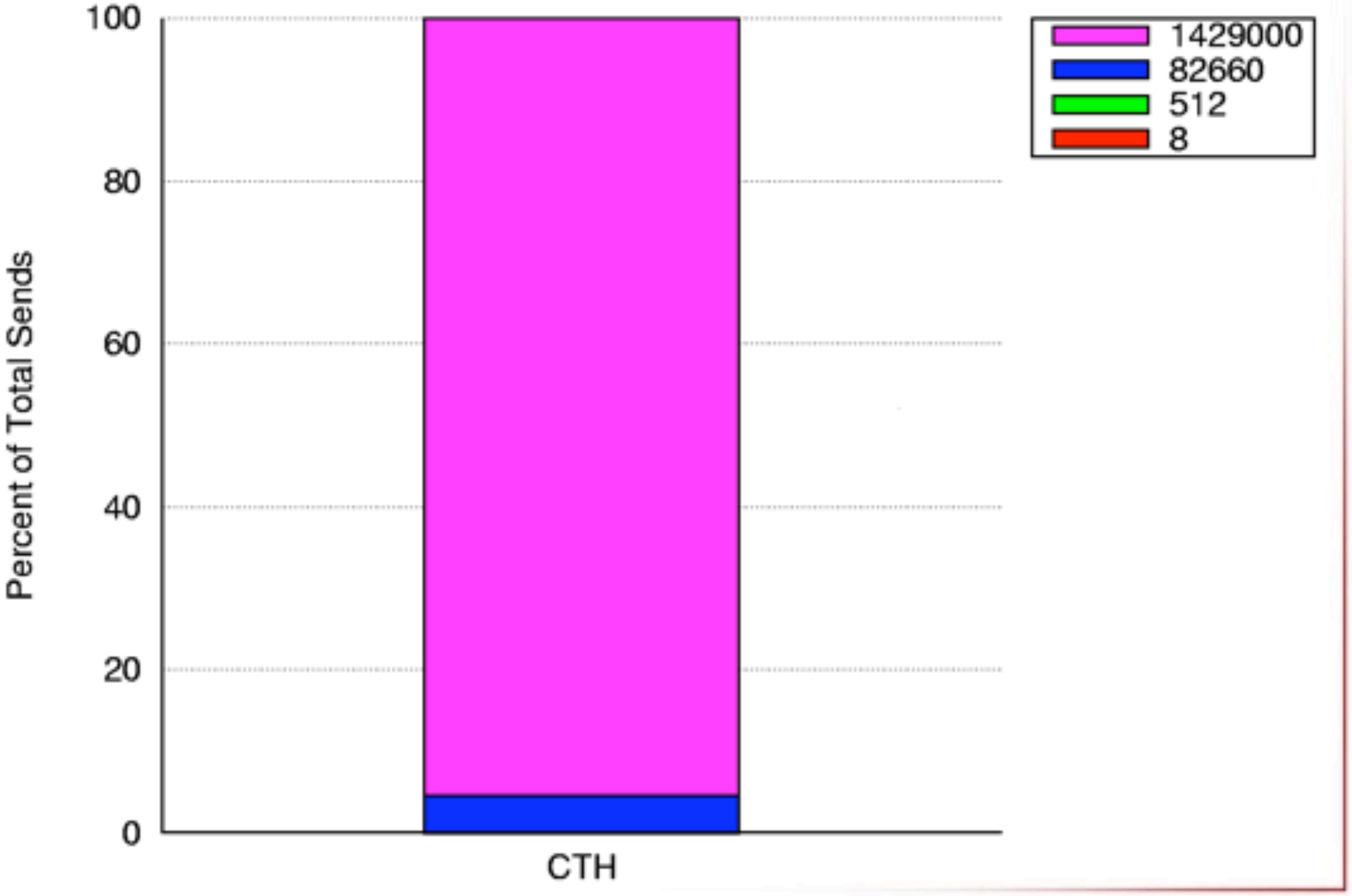
Application Case Study: CTH

Shaped Charge Problem (weak scaling)



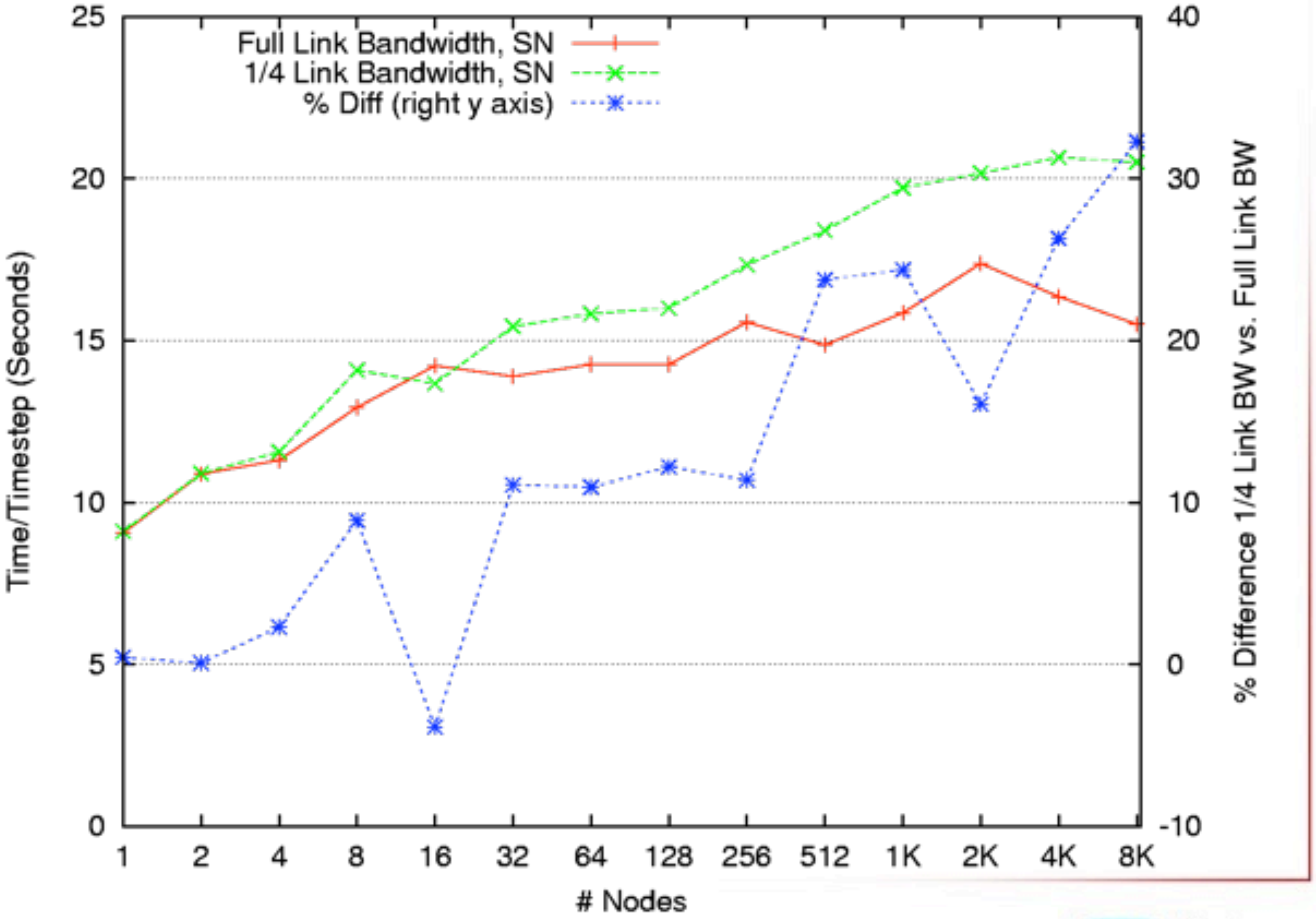
As job size increases, communication time can grow to consume around 40-50% of the runtime.

CTH communication is dominated by long messages.



CTH Bandwidth Degradation Study

- **Uses capabilities built into the Red Storm SeaStar interconnect to turn off interconnect router lanes at boot time**
  - **Links are made up of 4 3-bit subchannels that can be independently enabled**
- **Measure application performance at full and one-quarter link bandwidth**
- **At largest measured job size, quartering bandwidth leads to 32% longer runtime**

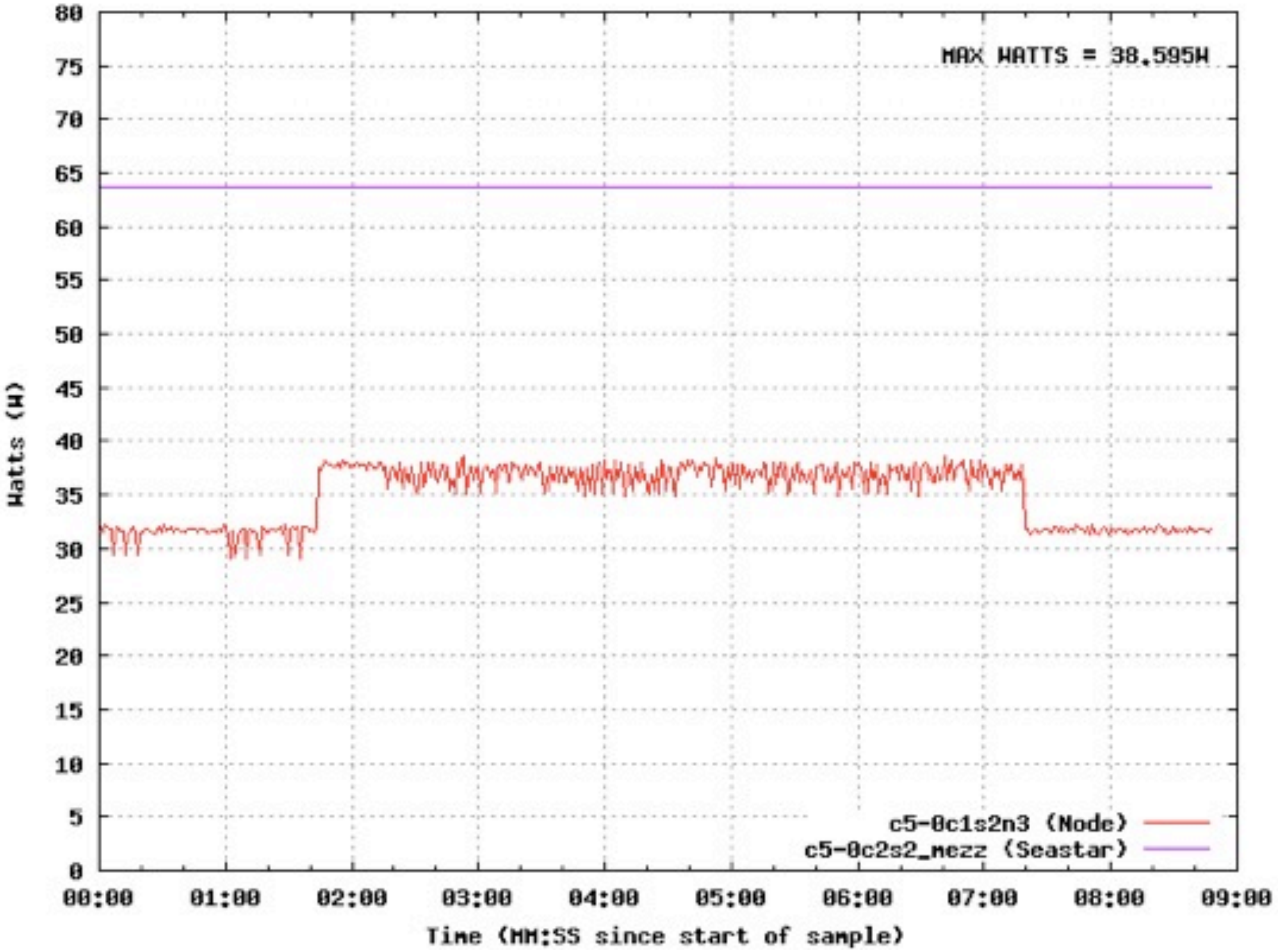


CTH Power Signature Study

- Power measured using Red Storm's built-in current monitors

Total Node Power:

CPU:	**37 (red)**
SeaStar:	**16 (blue/4)**
Memory:	**20 (estimated)**
	73 Watts



Putting it all Together

- Assume interconnect power drops linearly with bandwidth
  - 68% of the performance for 25% of the interconnect power
- Total power for ¼ bandwidth = 61 Watts (down from 73 watts)
  - 68% of the performance for 83.6% of the system power

- Total Energy for two cases assuming full bandwidth runtime of X

$$\text{Energy}_{\text{full}} = 73X \qquad \text{Energy}_{1/4} = 1.32X * 61 = 80.5X$$

$$\frac{\text{Energy}_{1/4}}{\text{Energy}_{\text{full}}} = \frac{80.5X}{73X} = 1.10$$

- Net energy increase of 10% for ¼ bandwidth case
  - Keep in mind this doesn’t count the energy used for the file system attached to the machine or other machine room costs

Application Case Study: POP

- POP is an ocean circulation model derived from earlier models of Bryan, Cox, Semtner and Chervin in which depth is used as the vertical coordinate. The model solves the three-dimensional primitive equations for fluid motions on the sphere under hydrostatic and Boussinesq approximations.

- POP sends small-ish messages (one run showed 16KB average message size) and spends large portions of it's MPI time in MPI_Allreduce (at large node counts)

- POP is generally believed to be a latency and/or message rate bound application

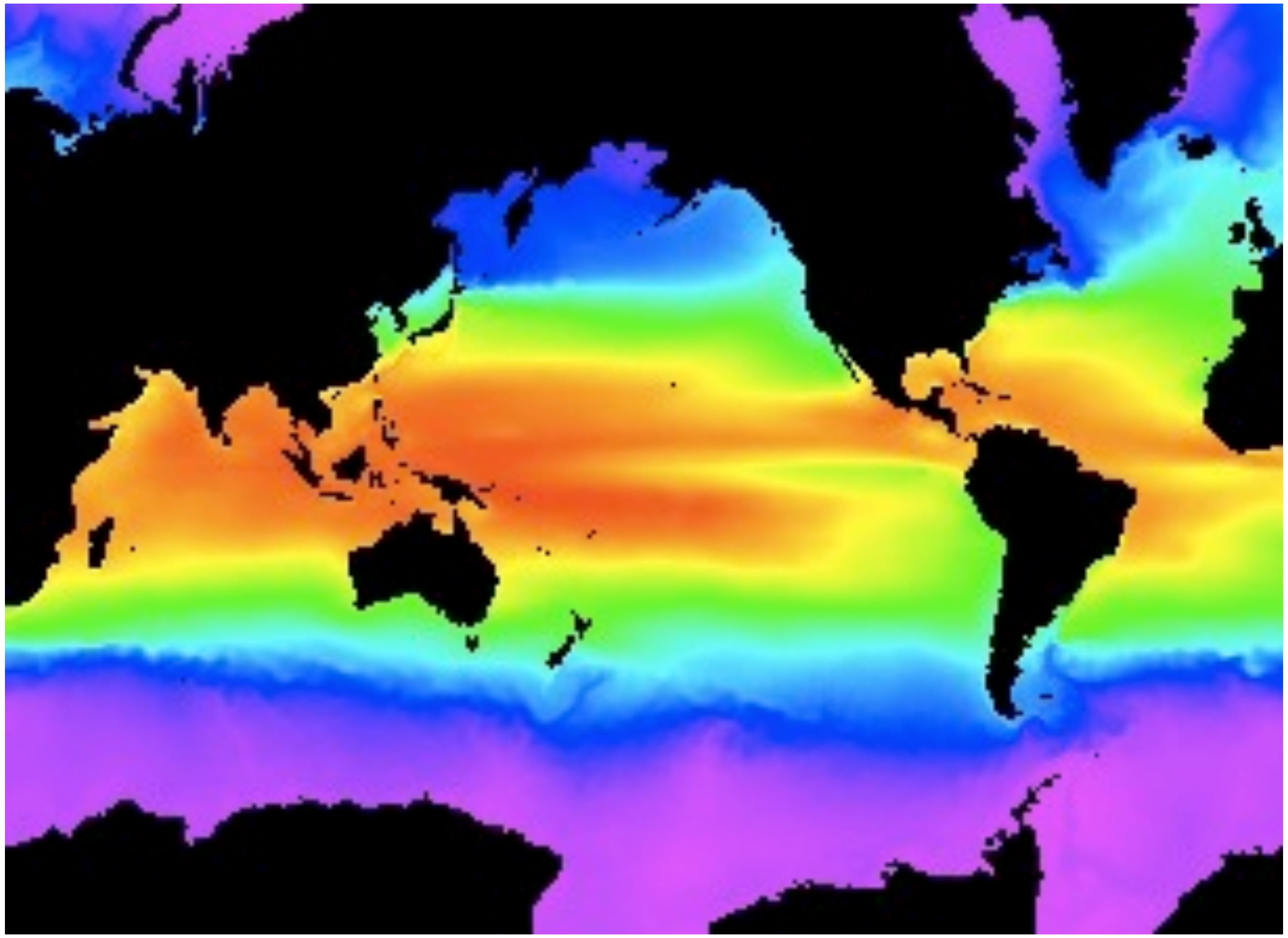
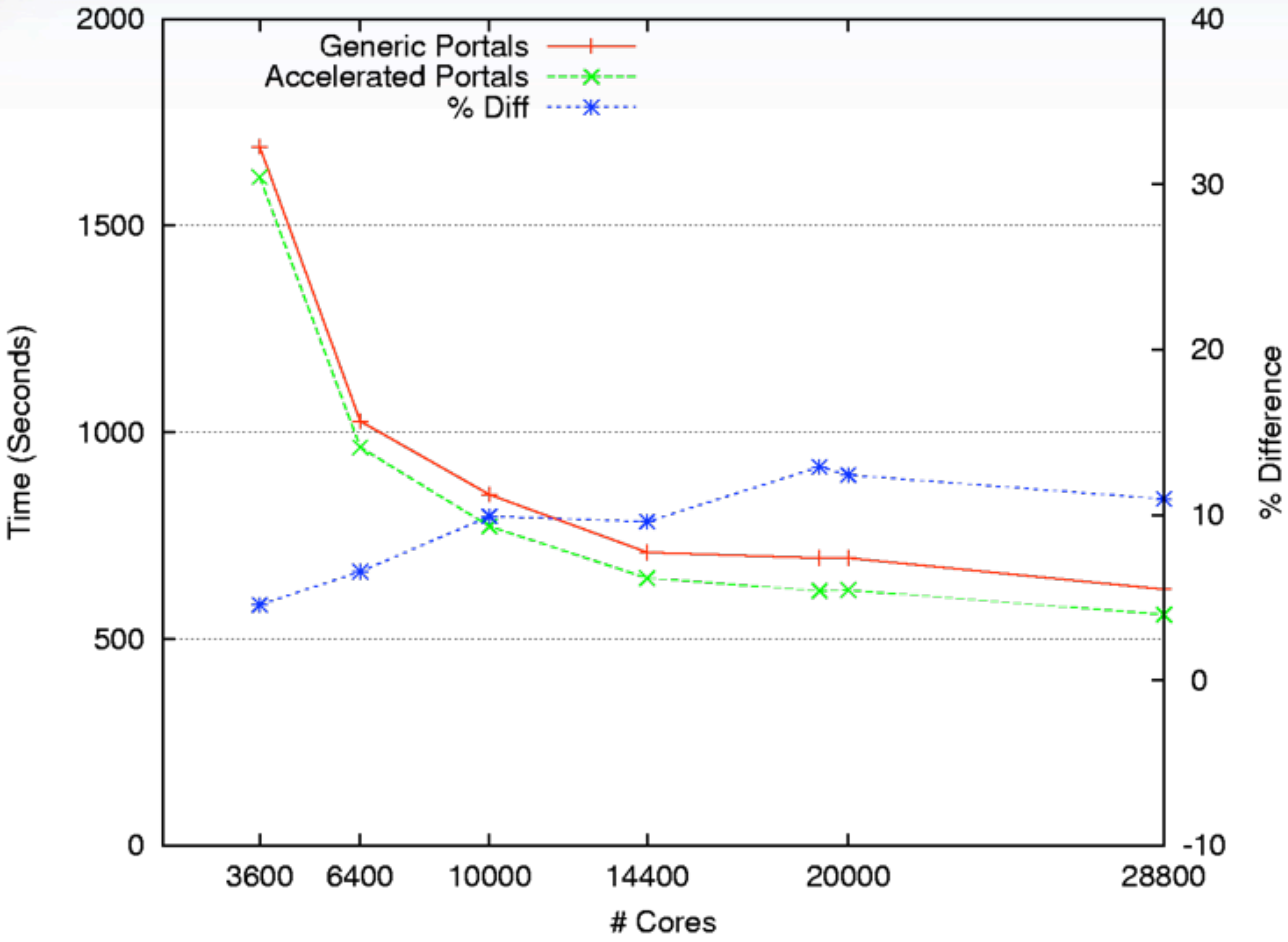
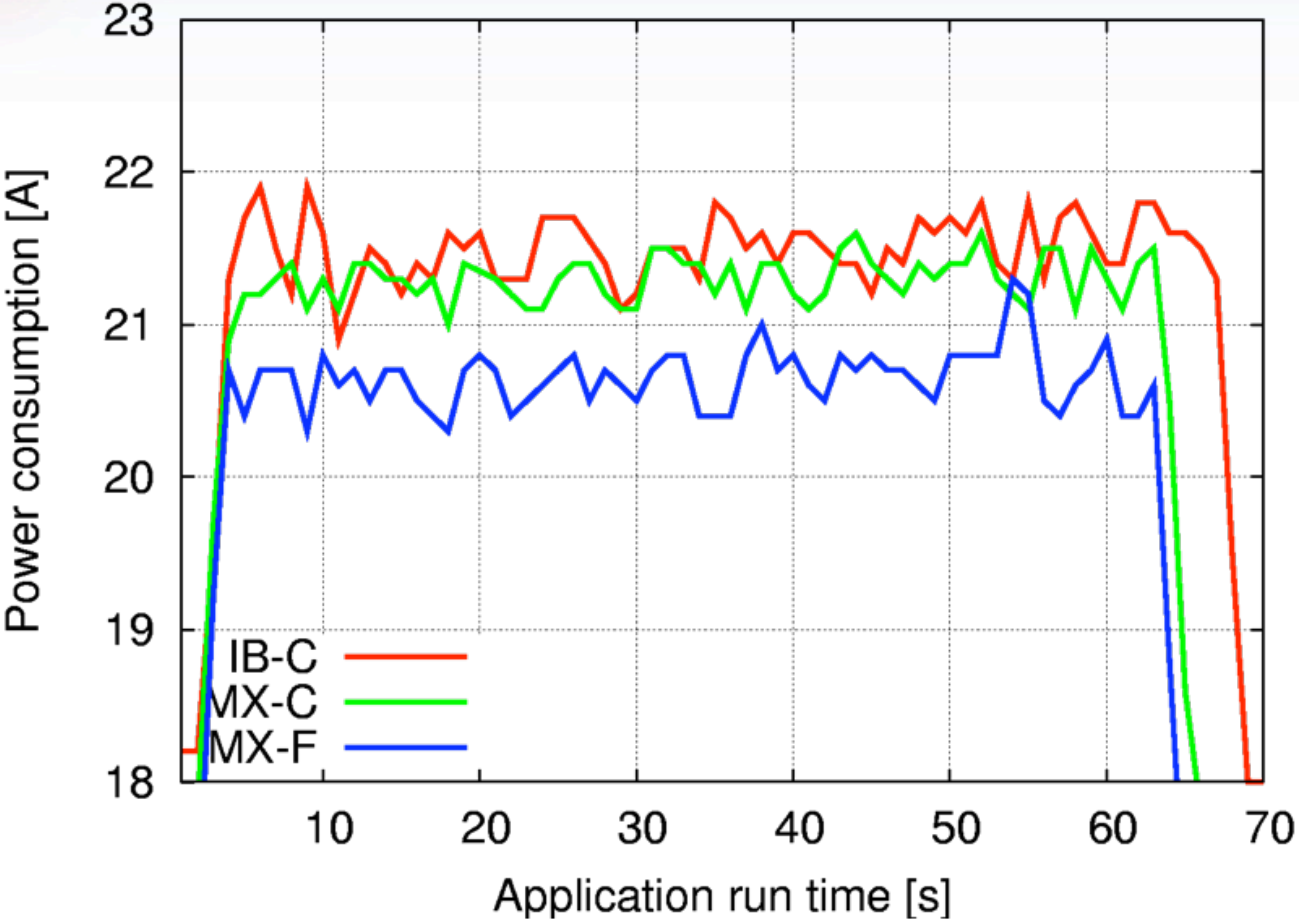


Image and description from http://www.lanl.gov/orgs/t/t3/codes/pop.shtml

Red Storm GP vs AP

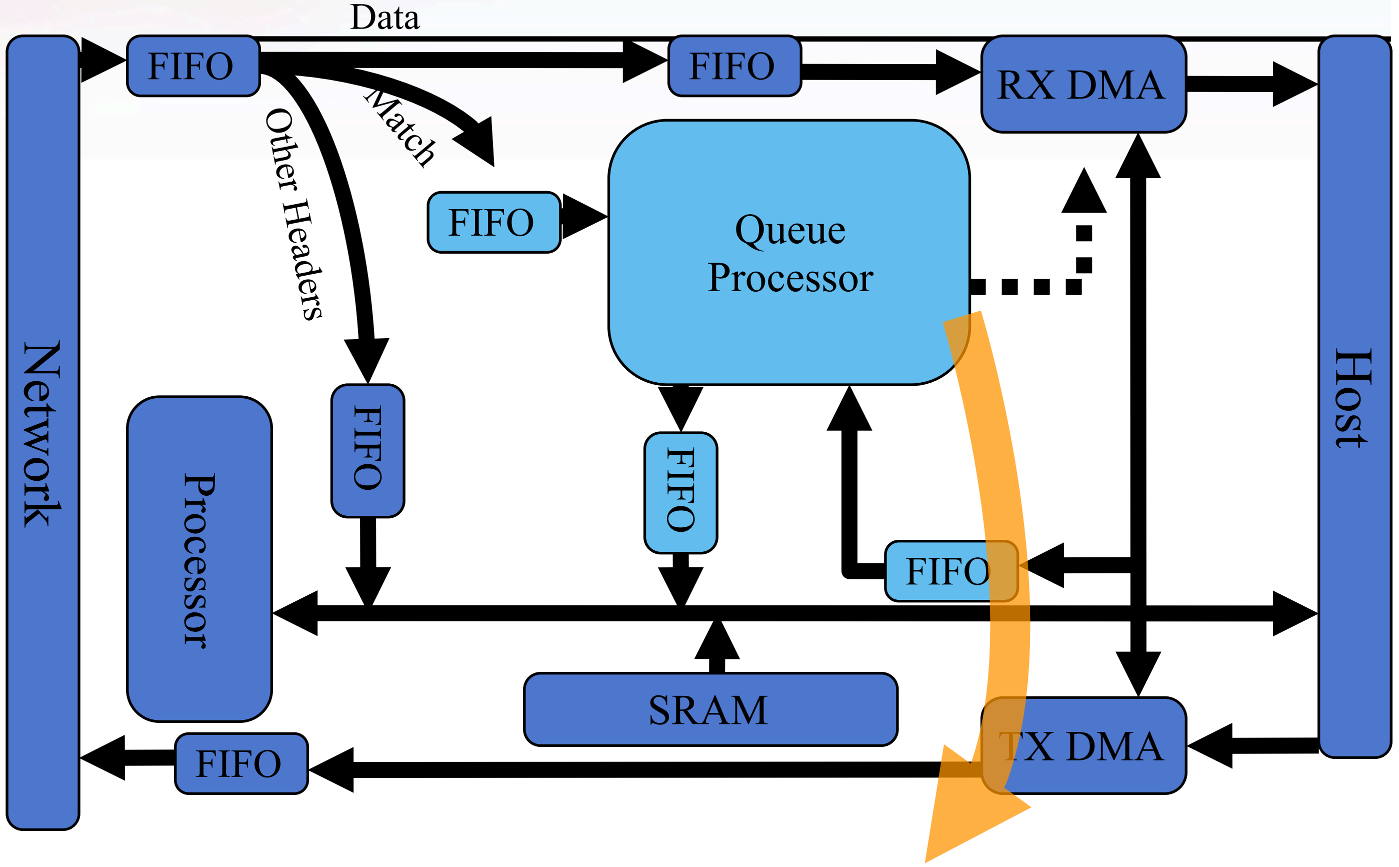


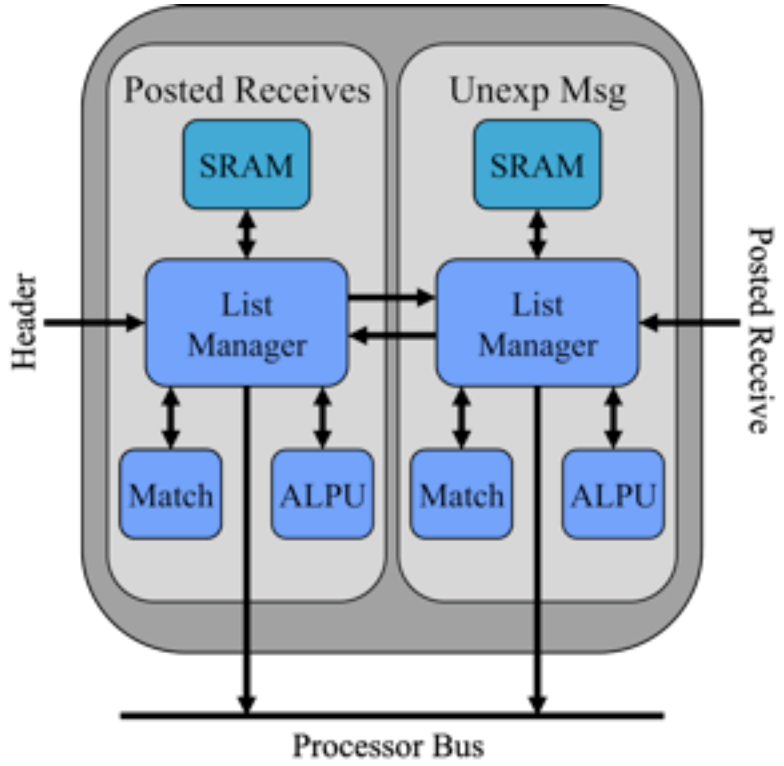
Power Study from Indiana University



Torsten Hoefler, Timo Schneider and Andrew Lumsdaine, “A Power-Aware, Application-Based Performance Study of Modern Commodity Cluster Interconnection Networks.” To appear in IPDPS/CAC09, May 2009.

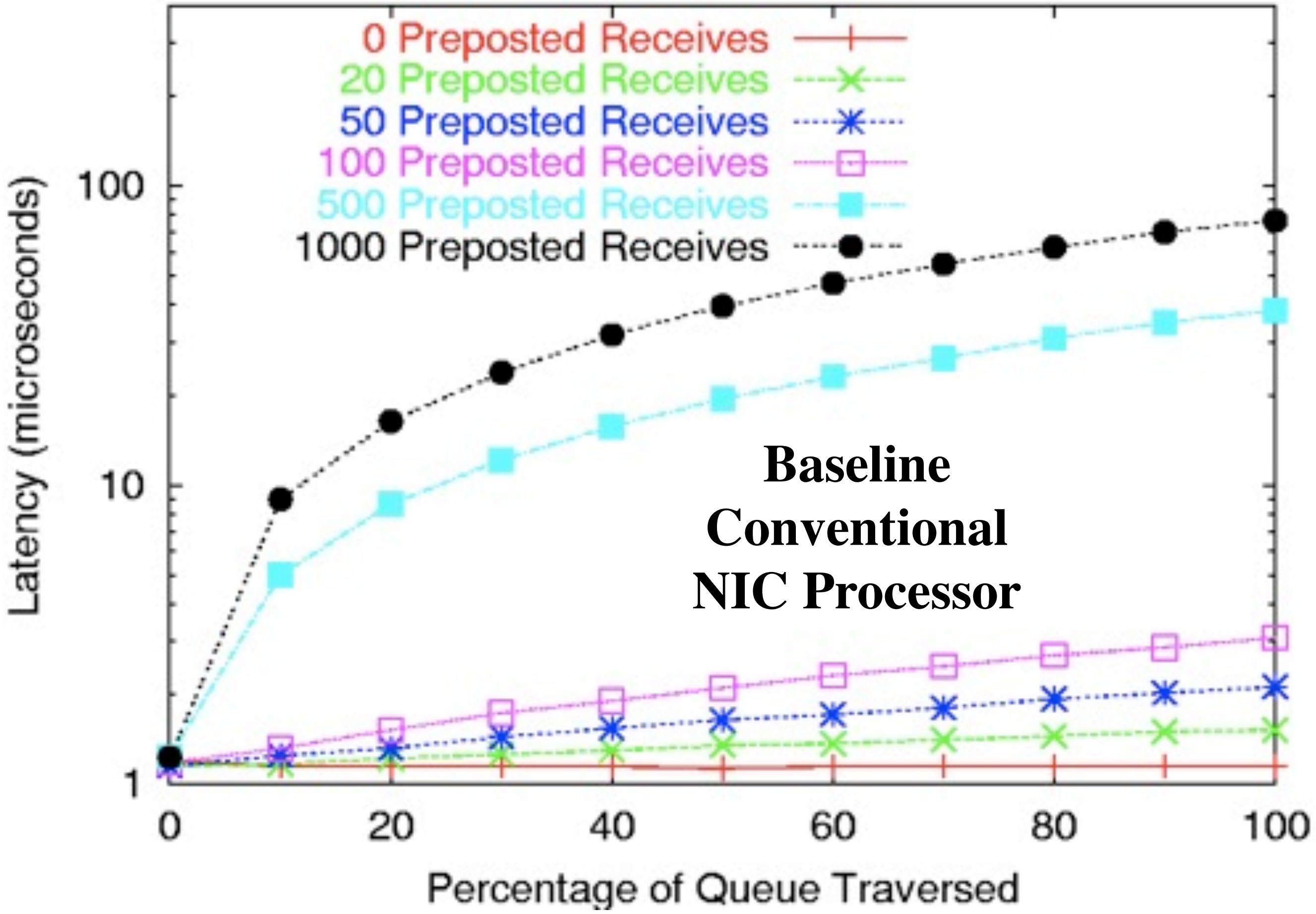
Solving the Message Rate Problem

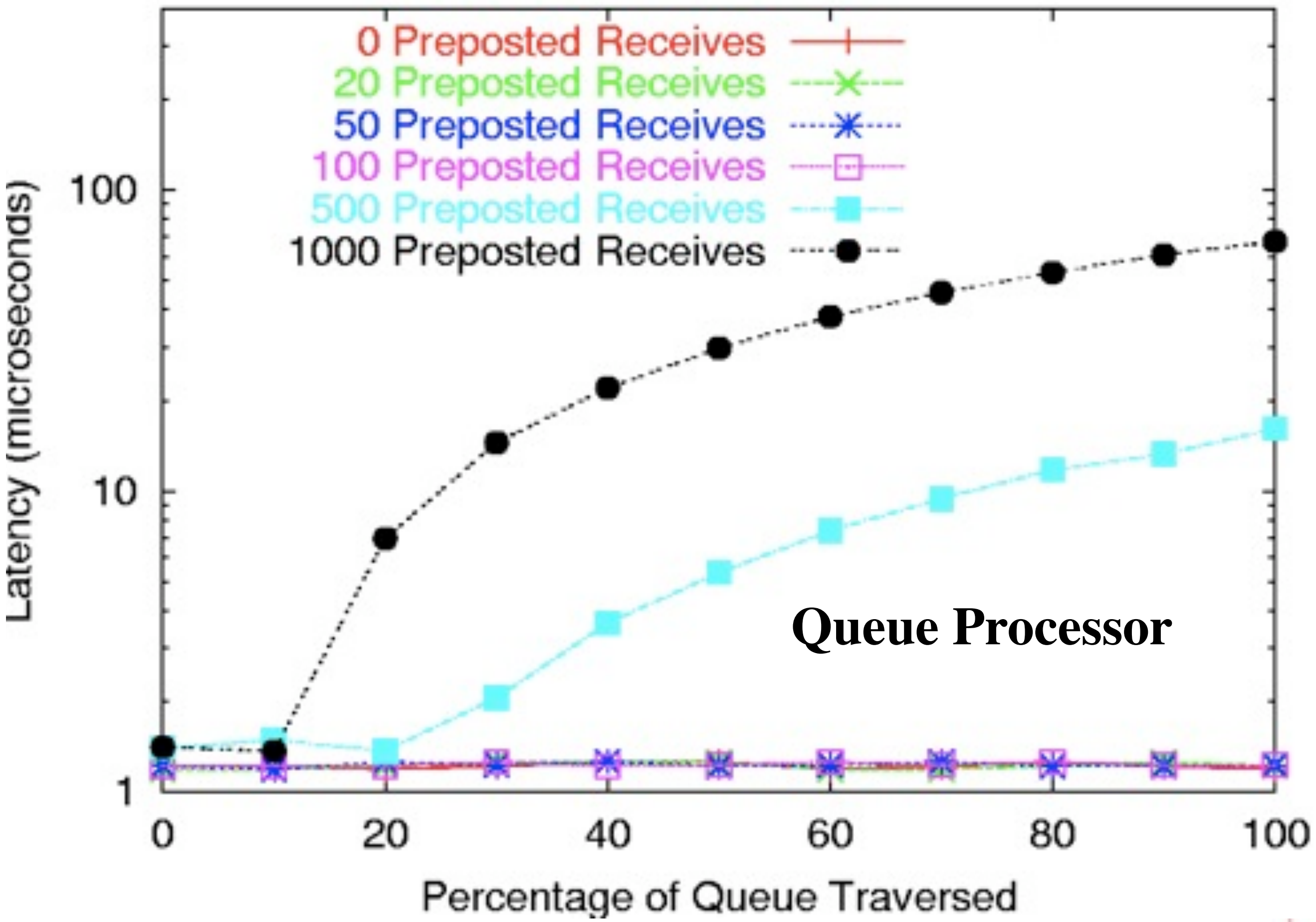


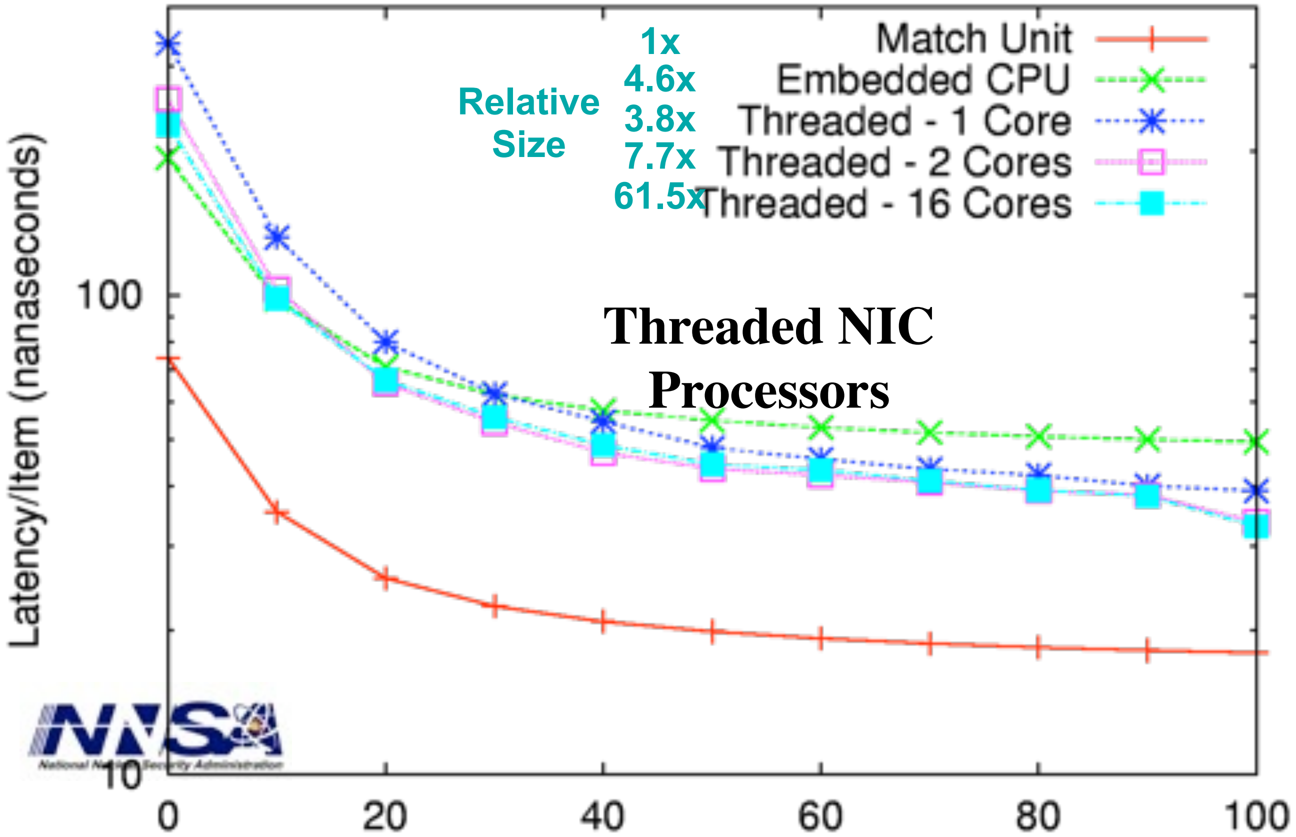


- Problem: Message rate determines <u>effective</u> bandwidth
- SST Analysis: MPI matching limits performance
- Solution: Accelerate list management with hardware
- SST Simulation
  - Hardware
    - Implement NIC/Router based on RedStorm SeaStar
    - Implement List Manager, ALPU, and Match Unit, integrate with NIC
      - "Translated" from FPGA RTL
    - Validate against FPGA & Red Storm
  - Software
    - Create baseline offload MPI
    - Modify MPI to use acceleration HQ
- Impact: Collaboration w/ Intel
  - Intel licensing
  - Keith Underwood "on loan" to Intel
  - Foundation for IAA network project

Dramatic MPI Acceleration







- Long MPI message queues increase effective latency dramatically
- Queue processor processes messages more quickly
- Queue processor more area effective than conventional or threaded NIC processor

Sandia National Laboratories

System View is Vital

- First example showed a case where higher interconnect power leads to lower energy to solution
- Second example illustrates how advanced features which add very little to system power can improve performance, thus improving energy to solution
- The system view is critical
  - Interconnect is not an isolated system and only accounts for a portion of the total system power
  - Thus, higher interconnect power can actually lead to lower energy
  - Understanding the true impact of the interconnect trade-offs can lead to more energy efficient systems

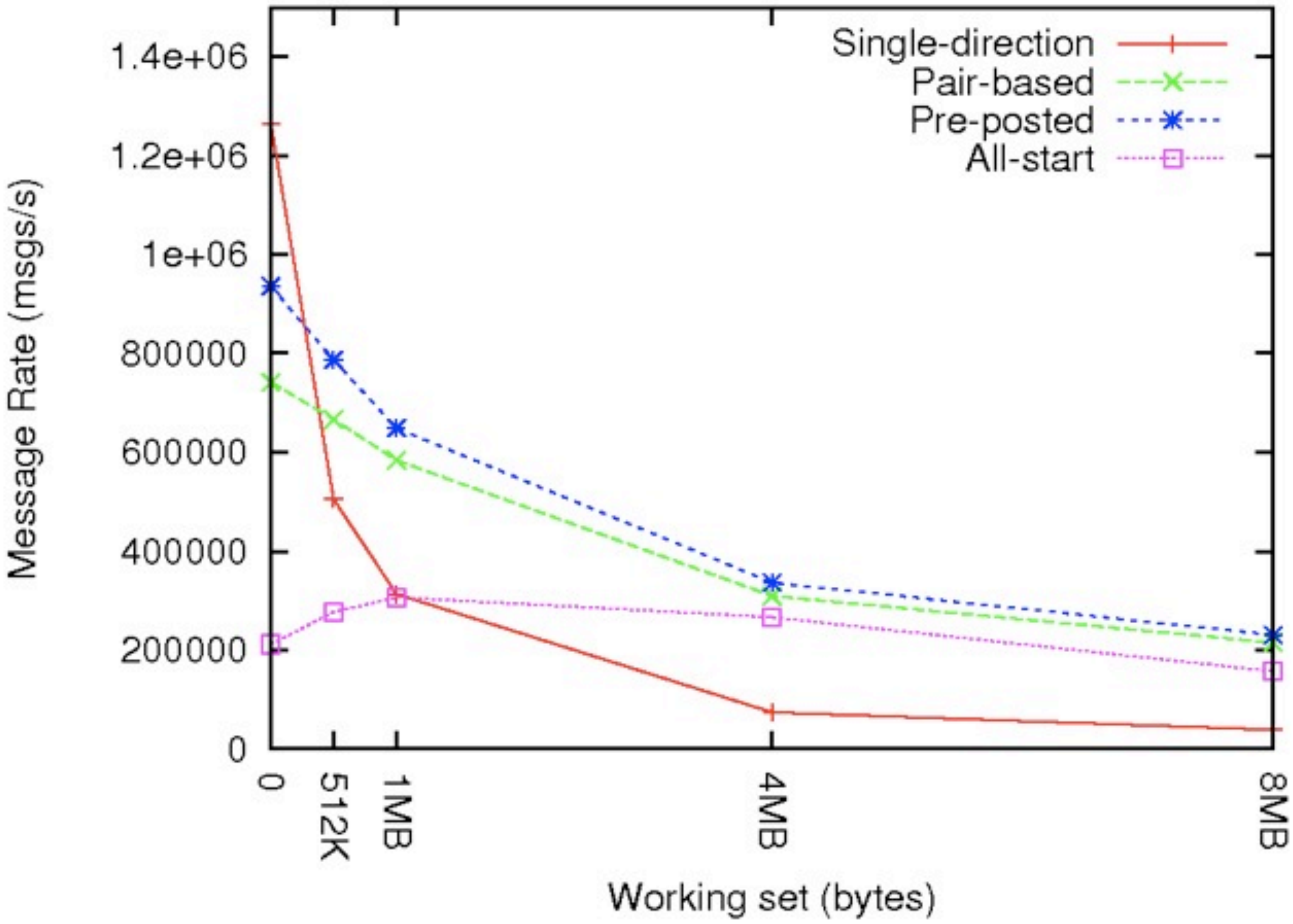
Microbenchmarks

- **Fallacy: Optimizing interconnects and MPI implementations to microbenchmarks will necessarily improve application performance (or at least won't hurt it).**
- **Any optimization that reduces performance without reducing power will lead to less energy efficient system**
  - **Conversely, any optimization that increases performance without increasing power will lead to more energy efficient systems**

- **Removing useful advanced features to improve NetPipe latency and bandwidth will not generally translate to improved application performance (and may actually make it worse)**
- **Coalescing identical zero-byte messages will not help any application of which I am aware**
- **Measuring message rate under ideal conditions does not provide useful information about message rate achievable by an application**

Sandia Message Throughput Benchmark

- **Measures message rate using communication patterns mimicking those of scientific applications**
  - **Simulation of computation/communication phase with variable working set sizes (compute stage modeled by touching data to invalidate some portion of cache)**
  - **Each MPI rank both sends and receives**
  - **Variable number of peers**

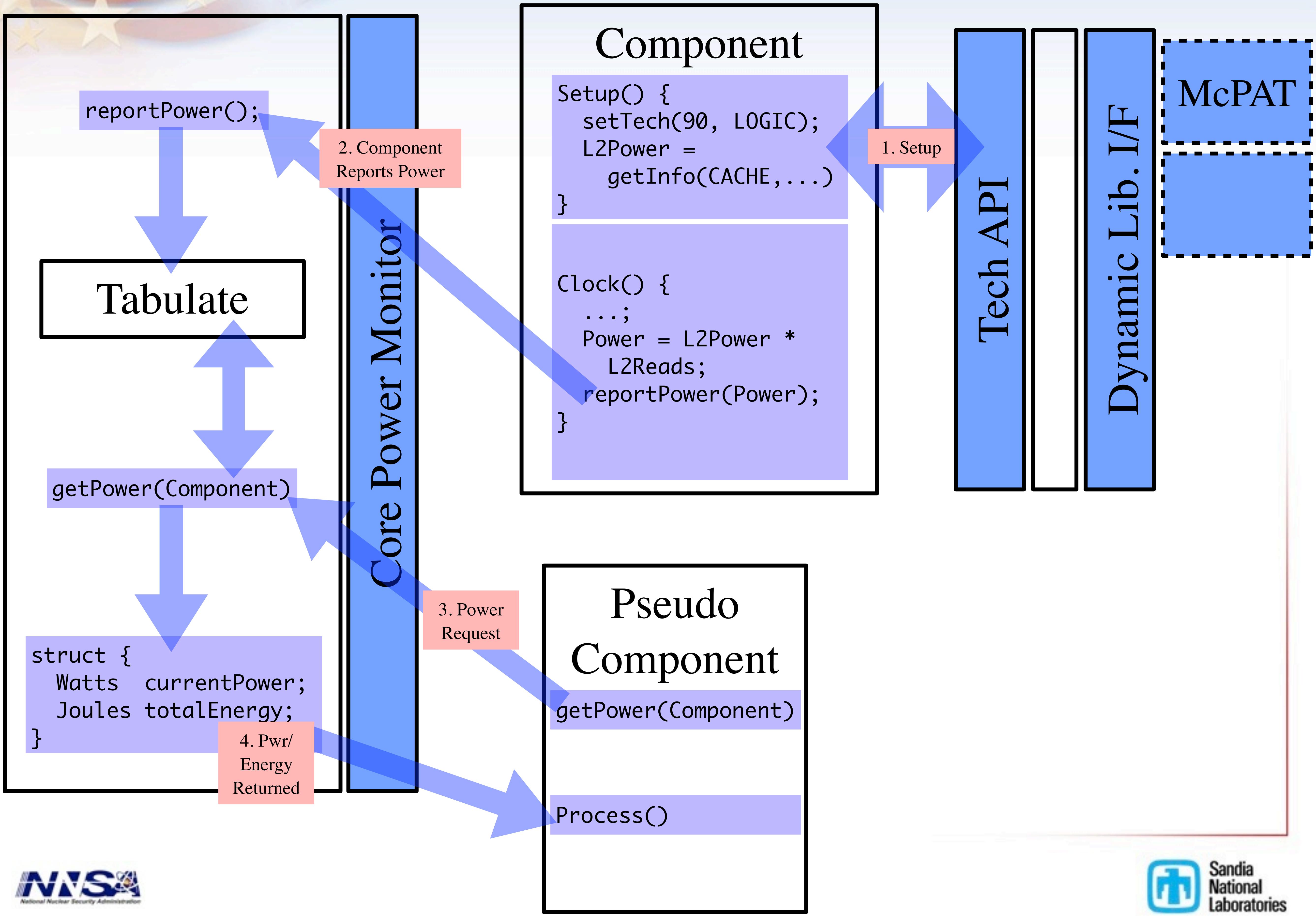


Conclusions

- It’s not necessarily about power, it’s about energy to solution
  - Higher power systems can actually lead to lower energy to solution
  - When peak power is a limiter, likely better off with a “smaller”, more balanced system, than a larger, unbalanced system
- It’s not about peak FLOPS/Watt, it’s about the percent of peak that can be sustained
  - We pay an energy penalty for unused operations
  - With rising awareness of energy-efficient computing, FLOPS/Watt threatens to become the new HPL. Let’s not let this happen!
- This talk focused on interconnects, but other areas are equally important
  - What’s the application impact of slower, less complex cores
    - Can in-order cores use wide floating-point units?
    - Can applications scale to the dramatically increased number of cores?
- Components should be designed with a system view and understanding of the application needs

Bonus

Model of Operation 2



```
reportPower();
```

Tabulate

```
getPower(Component)
```

```
struct {
  Watts  currentPower;
  Joules totalEnergy;
}
```

Core Power Monitor

Component

```
Setup() {
  setTech(90, LOGIC);
  L2Power =
    getInfo(CACHE,...)
}
```

```
Clock() {
  ...;
  Power = L2Power *
    L2Reads;
  reportPower(Power);
}
```

Pseudo Component

```
getPower(Component)
```

```
Process()
```

1. Setup

2. Component Reports Power

3. Power Request

4. Pwr/ Energy Returned

DES APIs

Key API Calls

Construction

```
Constructor(map<str,str> param);
```

Component

```
serialize(Archive &a);
addLink(int link_num, Handler);
ClockRegister(Freq, Handler);
handleClock();
recvHander(event *e);
```

Link

```
Send(Time_t t, Event *e);
Event* Recv();
Link(latency);
```

Event

```
arrivalTime();
source();
destination();
```

Current Strawman

User Defined

Simulation::Run()

```
while(1) {
  cycle++;
  foreach component {
    component->preTic();
  }

  while(event =
        getNextEventThisClock(queues)){
    event->component->handleEvent();
  }
}
```

Current Strawman

```
while(event = getNextEvent(queues)) {
  cycle = event->time;
  if (event->isClockEvent) {
    event->component->preTic();
    reschedule(event, queue);
  } else {
    event->component->handleEvent();
  }
}
```

Future ParallelProto

- **Current strawman**
  - **–Advances through each cycle**
  - **–No Opt-out mechanism for clock**
  - **–No easy support for multiple clock domains**
- **Future Parallel Proto**
  - **–Clocks and communication events all in queue**
  - **–Skip event-less cycles**
  - **–Clock events special cased**

Parallel

```
while(event = getNextEvent(queues)) {
  cycle = event->time;
  if (event->isClockEvent) {
    event->component->preTic();
    reschedule(event, clockPeriod);
  } else if (event->exchange) {
    startSends();
    recv();
    finishSends();
    reschedule(event, minPartition);
  } else if (event->checkpoint) {
    checkpoint();
    reschedule(event, checkPointTime);
  } else {
    event->component->handleEvent();
  }
}
```

- **Checkpointing and message exchange are also queued events**
- **Possible optimization: different exchange times for each neighbor, based on partition**

Sending an Event

Calculate arrival time

(may include BW calculation)

```
void compLink::sendEvent(event *e) {
  e->_arrivalTime = theSim->cycle + latency;
  e->_src = source;
  e->_dest = dest;
  if (destination is local) {
    theSim->eventQ.push_back(e);
  } else {
    theSim->
      remoteEvents[destRank].push_back(e);
  }
}
```

Queue local events

Store remote events

(To send later)