Toward Integrated Multi-Resolution HPC Modeling for Rapid Performance Prediction

Jason Liu

Florida International University

Stephan Eidenbenz

Los Alamos National Laboratory

Workshop on Modeling & Simulation of Systems and Applications
August 10-12, 2016 ♦ University of Washington, Seattle, Washington

HPC Architecture Is Changing Rapidly

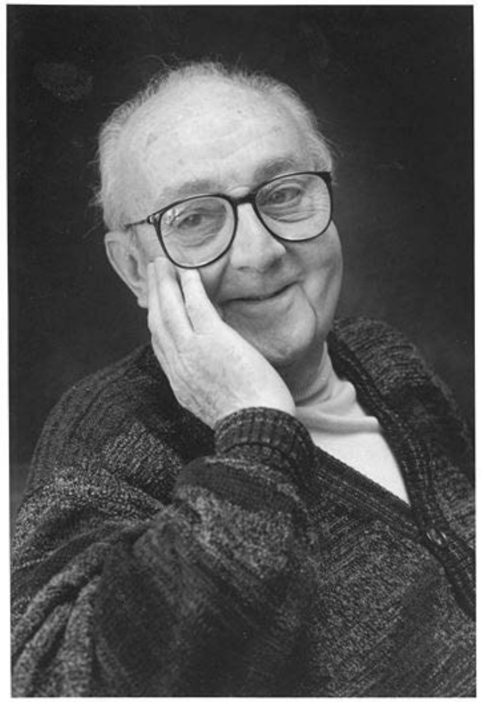
- End of processor scaling (circa 2005) led to novel architectural design
    - Changes can be transitional and disruptive
- HPC software adaptation is a constant theme:
    - *No code is left behind*: must guarantee good performance
    - Need high-skilled software architects and computational physicists
- Traditional methods are insufficient
    - Middleware libraries, code instrumentation, mini-apps…
- **Need modeling & simulation of large-scale HPC systems and applications**
    - And the systems are getting larger (exascale is around the corner)

Full-Scale Cycle-Accurate Simulation of HPC Systems and Applications

- It is unrealistic
  - Extremely high computational and spatial demand
  - Accurate models only limited to certain components and timescale
- And it is unnecessary
  - Modeling uncertainties greater than errors from cycle-accurate models
    - Languages, compilers, libraries, operating systems, …
    - System cross traffic
  - Design uncertainties defies specificity of cycle-accurate models
- **There has to be tradeoff between accuracy and performance**

“All models are wrong but some are useful”

- George Box, 1976



- Managing expectations:
  - Ask what-if questions
  - Evaluate alternative designs
  - Explore parameter space
- Will models ever catch up with real-system refresh?
  - As valuable tools for prototyping new hardware, new algorithms, new applications?
- **Need tools for rapid assessment and performance prediction**

Requirements of Rapid Assessment and Performance Prediction

- Easy integration of models of varying abstraction
- Easy integration of architectures and applications
- Short development cycle
- Performance and scale

Selective Refinement Modeling

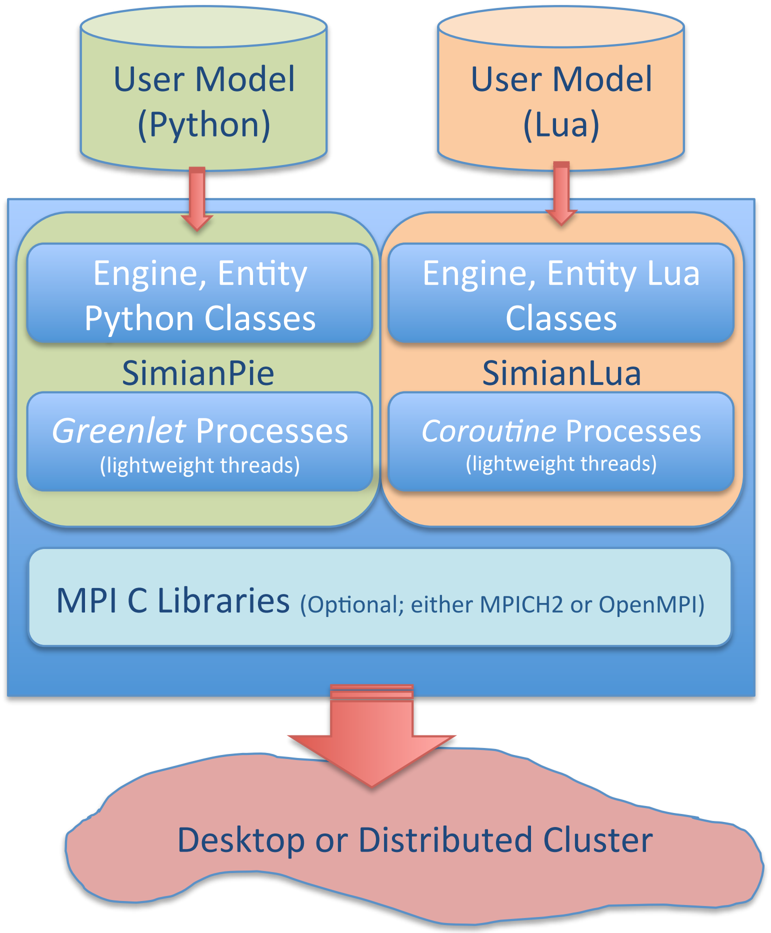
- The art of finding the "right" level of modeling details
  - Just enough to answer the research questions
- Conceptually it is an iterative process:
  ① Start from coarse-level models
  ② Gather experiment results
  ③ Identify components as potential performance bottlenecks
  ④ Replace those components by plugging in more refined models
  ⑤ Go to #2 until satisfaction

Performance Prediction Toolkit (PPT)

- "Scalable Codesign Performance Prediction for Computational Physics" project at LANL
  - **Simian** – parallel discrete-event simulation engine
  - **Configurable hardware models**: clusters, compute nodes, processes/cores, accelerators (GPU), interconnect, parallel file systems
  - **Application library**: benchmark applications (PolyBenchSim, ParboilSim), production applications (SNAPSim, SPHSim, SpecTADSim)
  - **Data**: application instrument data (PolyBench, SNAP, SPH, CloverLeaf), hardware specs data (Mustang, Haswell, IvyBridge, SandyBridge, Vortex)

Simian: Parallel Discrete Event Simulation using Interpreted Languages

- Open source, general purpose parallel discrete-event library
- Independent implementation in two interpreted languages: Python and Lua, with optional C libraries (such as MPI)
- **Minimalistic design: LOC=500 with 8 common methods**
- Simulation code can be Just-In-Time (JIT) compiled to achieve very competitive event-rates, even outperforming C++ implementation in some cases
- Support process-oriented world view (using Python greenlets and LUA coroutines)


User Model (Python)
User Model (Lua)
Engine, Entity Python Classes
Engine, Entity Lua Classes
SimianPie
SimianLua
Greenlet Processes (lightweight threads)
Coroutine Processes (lightweight threads)
MPI C Libraries (Optional; either MPICH2 or OpenMPI)
Desktop or Distributed Cluster

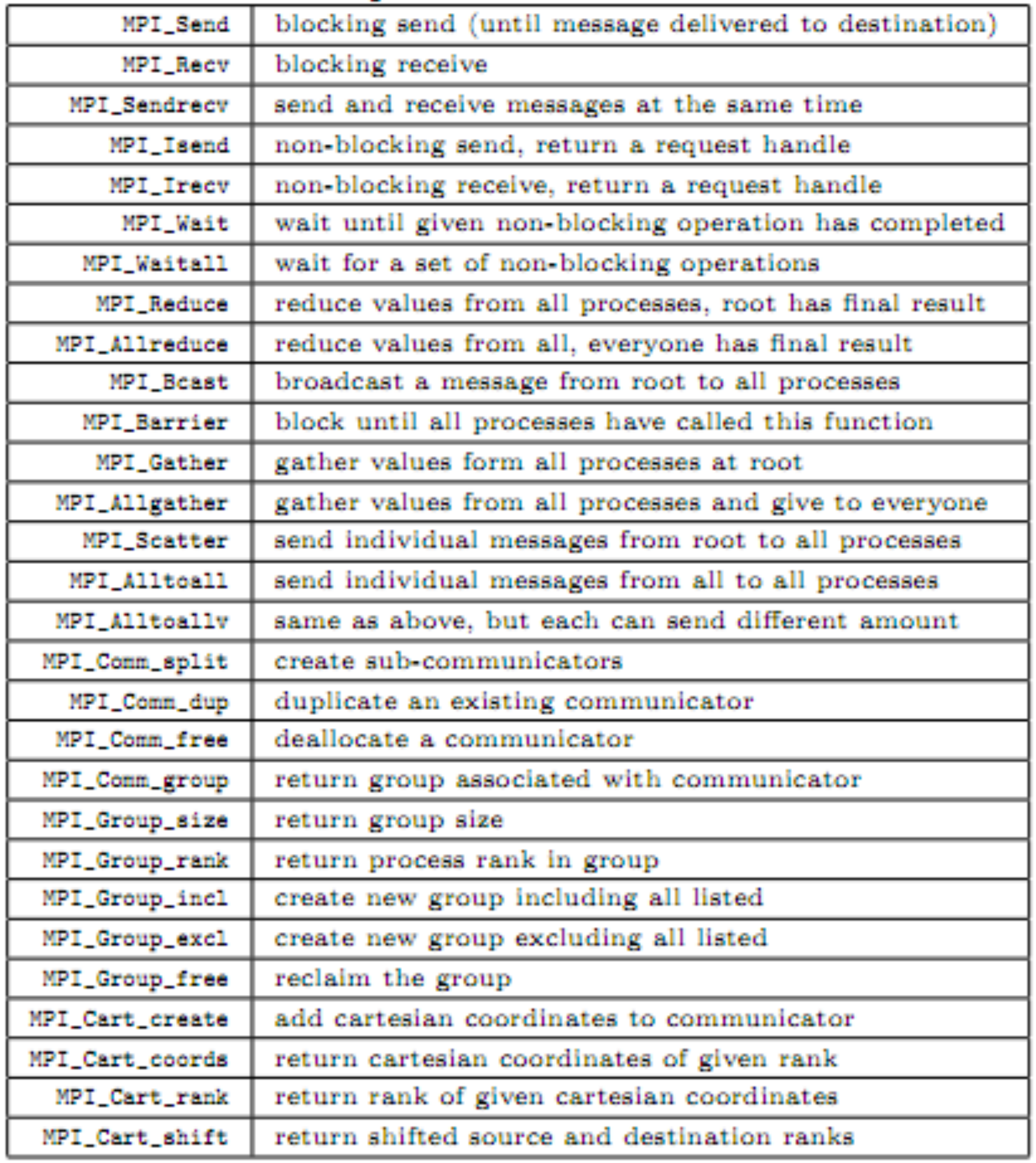
Interconnection Network Models

- Common interconnect topologies
  - Torus (Gemini, Blue Gene/Q)
  - Dragonfly (Aries)
  - Fat Tree (Infiniband)
- Distinction from previous approaches:
  - BigSim, xSim, SST, CODES
- Emphasis on production systems
  - Cielo, Darter, Edison, Hopper, Mira, Sequoia, Stampede, Titan, Vulcan, …
- Packet-level as opposed to phit-level
  - For performance and scale (speed advantage in several orders of magnitude, allow for full scale models, sufficient accuracy)
- Seamlessly integrated with MPI

Integrated MPI Model

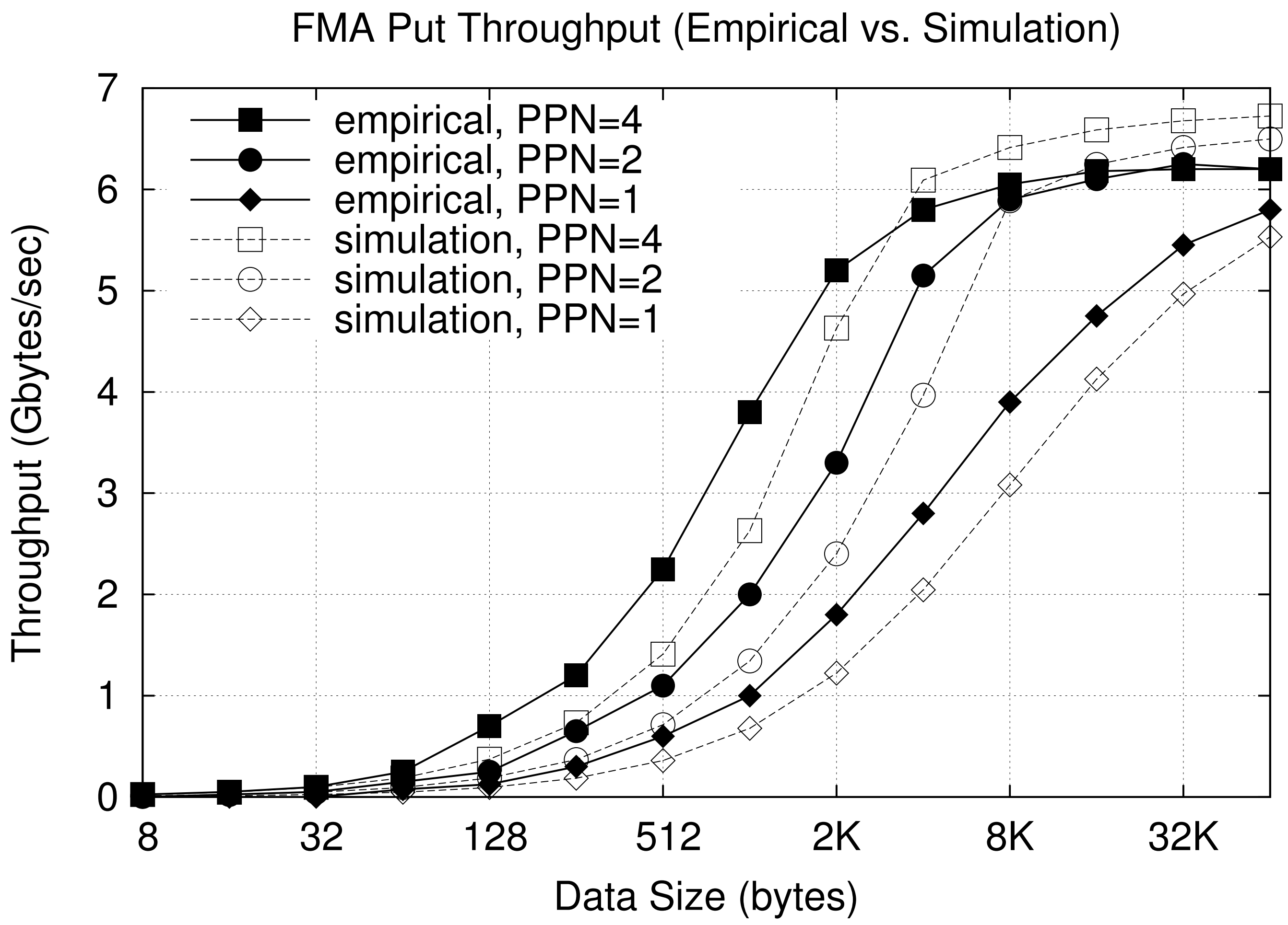
- Developed based on Simian (entities, processes, services)
- Include all common MPI functions
  - Simplistic implementation
  - Point-to-point and collective operations
  - Blocking and non-blocking operations
  - Sub-communicators and sub-groups
- Process-oriented approach
  - Can easily integrate with most application models
- Packet-oriented model
  - Large messages are broken down into packets (say, 64B)
- Reliable data transfer

Table 1: Implemented MPI Functions



MPI_Send	blocking send (until message delivered to destination)
MPI_Recv	blocking receive
MPI_Sendrecv	send and receive messages at the same time
MPI_Isend	non-blocking send, return a request handle
MPI_Irecv	non-blocking receive, return a request handle
MPI_Wait	wait until given non-blocking operation has completed
MPI_Waitall	wait for a set of non-blocking operations
MPI_Reduce	reduce values from all processes, root has final result
MPI_Allreduce	reduce values from all, everyone has final result
MPI_Bcast	broadcast a message from root to all processes
MPI_Barrier	block until all processes have called this function
MPI_Gather	gather values form all processes at root
MPI_Allgather	gather values from all processes and give to everyone
MPI_Scatter	send individual messages from root to all processes
MPI_Alltoall	send individual messages from all to all processes
MPI_Alltoallv	same as above, but each can send different amount
MPI_Comm_split	create sub-communicators
MPI_Comm_dup	duplicate an existing communicator
MPI_Comm_free	deallocate a communicator
MPI_Comm_group	return group associated with communicator
MPI_Group_size	return group size
MPI_Group_rank	return process rank in group
MPI_Group_incl	create new group including all listed
MPI_Group_excl	create new group excluding all listed
MPI_Group_free	reclaim the group
MPI_Cart_create	add cartesian coordinates to communicator
MPI_Cart_coords	return cartesian coordinates of given rank
MPI_Cart_rank	return rank of given cartesian coordinates
MPI_Cart_shift	return shifted source and destination ranks

Validation – Cray's Gemini Interconnect



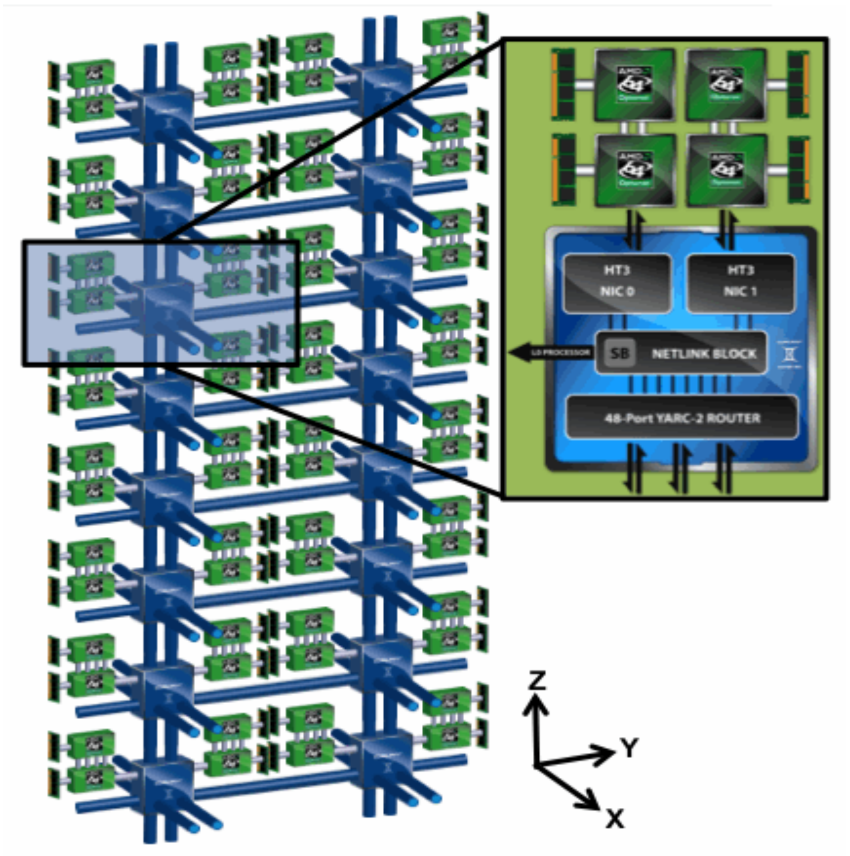
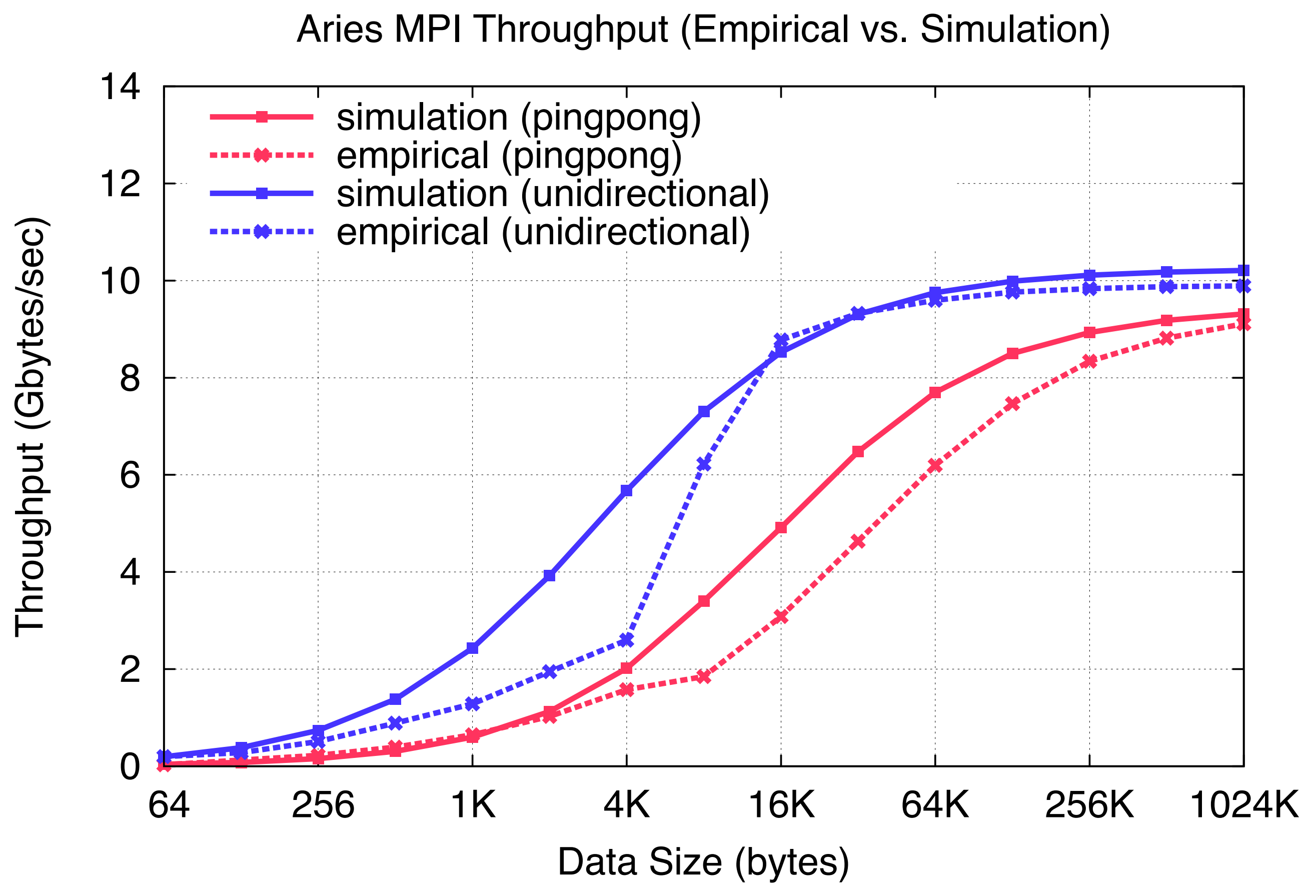
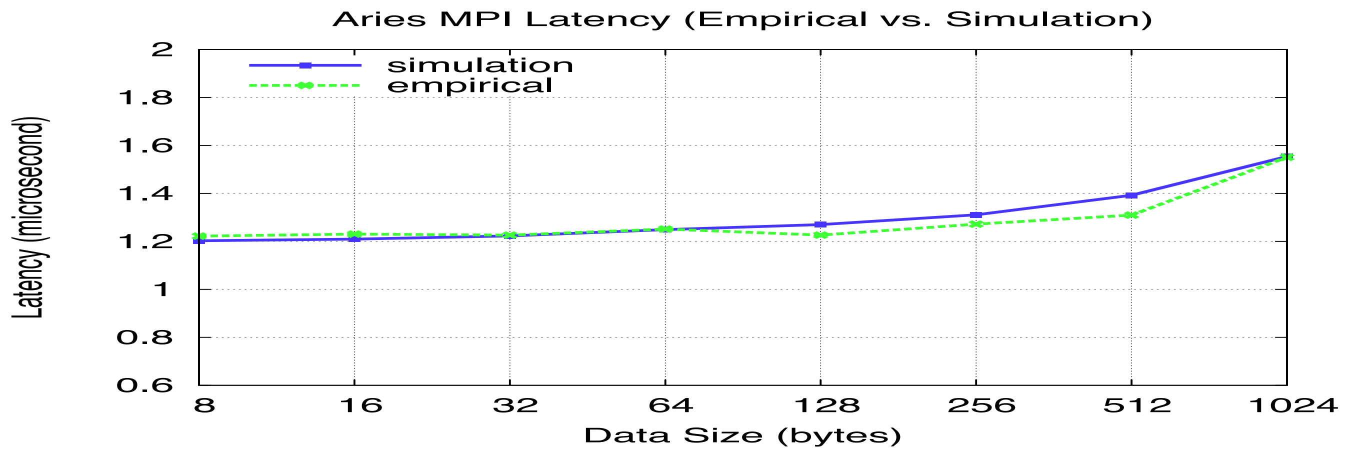


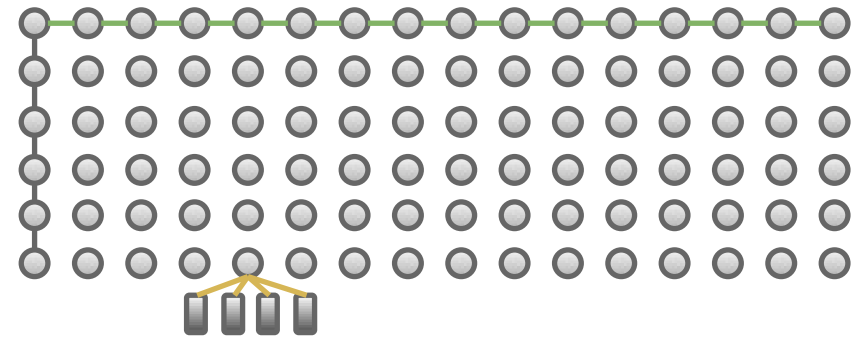
Image courtesy of Cray, Inc.

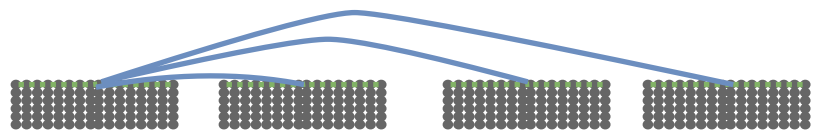
Gemini FMA put throughput (as reported in [2]) versus simulated throughput as a function of transfer size for 1, 2, and 4 processes per node.

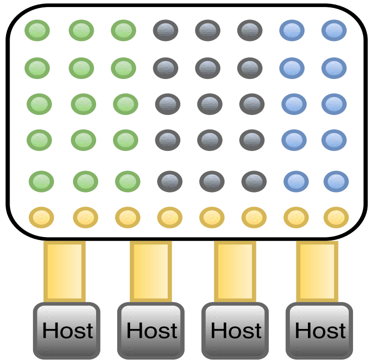
Validation - Cray's Aries Interconnect



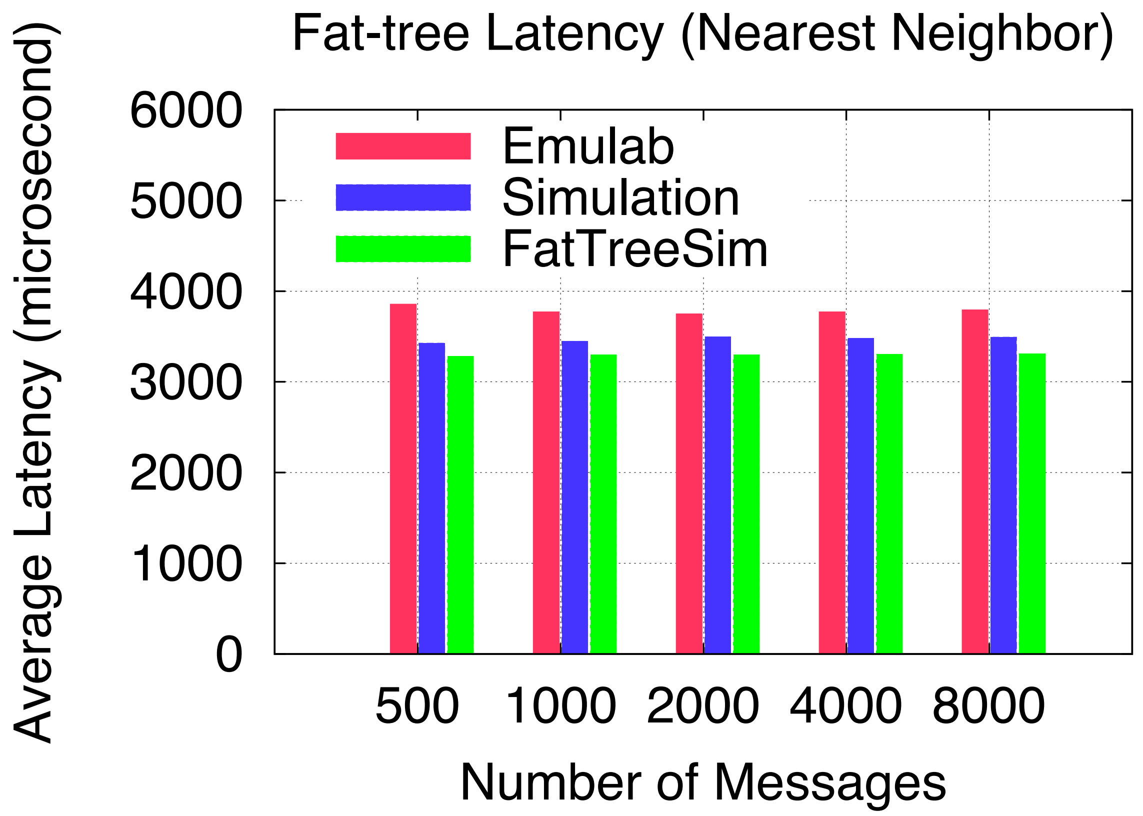


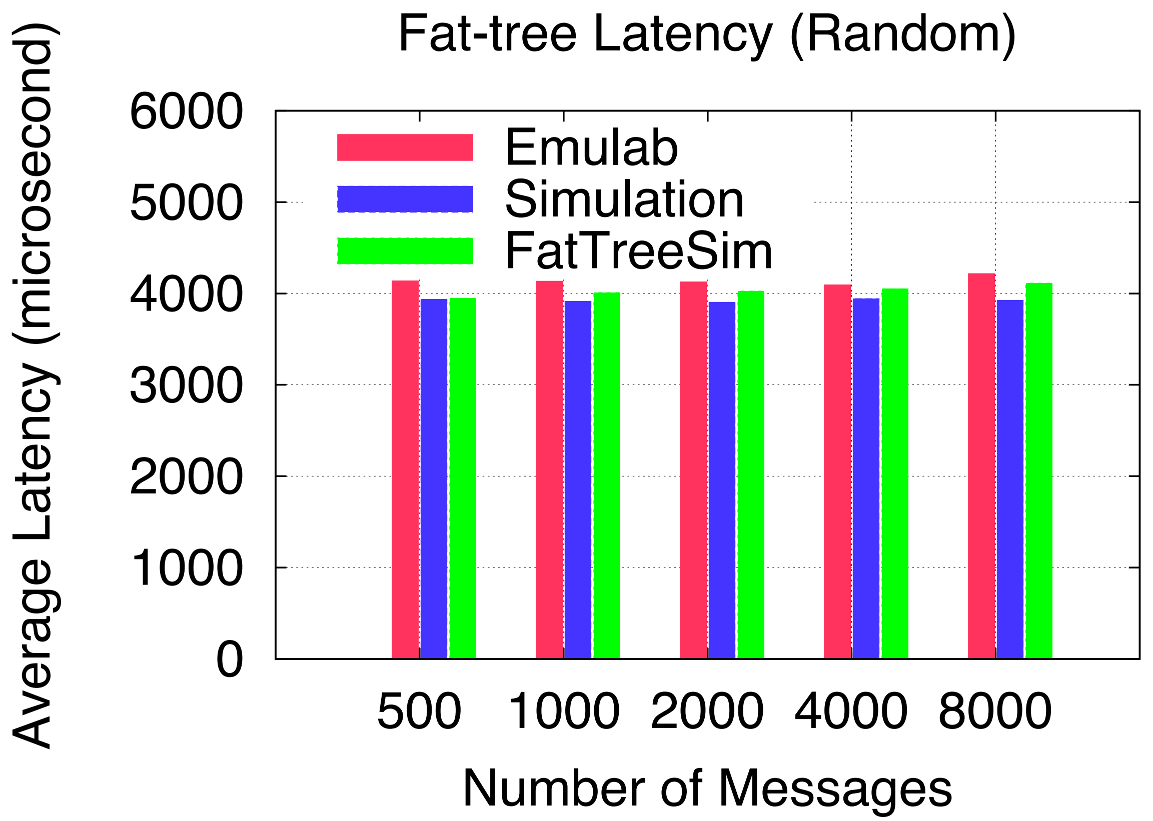






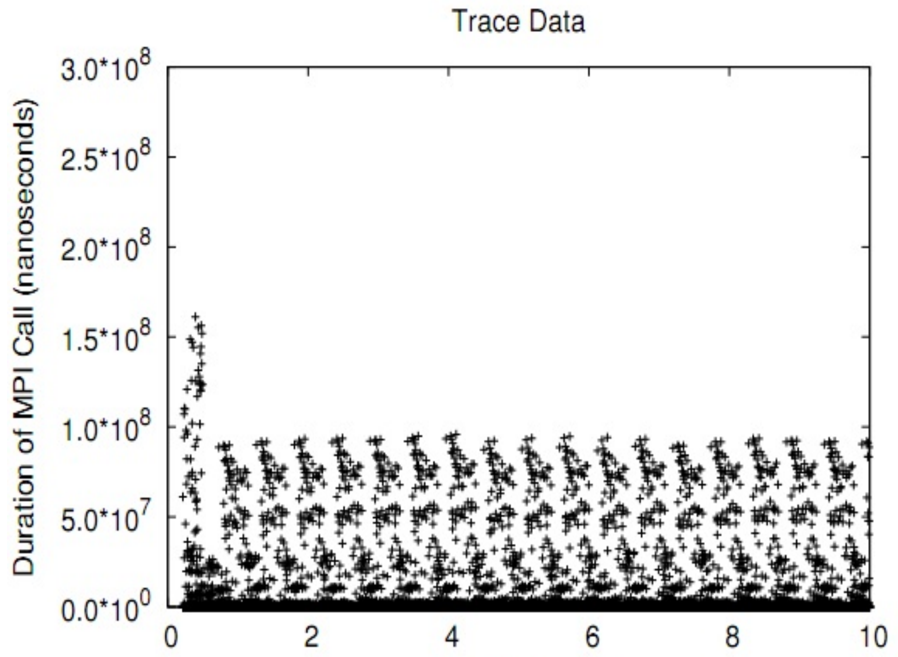
Validation – Fat-tree FDR Infiniband

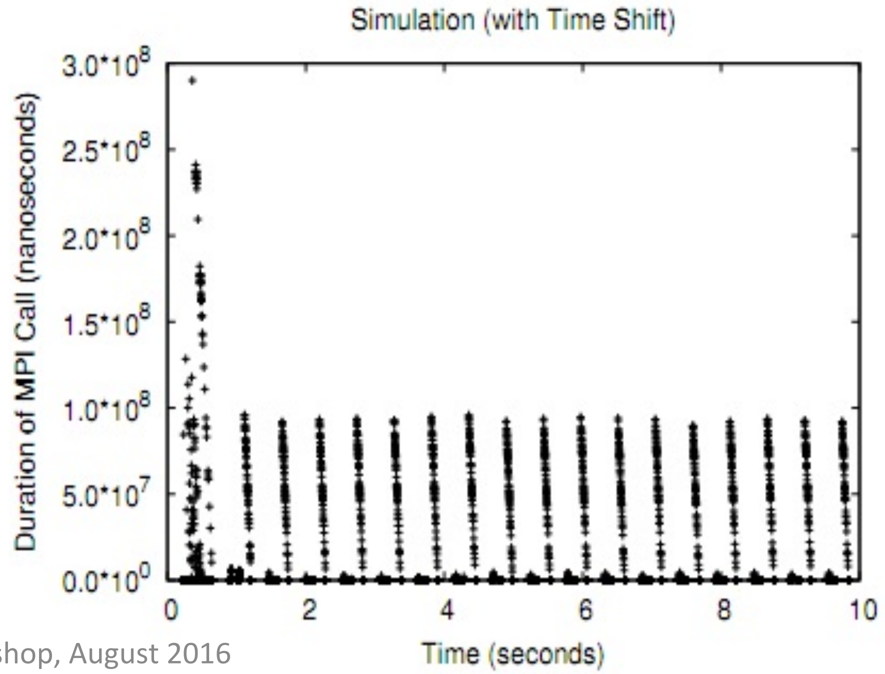




Trace Driven Simulation

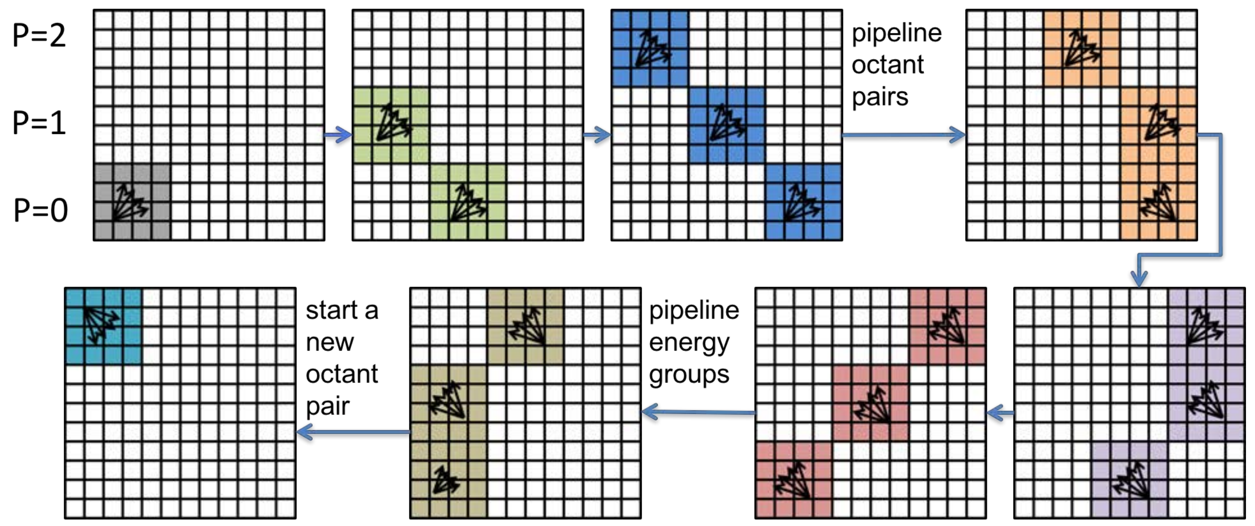
- Mini-app MPI traces:
  - Trace generated when running mini-apps on NERSC Hopper (Cray XE06) with <=1024 cores
  - Trace contains information of the MPI calls (including timing, source/destination ranks, data size, …)
- For this experiment, we use:
  - LULESH mini-app from ExMatEx
    - Approximates hydro-dynamic model and solves Sedov blast wave problem
  - 64 MPI processes
- Run trace for each MPI rank:
  - Start MPI call at exactly same time indicated in trace file
  - Store completion time of MPI call
  - Compare it with the completion time in trace file





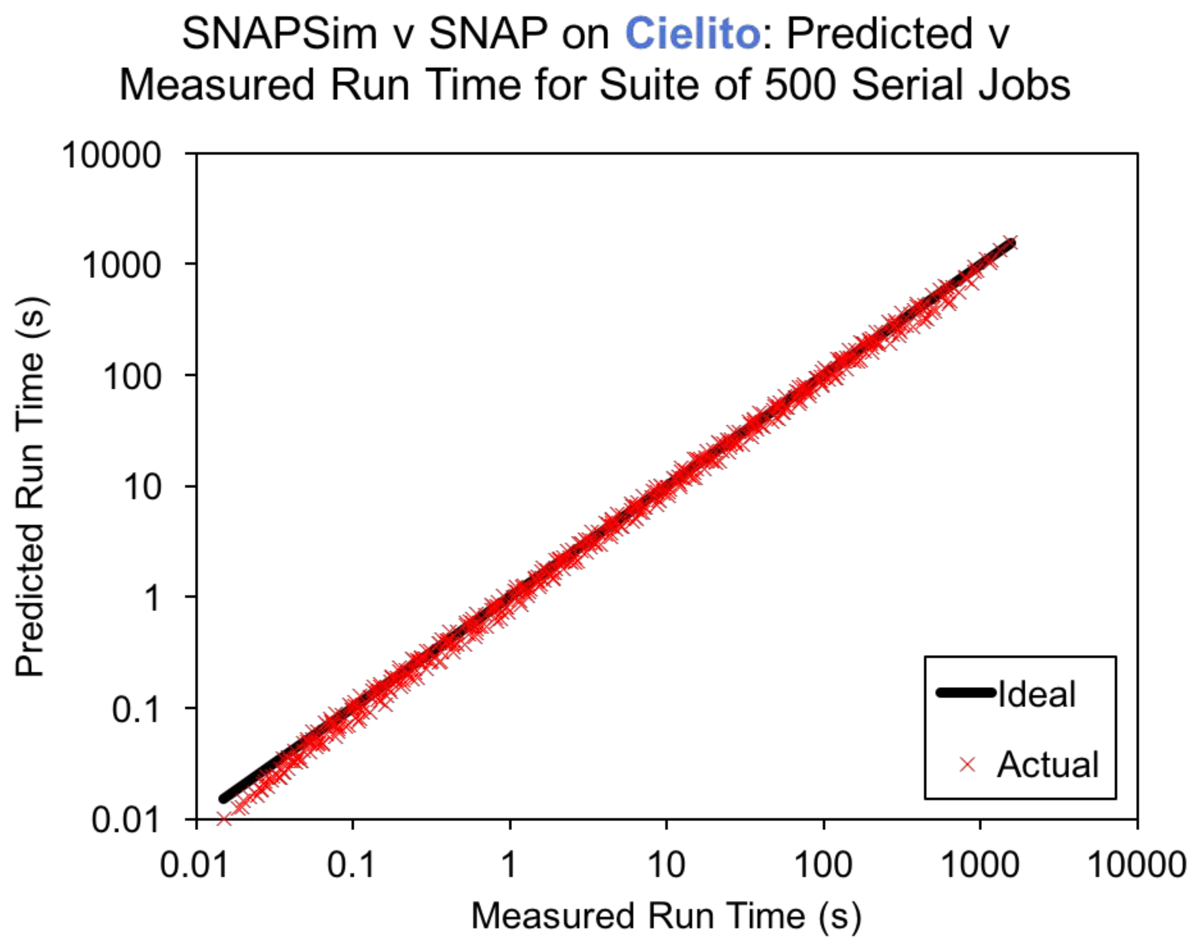
Application Models

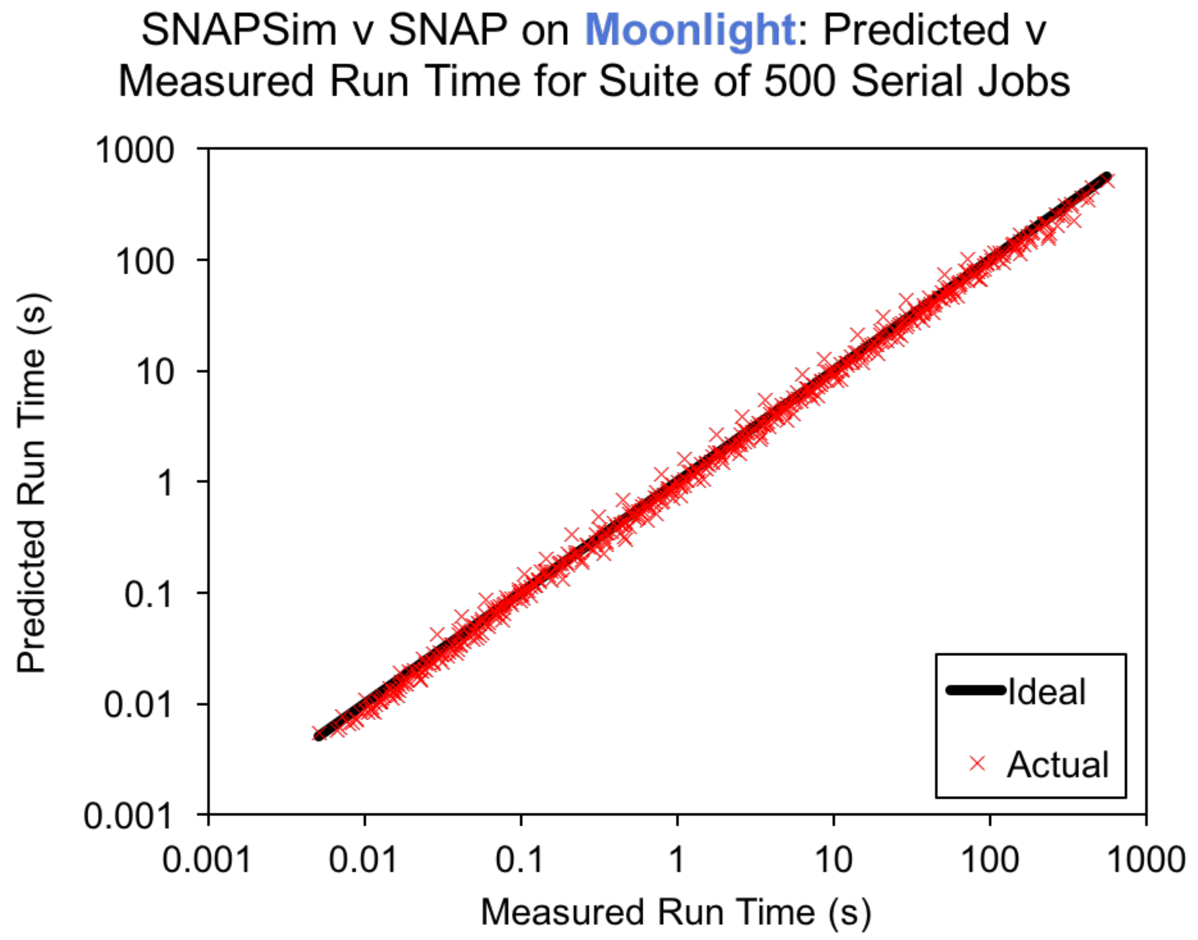
A 2-D illustration of the parallel wavefront solution technique for SNAP



- Stylized version of actual applications
  - Focus on loop structures, important numerical kernels
- Use MPI to facilitate communication
- Use node model to compute time:
  - Hardware configuration based on clock-speed, cache-level access times, memory bandwidth, etc.
  - Input is a task-list that consists of a set of commands to be executed by the hardware, including, for example, the number of integer operations, the number of floating-point operations, the number of memory accesses, etc.
  - Predict the execution time for retrieving data from memory, performing ALU operations, and storing results

SNAP: SN Application Proxy



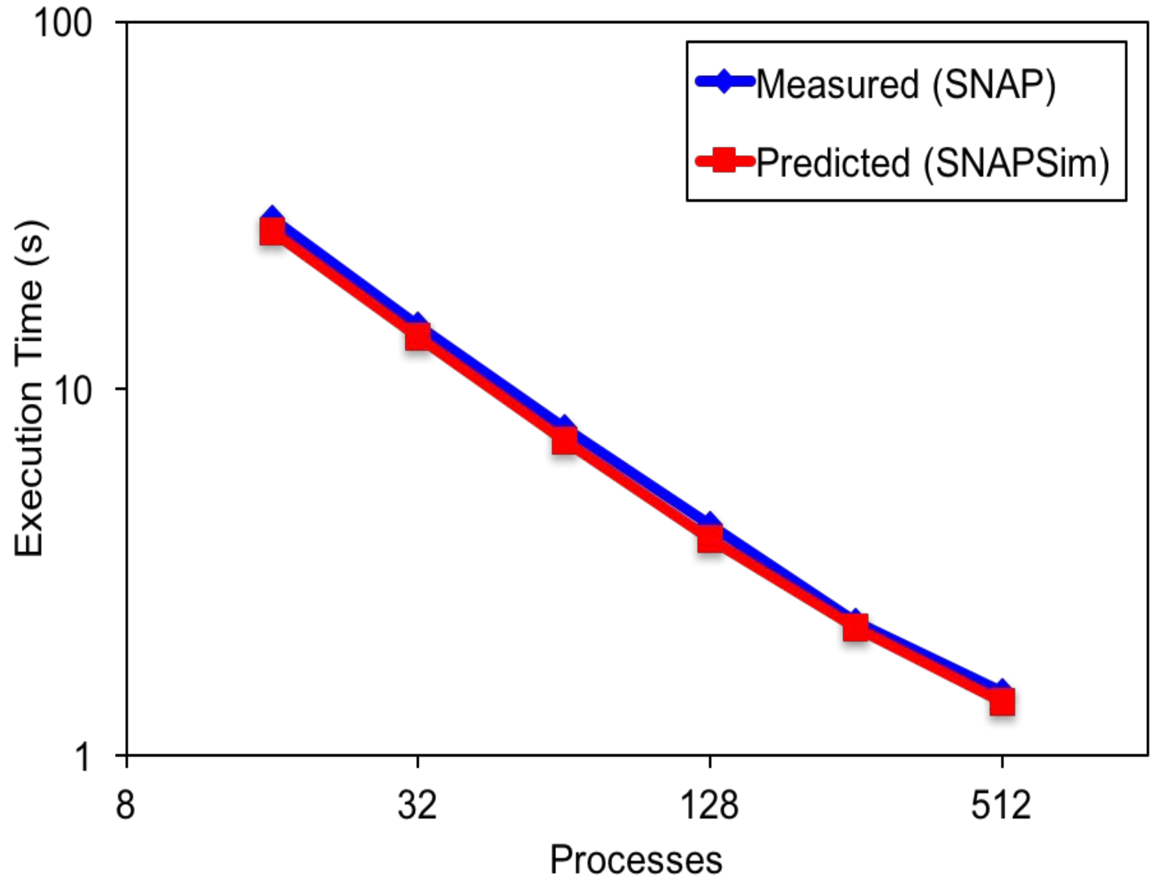


A suite of 500 SNAP and SNAPSim jobs, varying the number of spatial cells, the number of angular directions per octant, the number of energy groups, and the number of angular moments for particle scattering approximation. Changing them has effects on memory hierarchy and parallelism.

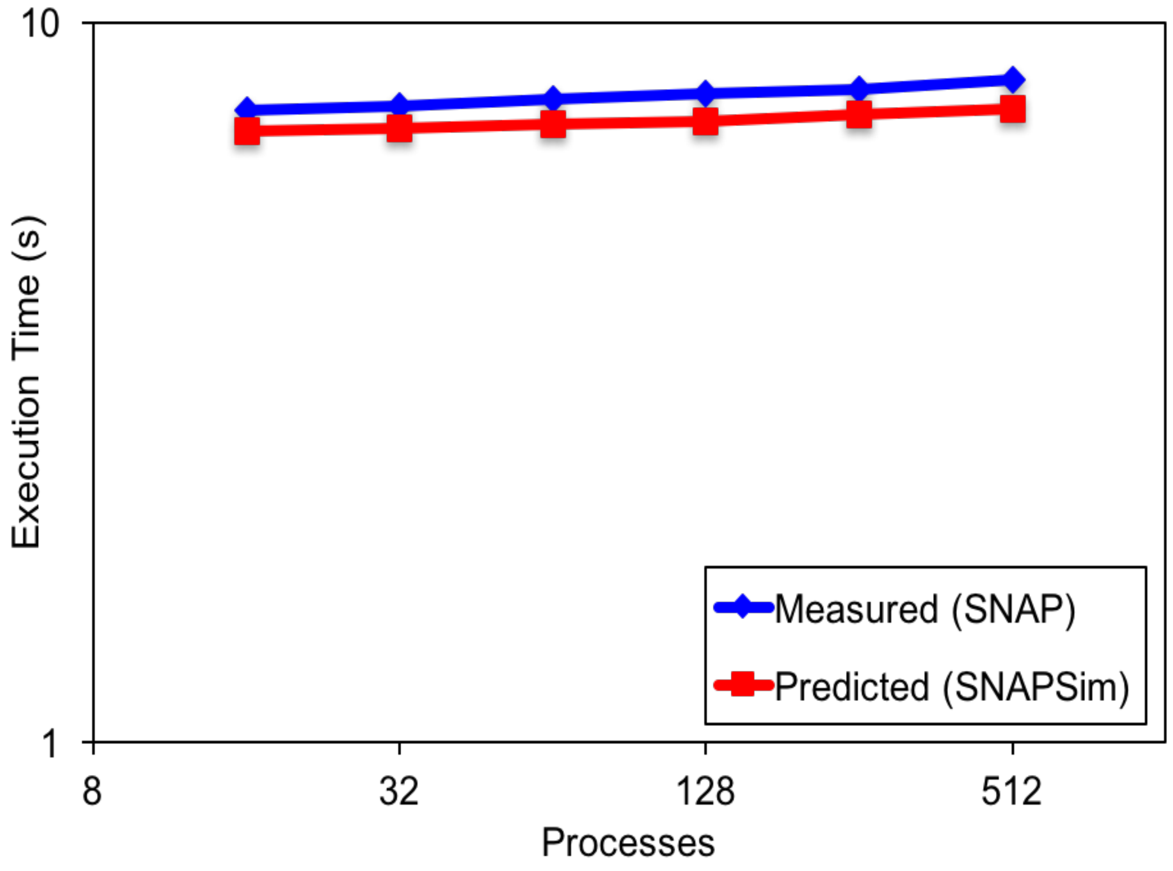
Experiments conducted by Joe Zerr at LANL.

Scaling Experiments on Cielito

Strong Scaling Results



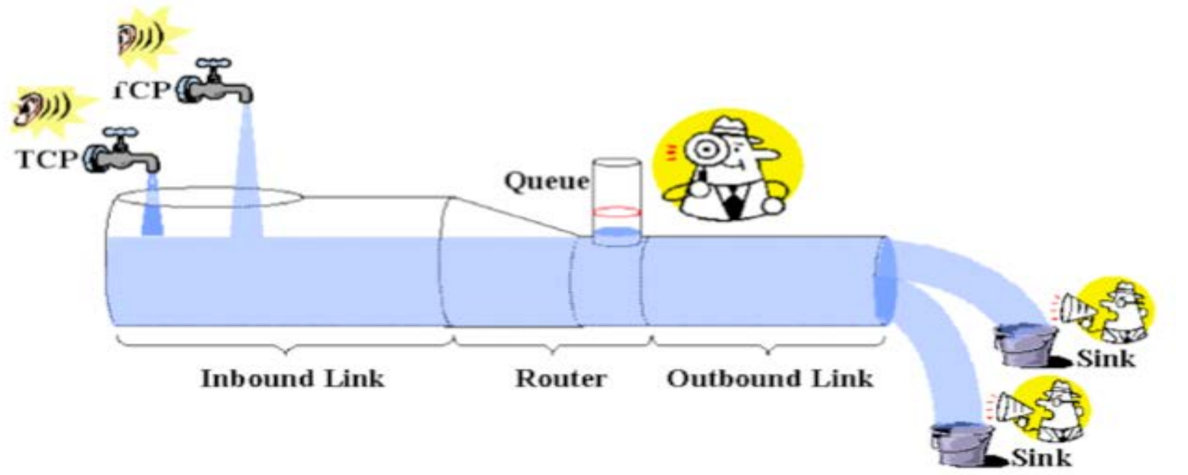
Weak Scaling Results



Experiments conducted by Joe Zerr at LANL.

Thoughts on Integrated Multi-Resolution HPC Modeling and Simulation

- Foreground vs background applications:
  - **Foreground applications** are the target of our study; they need to be modeled with high fidelity
  - **Background applications** cannot be modeled in great detail due to uncertainties and high computational complexity, but they may have significant influence on the overall performance
- Multi-resolution models:
  - Fluid models, hybrid models
  - Large-scale application behaviorals
  - Traffic matrices



Thanks