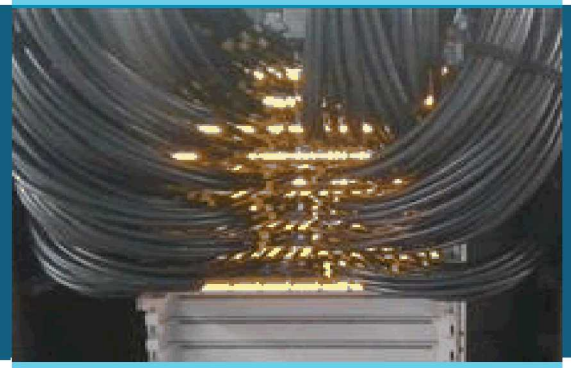
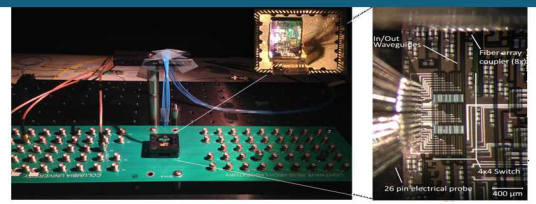
SAND2019-9854PE

Supercomputer in a workstation: simulation as a development platform for network architectures





PRESENTED BY

Jeremiah Wilke, Scalable Modeling and Analysis (8753)

Collaborators: Joseph Kenny, Cannada Lewis, Samuel Knight, Scott Hemmert

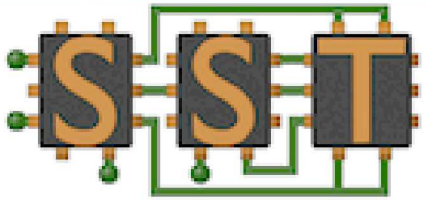
CIS External Review, August 26-29, 2019

SAND 2019XXX-XX

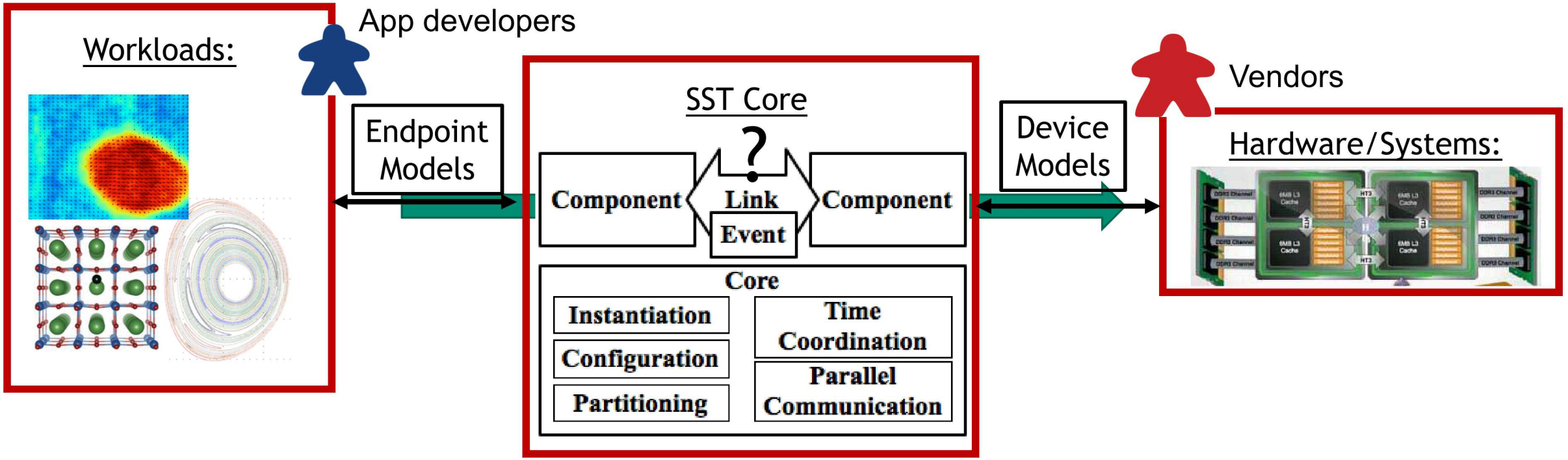
Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

The Structural Simulation Toolkit (SST) is analysis tool covering a broad range of architectural questions for the lab

PM: Rob Hoekstra PI: Si Hammond				
Interconnects	**Node (memory/compute)**	**New Architectures (Post-Moore's)**	**Compiler Tools**	**Simulation Core**
PI: Scott Hemmert Joseph Kenny Jeremiah Wilke Mike Levenhagen CA Budget: 600K	Gwendolyn Voskuilen Clayton Hughes Arun Rodrigues	Arun Rodrigues	Jeremiah Wilke Cannada Lewis CA Budget: 200K	Scott Hemmert Jeremiah Wilke Jon Wilson Jon VanDyke CA Budget: 150K
ASC/CSSE, ECP H&I	ASC/CSSE, ECP H&I	ASCR, DOD	ASC/CSSE	ASC/CSSE



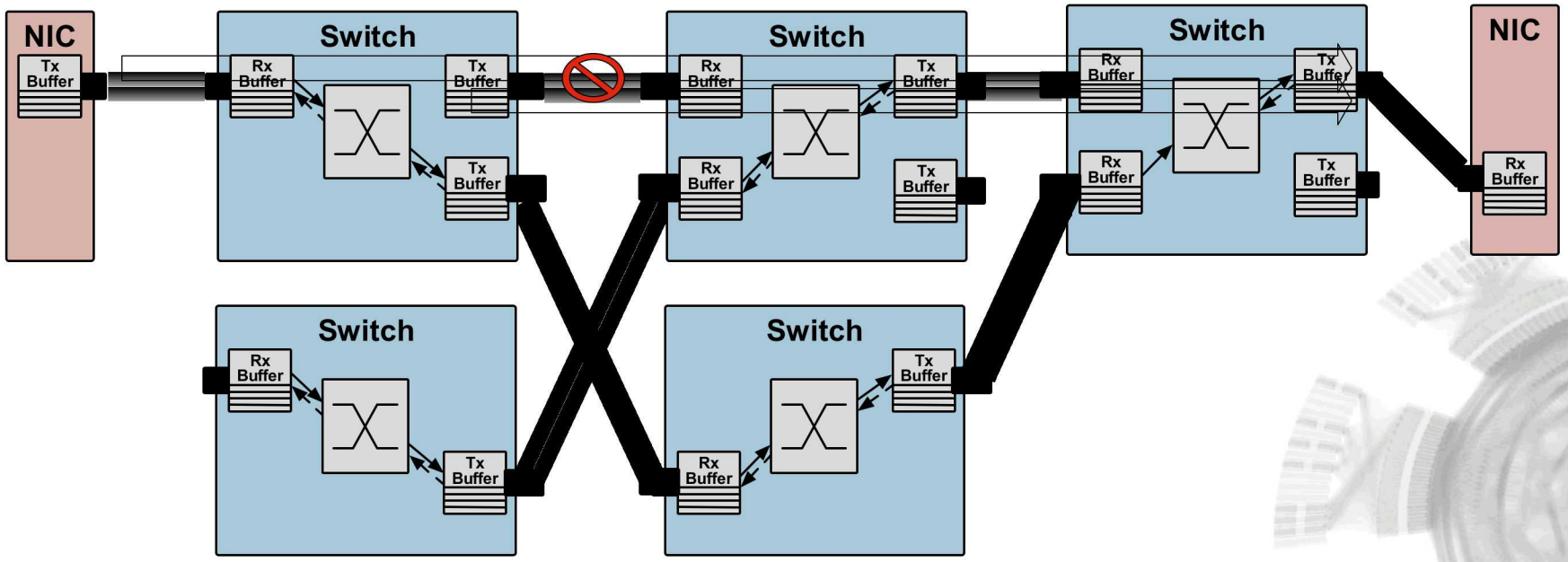
Application teams need mechanism to convey requirements between application teams and HPC system vendors



The Structural Simulation Toolkit provides analysis framework for answering these questions

For interconnects, lab must ensure systems have scalable networks covering topologies, routing, and the software stack

Figure: Two traffic flows contending for bandwidth across the network



Goal for interconnects is simple:
Get data onto the network as quickly as possible. Avoid contention on the network!

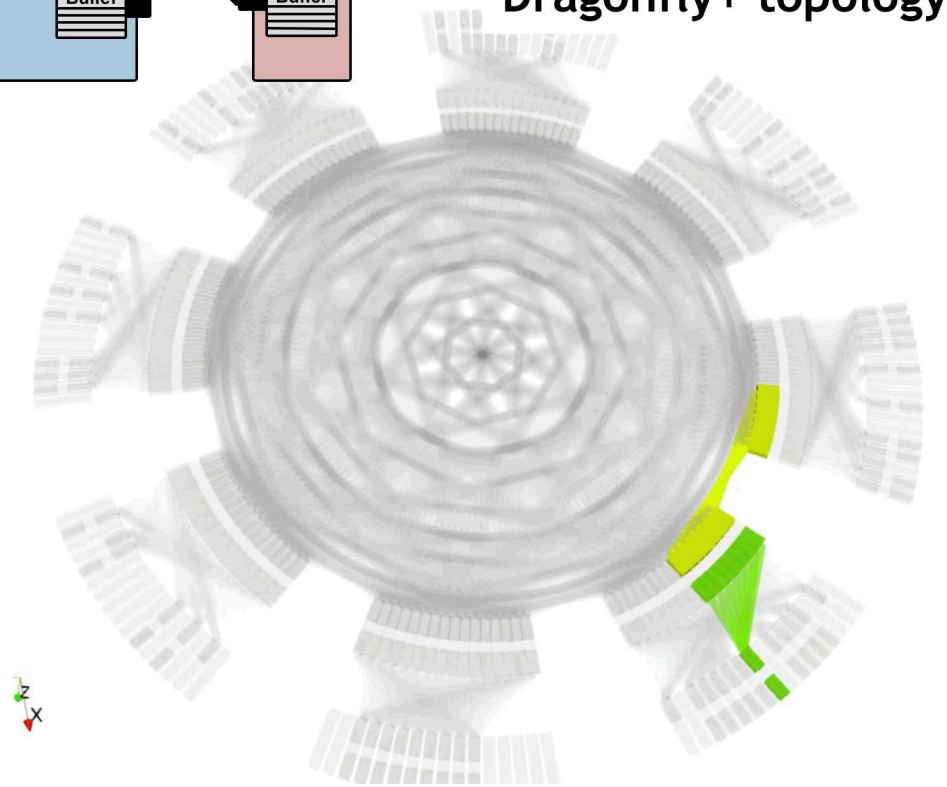


Figure: Traffic on a Dragonfly+ topology

The lab needs to work with vendors to advance new software and new hardware from idea to production

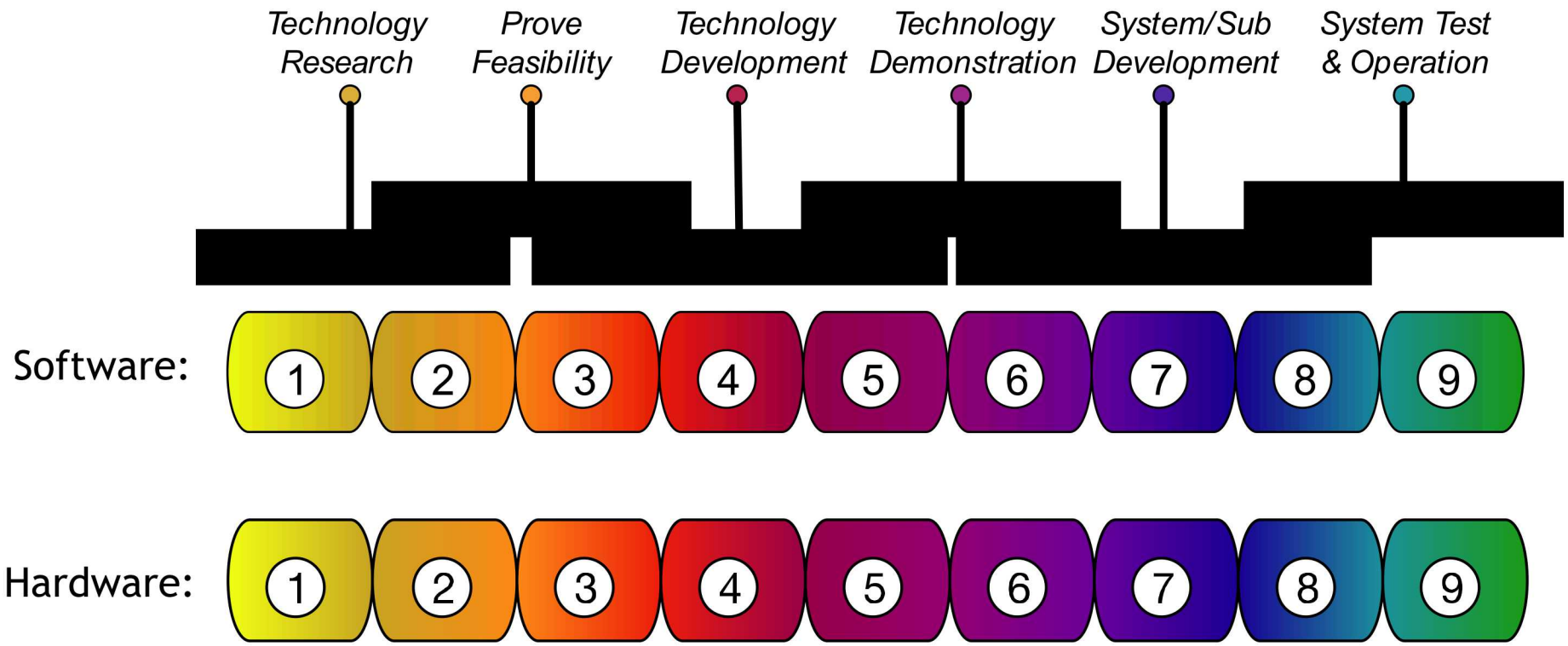


Figure: Technology readiness levels used by Sandia to categorize transition from idea to product

Main SST use case: simulation is an analysis tool for deciding on best ***mature*** interconnect designs for procurement

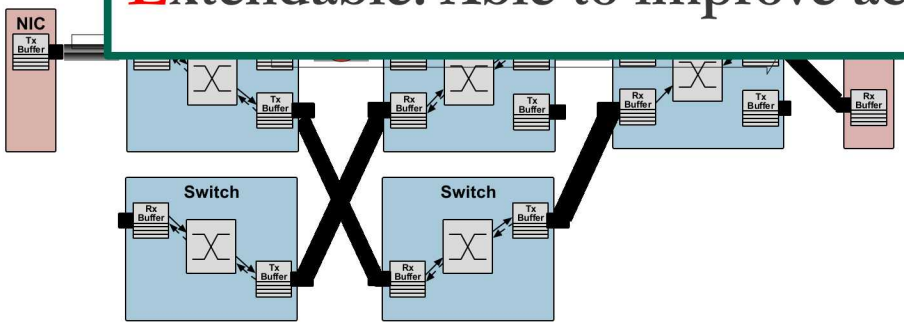
- Hardware and procurement teams choose what to buy and how to configure it
  - **Achievement:** Key component of Trilab L2 milestone exploring next-generation designs
  - **Achievement:** Key component of hardware evaluation within ECP H&I (Hardware and Integration)
- Vendors (particularly PathForward) prioritize development based on customer feedback
  - **Achievement:** Positive feedback from vendors using SST to prototype designs

Conveying application requirements through simulation requires "endpoint model" that generates realistic traffic

Challenge is scale: Can I simulate a supercomputer without an even bigger supercomputer?

- Opti
  - V
- Opti
  - E
- Opti
  - F
- Opti

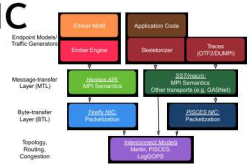
- **CO**llaboration: Engagement with both app developers and network vendors
- **V**alidation/Verification: Possible to demonstrate correctness on existing system
- **F**lexibility: Able to tune with different parameters
- **E**fficiency: Able to execute on limited compute resources
- **F**ruitful: Provides useful results, preferably more than one-off study
- **E**xtendable: Able to improve accuracy and detail if needed



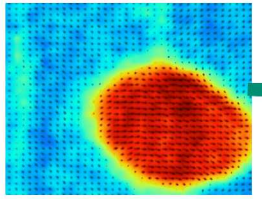
The "traffic pattern" on the network characterizes our unique requirements

Compiler tools can eliminate rate-limiting step in generating endpoint models for interconnect designs

Simulator-specific models

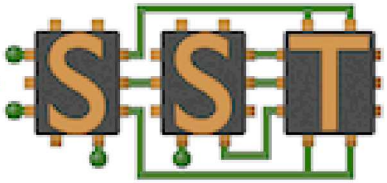


Auto-generate "skeleton" with compiler



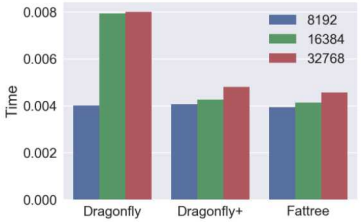
Run online model

Run skeleton app

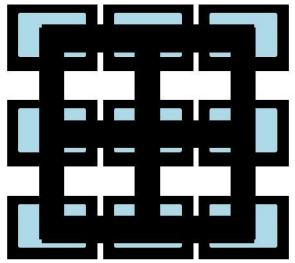


On-line models directly running applications and real network stacks are more accurate, more useful

Output results



Choose design




```
MPI_Send(…)
…
MPI_Allreduce()
```

Run app

Output trace

Trace replay

```
MPI_Send(…)
Compute(x ms)
MPI_Allreduce(...)
Compute(y ms)
...
```

Related Work: Simulators, Performance Analysis Tools, and Network Runtimes

Related Project	Description	Where	
Score-P + OTF2	Profiling and tracing tools	Jülich (with DOE funding)	https://www.vi-hps.org/projects/score-p/
Tracer/CODES	Interconnect simulator largely based on traces	Argonne and Lawrence Livermore	https://github.com/LLNL/TraceR/
OMNet++	Parallel simulation framework popular with internet networks	Academic Community	http://omnetpp.org
SMPI/SimGrid	Simulation framework for running MPI apps	INRIA	https://github.com/simgrid/simgrid

SST/macro is unique in its ability to leverage compiler support, mixed fidelity models, and HPC focus

Designing exascale interconnects is a challenge across the entire software stack with many lab projects involved

These design questions often involve either hardware or software that doesn't exist yet!

Applications
1) Choose scalable algorithm (weak,strong)
2) Express communication pattern to network stack using API

Network Software Stack
1) Collective algorithms
2) Choose and implement protocols
3) Choose service levels
4) Provide API for applications
5) Place jobs on nodes

Interconnect Hardware
1) Choose topology
2) Implement adaptive routing
3) Implement service levels and congestion control
5) Support software-defined networking (SDN)
6) High throughput for both large and small messages

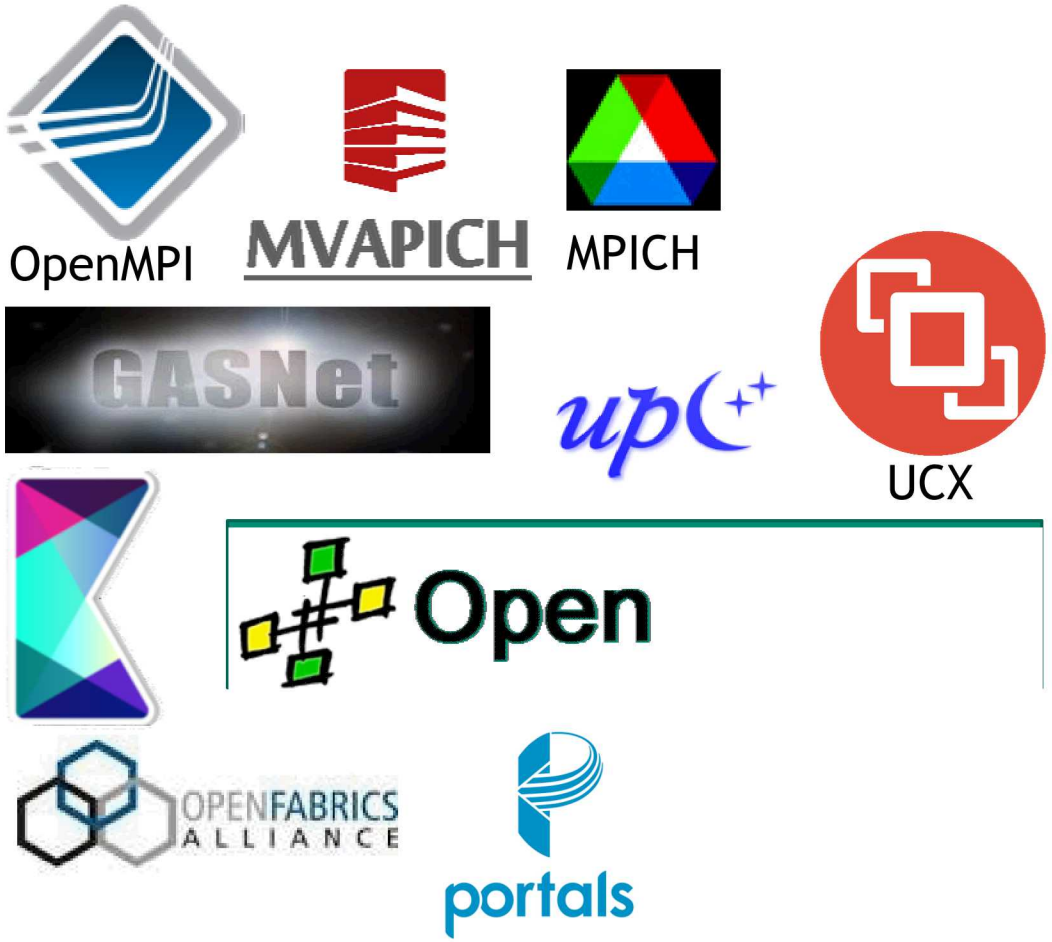


Figure: Some of the projects with DOE funding/collaborations affecting the network stack. Many others including Charm++, Legion, DARMA

The lab needs to work with vendors to advance new software and new hardware from idea to production

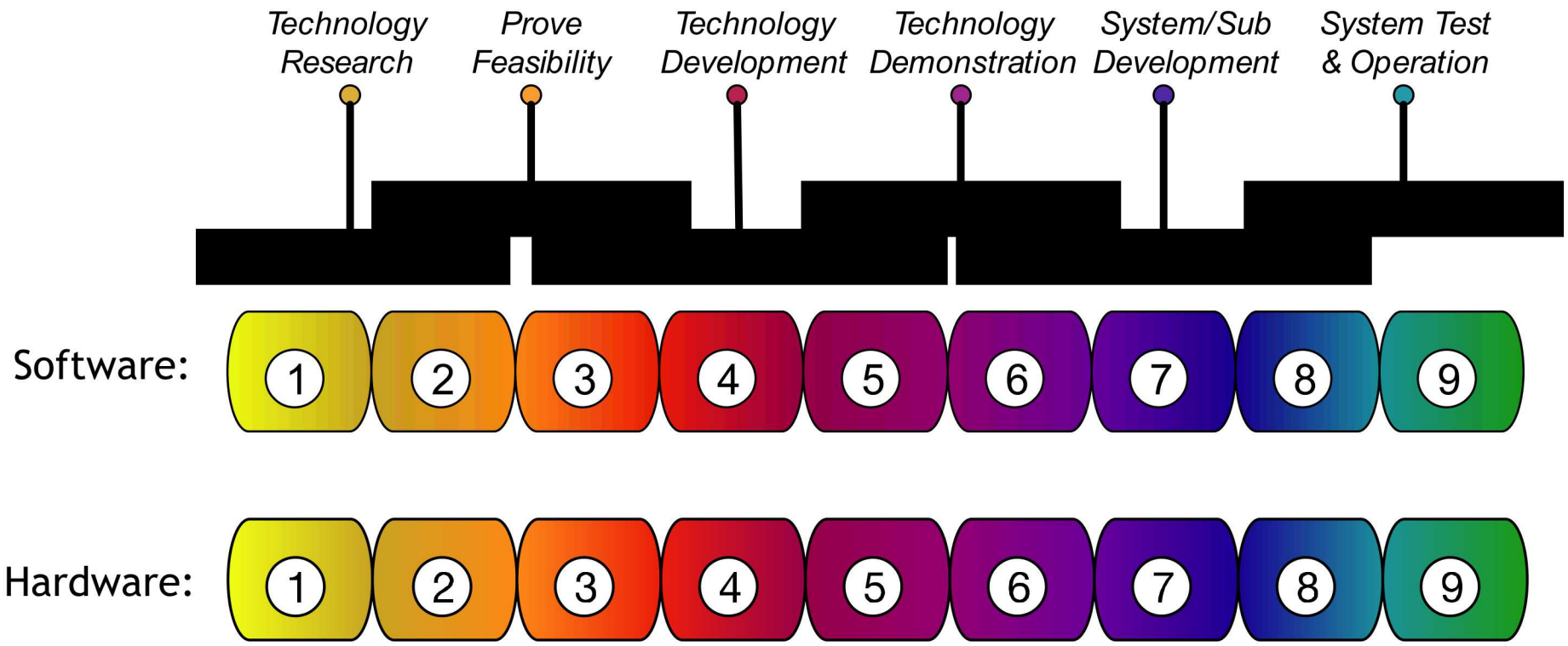
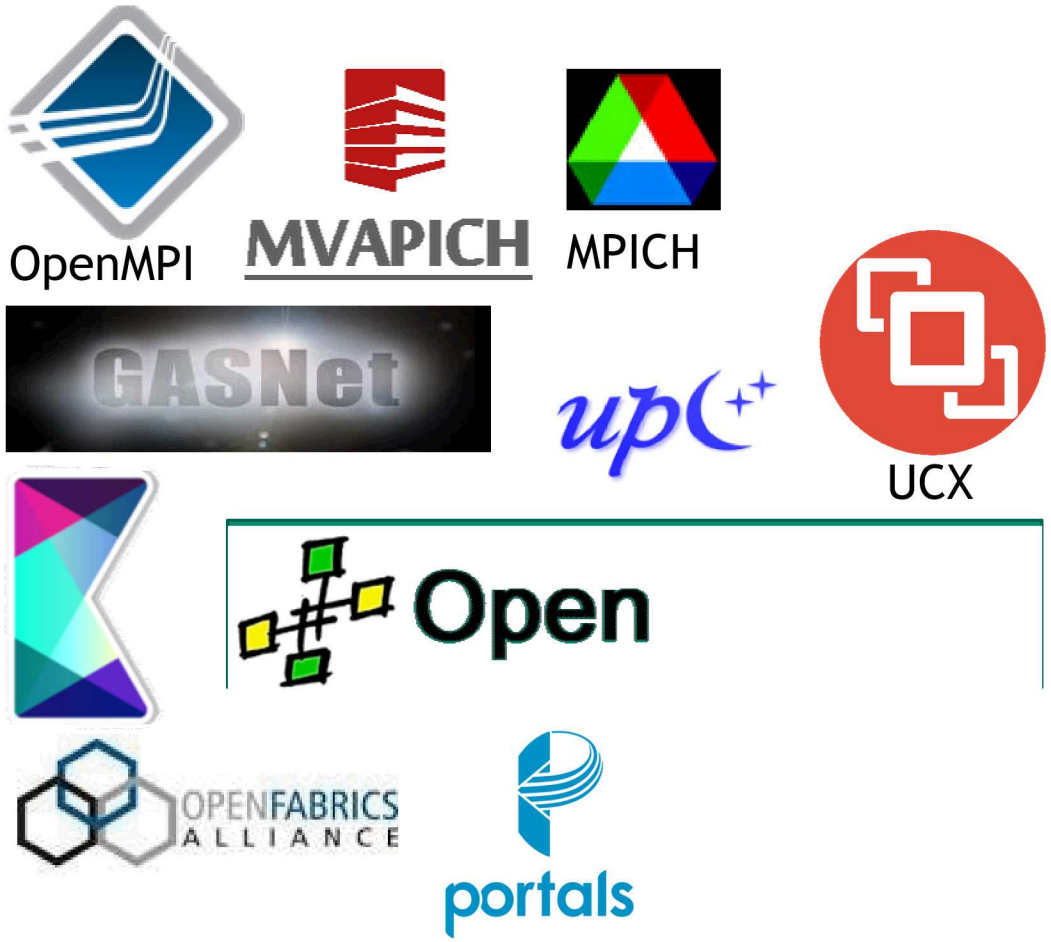
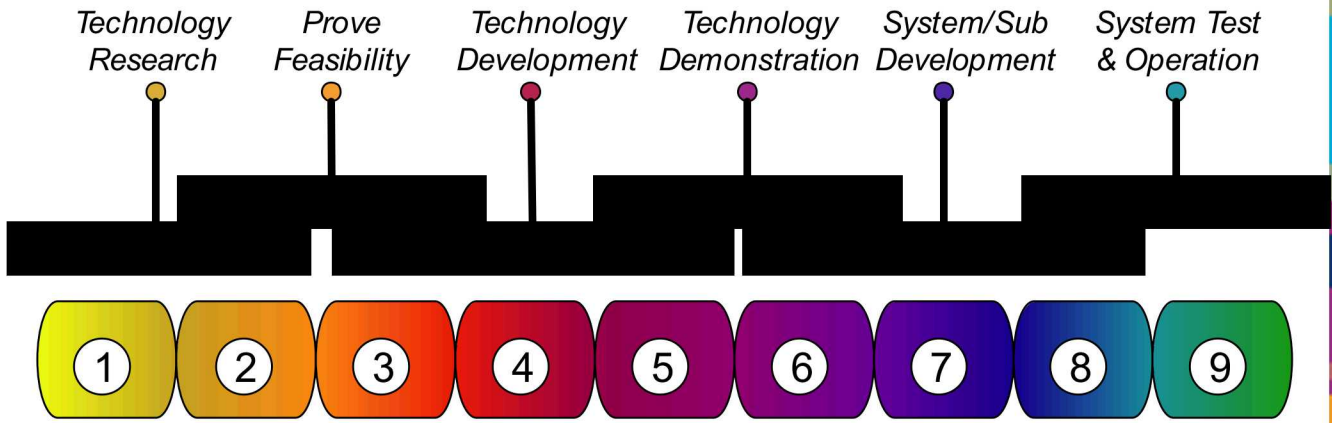


Figure: Technology readiness levels used by Sandia to categorize transition from idea to product

Numerous projects touching the network stack need the lab to develop or collaborate on development to advance TRL



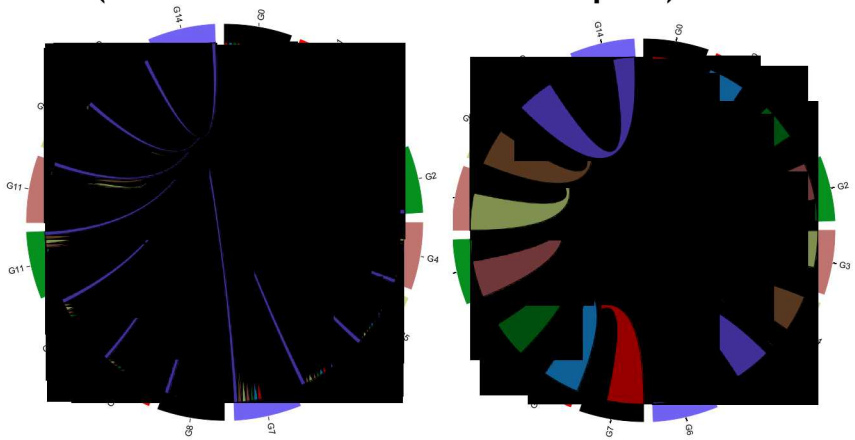


Initial Ideas → ? → Production Use

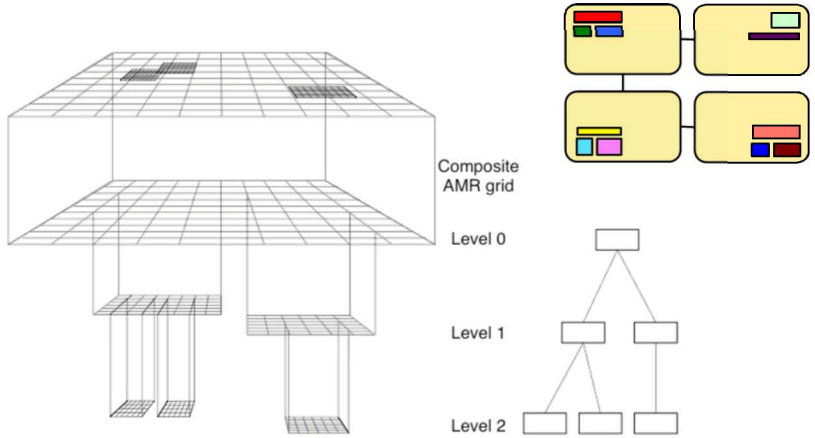
TRL can apply to both software and hardware!

Beyond procurement, simulator used to propose hardware and software solutions for future interconnect challenges

Flexfly bandwidth steering (SC' 16 Best Student Paper)

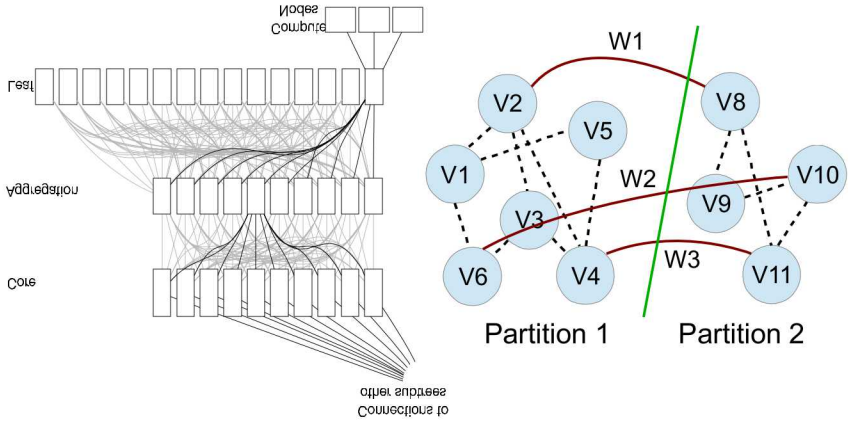


AMR topology-aware box placement (CommHPC '16 Best Paper)

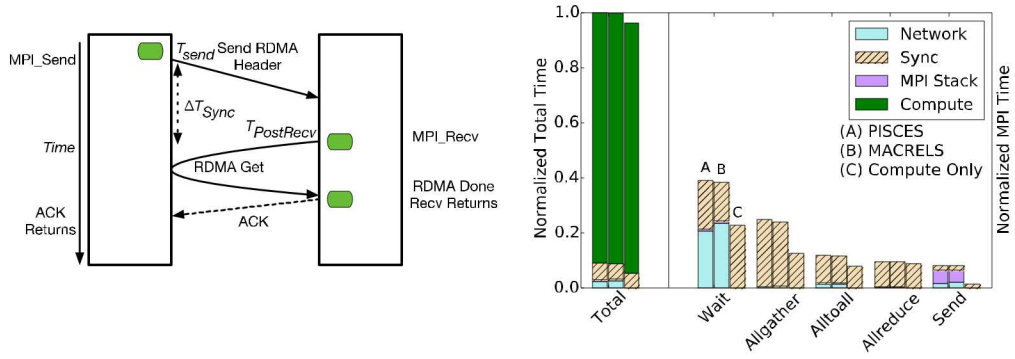


All of these studies required capturing application data, changing MPI runtimes, and simulating non-existent hardware!

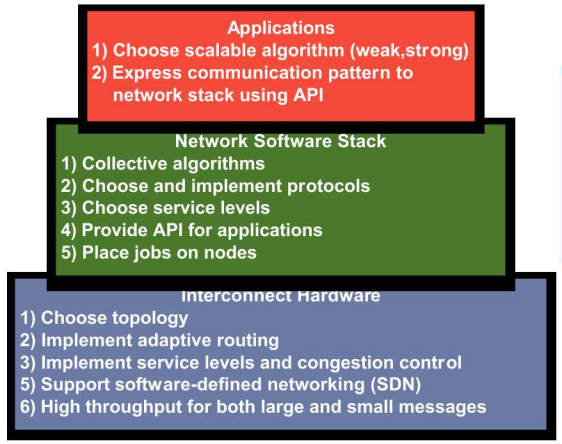
APHiD: Task placement to enable deeply tapered networks (CCGrid 17)



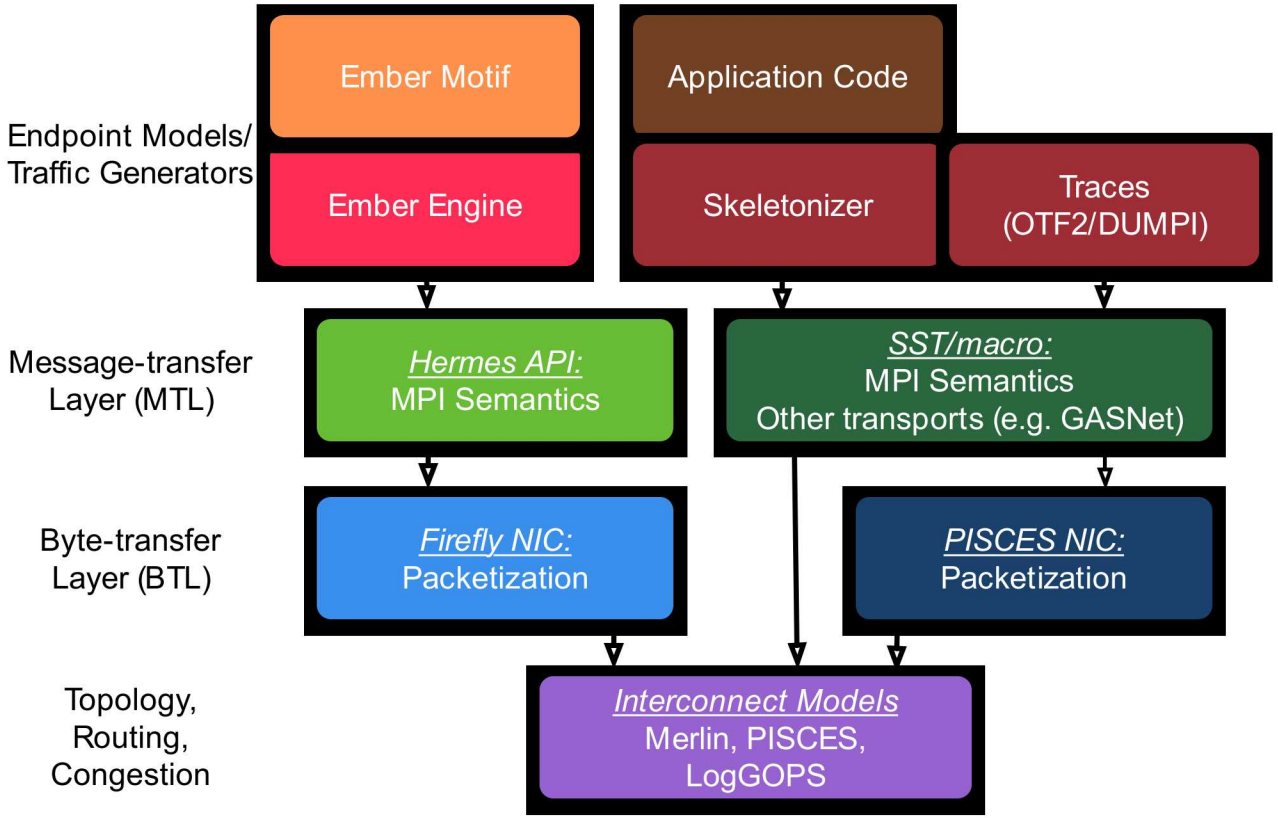
Network delay diagnostics with globally synchronous clock (ISC 18)



Theoretical studies difficult to extend into working products when only running *simulator-specific* communication libraries



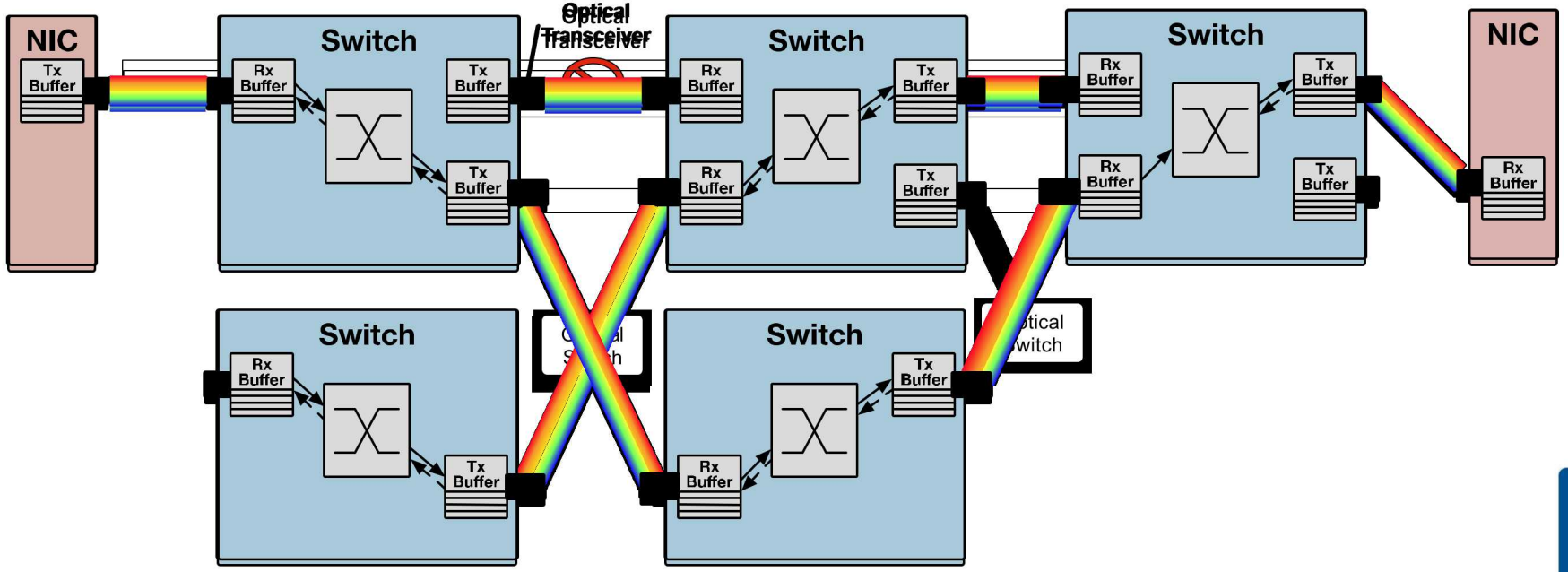
Each design issue requires an implementation in SST



OPENFABRICS ALLIANCE

MVAPICH

Illustrative example: Reconfigurable optical interconnects study shows how challenging technology transitions are



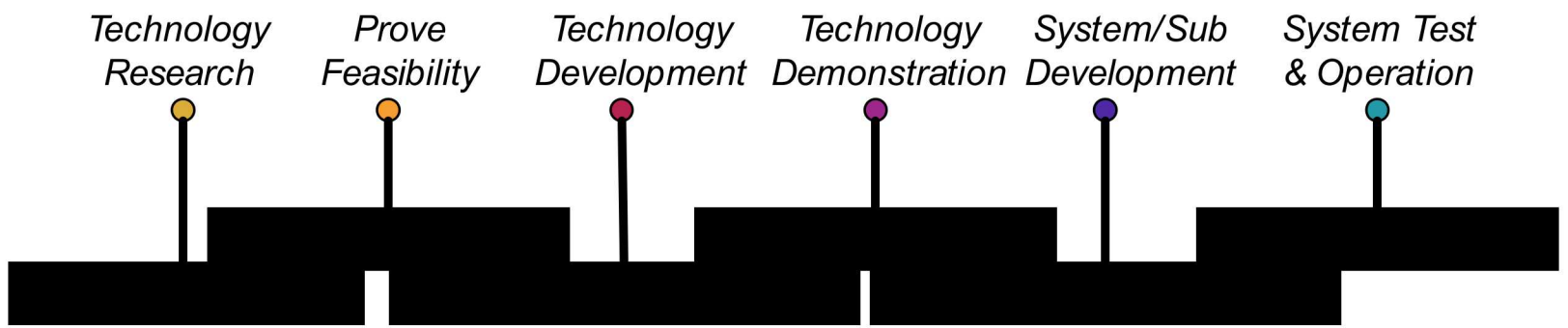
Results showed 2X speedup with *reduced* energy

Columbia University

Collaboration with Keren Bergman

Figure: Two traffic flows contend for bandwidth across electrical network
Figure: Electrical links replaced with optical links for higher bandwidth density
Figure: Reconfigurable switches *move* bandwidth to alleviate hotspots
Figure: Two traffic flows no longer contend for the same network path

Transitioning from an interesting idea in a simulator-specific model to a ready product is challenging



No hardware exists to advance TRL of software stack!

Initial idea -> Demo with OpenFlow -> Add to MPI stack -> Deploy on system

Simulator produces ideas at TRL 1-3

Theoretical studies difficult to extend into working products when only running *simulator-specific* communication libraries

Applications
1) Choose scalable algorithm (weak,strong)
2) Express communication pattern to network stack using API

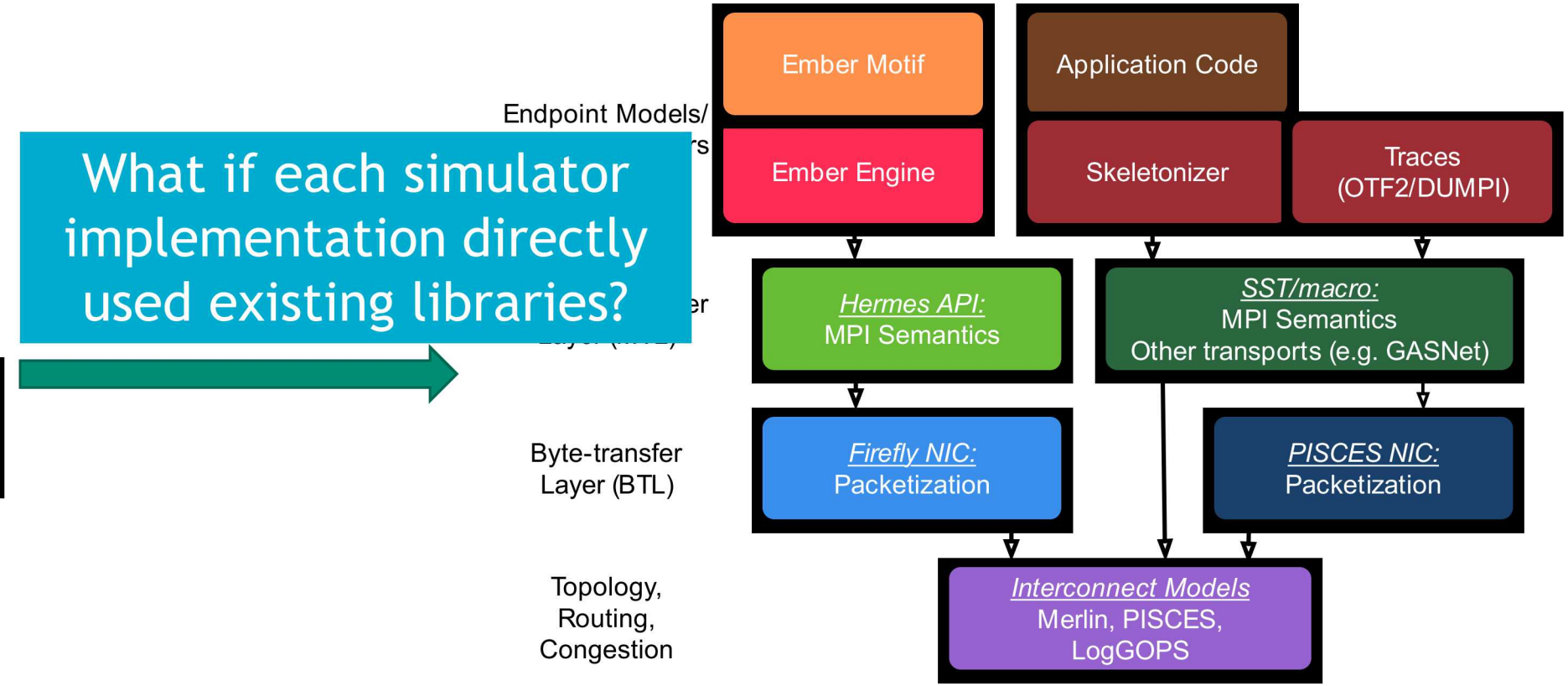
Network Software Stack
1) Collective algorithms
2) Choose and implement protocols
3) Choose service levels
4) Provide API for applications
5) Place jobs on nodes

Interconnect Hardware
1) Choose topology
2) Implement adaptive routing
3) Implement service levels and congestion control
5) Support software-defined networking (SDN)
6) High throughput for both large and small messages

OPENFABRICS ALLIANCE

MVAPICH

What if each simulator implementation directly used existing libraries?



To solve problem by directly simulating real application code requires overcoming the challenge of scale

Solution: Compiler support to automatically generate endpoint models by eliminating expensive memory/compute

Original Source Code:

```
double* big = new double[N];
MPI_Sendrecv(big,…);
for (i=0; i < N; ++i){
  expensive_compute();
}
MPI_Allreduce(…);
```

1) Developer adds pragmas

Modified Source Code:

```
#pragma sim null_variable
double* big = new double[N];
MPI_Sendrecv(big,…);
#pragma sim compute
for (i=0; i < N; ++i){
  expensive_compute();
}
MPI_Allreduce(…);
```

2) Clang source-to-source

Auto-skeletonized Source Code:

```
double* big = nullptr;
MPI_Sendrecv(big,…); //modeled
modelCompute(N,…);
MPI_Allreduce(…); //modeled
```

3) sim++ src.cpp
Redirect MPI calls

Auto-skeletonized Object Code:

```
call SIM_MPI_Sendrecv(….);
call modelCompute(N);
call SIM_MPI_Allreduce(…);
```

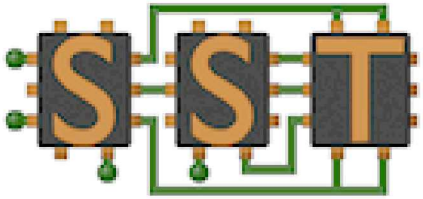
4) Link to simulator with SIM_MPI_X symbols
sim++ -o sim.x -lsim

Simulation Endpoint Model

64 GB memory
100 GF compute

High-fidelity simulation is possible for exascale network, but not for the entire exascale system

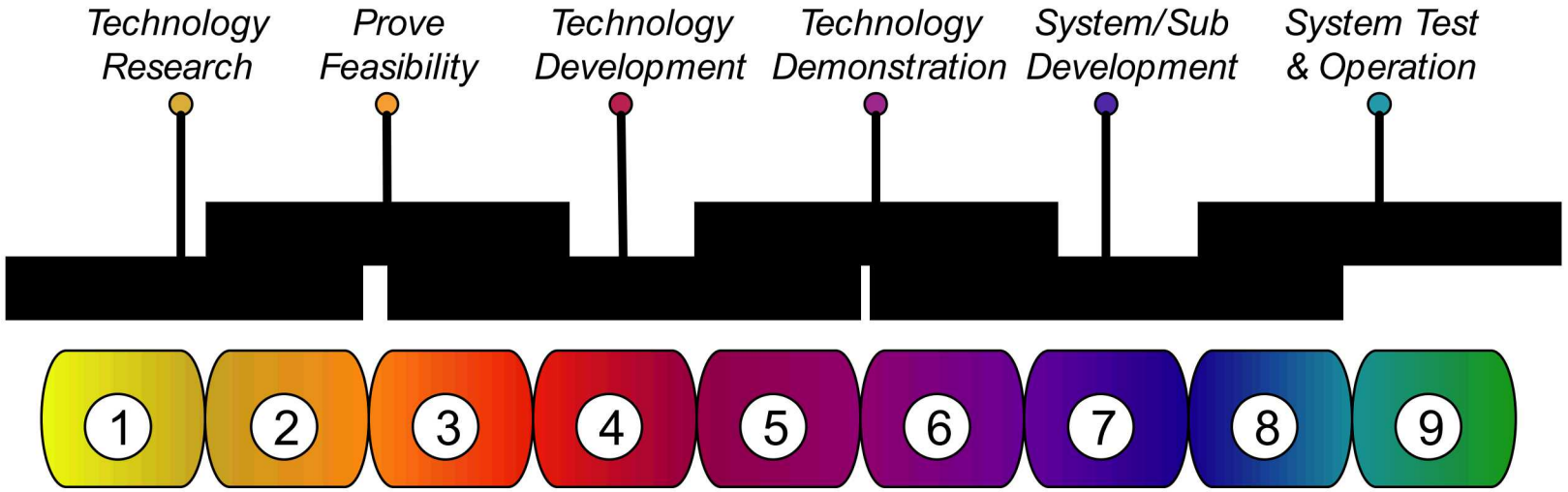
	High-Fidelity Sim of 1s (100x Overhead)		Exascale System		Coarse-Grained Sim of 1s (100x Cost Reduction)	
	Compute	Memory	Compute	Memory	Compute	Memory
Nodes	100 ExaOPs	25 PB	1 ExaOP/s	5 PB	5 TeraOPs	40 GB
Network Interface	1 PetaOPs	5 TB	400 GigaOP/s	500 GB	1 TeraOPs	5 GB
Switches	5 PetaOPs	100 GB	50 TeraOP/s	25 GB	5 TeraOPs	20 GB



Using the supercomputers of today to design the supercomputers of tomorrow

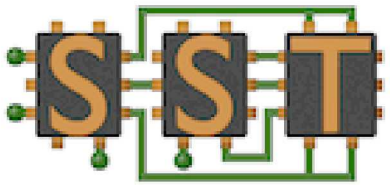
A coarse-grained simulation is feasible on a powerful workstation.
A mixed-fidelity (detailed network, coarse-grained nodes) is feasible with an existing supercomputer!

Shorten time to production-ready by eliminating rate-limiting step: don't need access to non-existent supercomputer



Compiler support allows simulator to advance TRL far beyond just interesting ideas

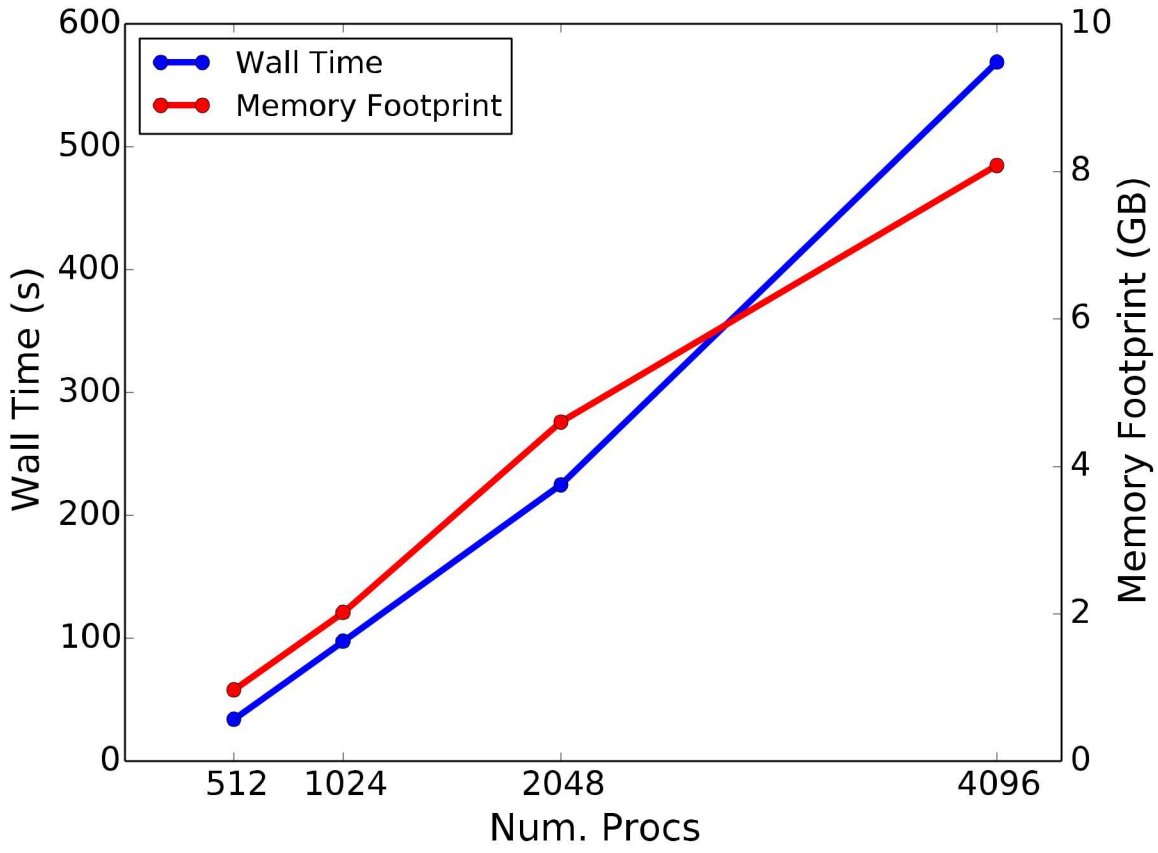
$$$$$
Every day hardware sits underutilized because software stack isn't ready costs money



Every day hardware or software development is delayed sacrifices lab computing leadership

Auto-skeletonization via compiler overcomes scaling challenges by reproducing behavior without expensive compute

Figure: Memory and compute of GASNet library in simulator

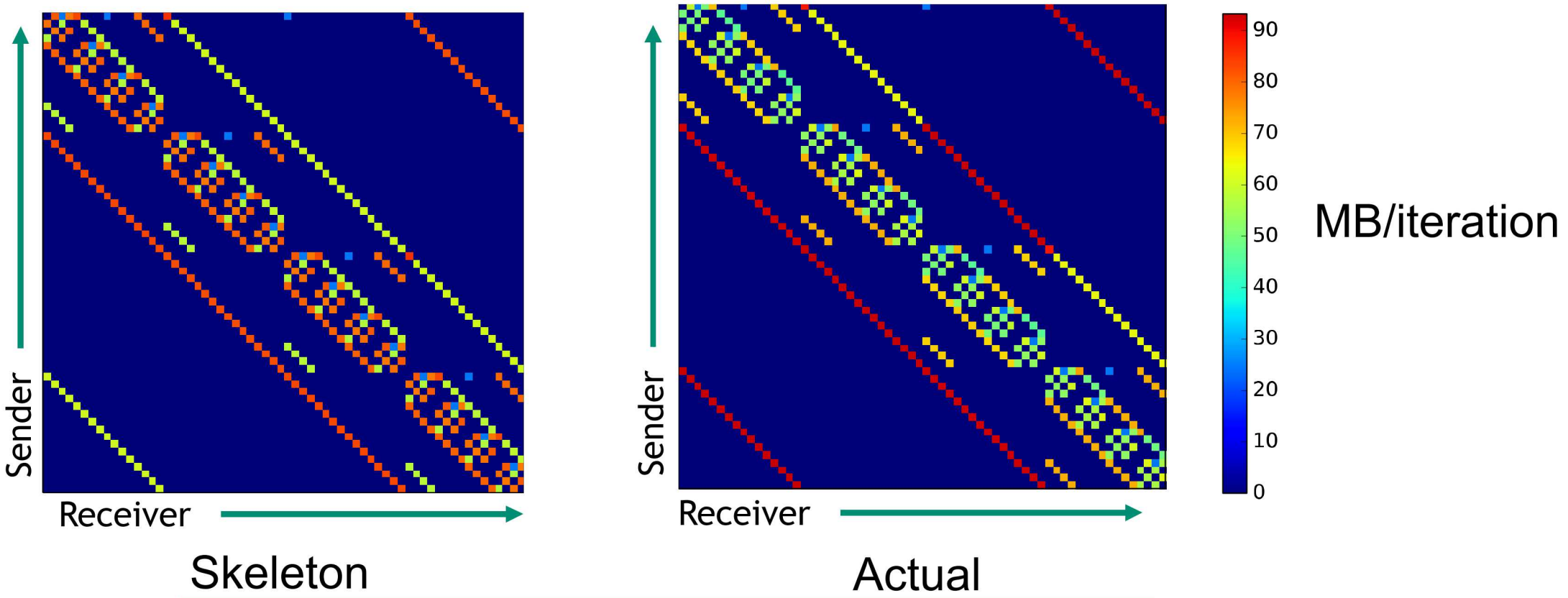


Application with GASNet runtime running directly in simulator, but injects traffic into *simulated network*

Running non-skeletonized version would be TBs memory!

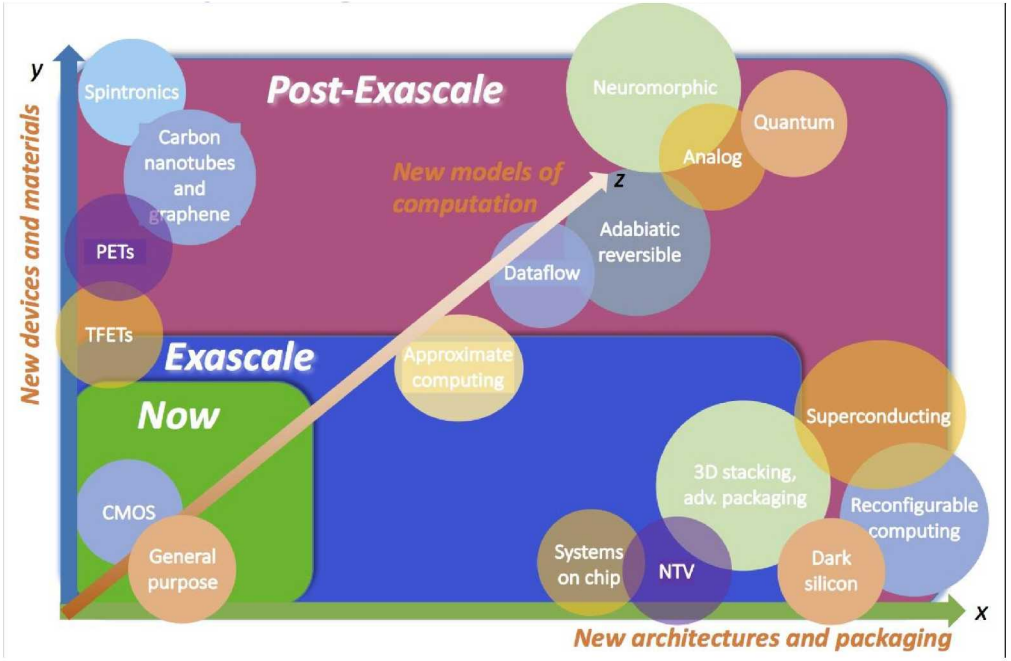
Auto-skeletonization via compiler overcomes scaling challenges by reproducing behavior without expensive compute

CoMD traffic pattern matrix plot
M(I,J) = traffic sent from Rank I to Rank J

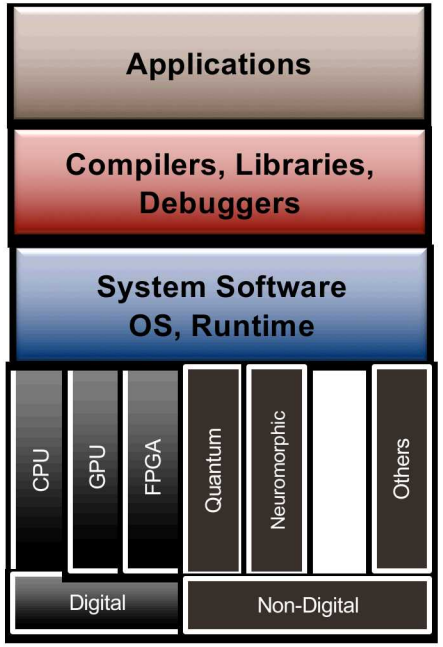


Despite compiler-introduced approximations, traffic pattern and compute times are reasonably reproduced

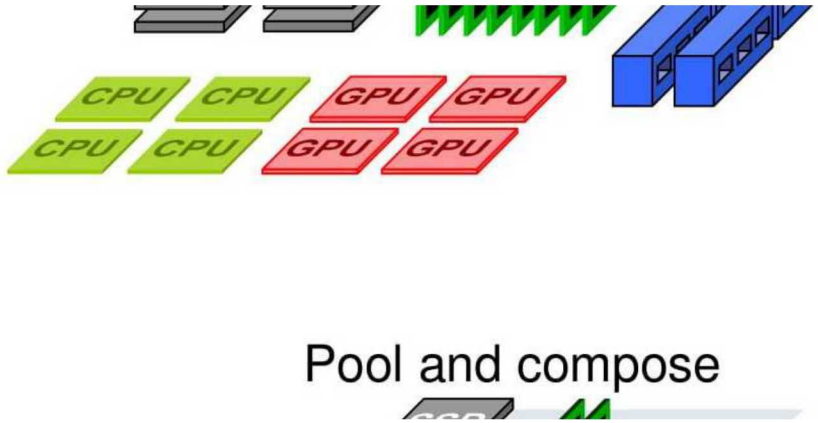
Future Work: Extreme heterogeneity, Post-Moore's, and disaggregated systems all require developing new network stacks



New devices and possibly new architectures (Brightwell, PMES 2018)



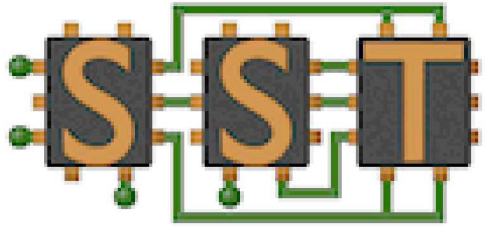
Develop software stacks for handling diverse set of accelerators (Brightwell, PMES 2018)



Disaggregating nodes creates new interconnect challenges (Bergman, 2018)

Summary

- Simulation provides the evidence for informed design decisions of both hardware and software
- Challenging problem to understand hardware or software that *doesn't exit yet* without expensive 1) code refactoring or 2) hardware test beds
- Endpoint models capture unique requirements of lab workloads
- Huge benefit in generating endpoint models from actual code via compiler tools over architecting simulator-specific solutions
- Value to Sandia:
  - Short-term: Institutional knowledge, aiding procurements and use of current systems
  - Medium-term: Predicting future problems and proposing solutions
  - Long-term: Explore experimental architectures

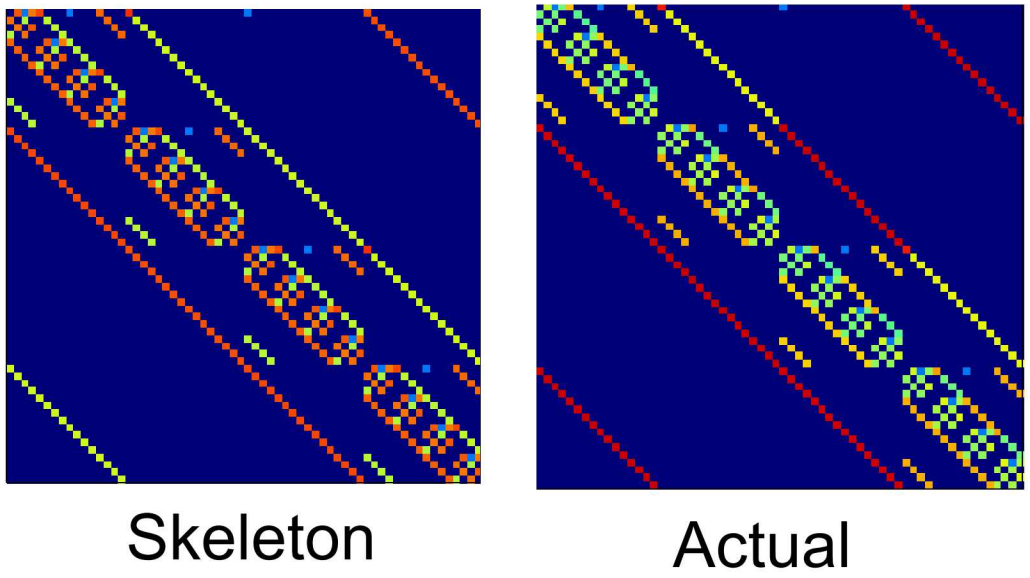


More Information

- Jeremiah Wilke (jjwilke@sandia.gov)
- SST: https://github.com/sstsimulator
- Representative Publications:
  - Wen et al, Flexfly: enabling a reconfigurable dragonfly through silicon photonics, SC 2016
  - Wilke et al, APHiD: Hierarchical Task Placement to Enable a Tapered Fat Tree Topology for Lower Power and Cost in HPC Networks CCGrid 2017
  - Teh et al, Design Space Exploration of the Dragonfly Topology, ExaComm 2017
  - Chan et al, Topology-aware performance optimization and modeling of adaptive mesh refinement codes for exascale, CommHPC 2016
  - Wilke et al, Compiler-assisted source-to-source skeletonization of application models for system simulation, ISC 2018
  - Kenny et al, The Pitfalls of Provisioning Exascale Networks: A Trace Replay Analysis for Understanding Communication Performance, ISC 2018

Auto-skeletonization via compiler overcomes scaling challenges by reproducing behavior without expensive compute

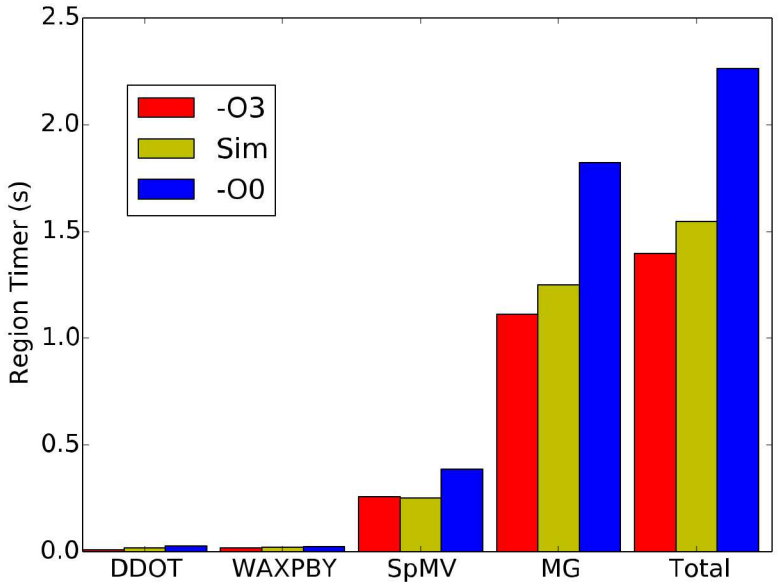
CoMD traffic patterns



Skeleton

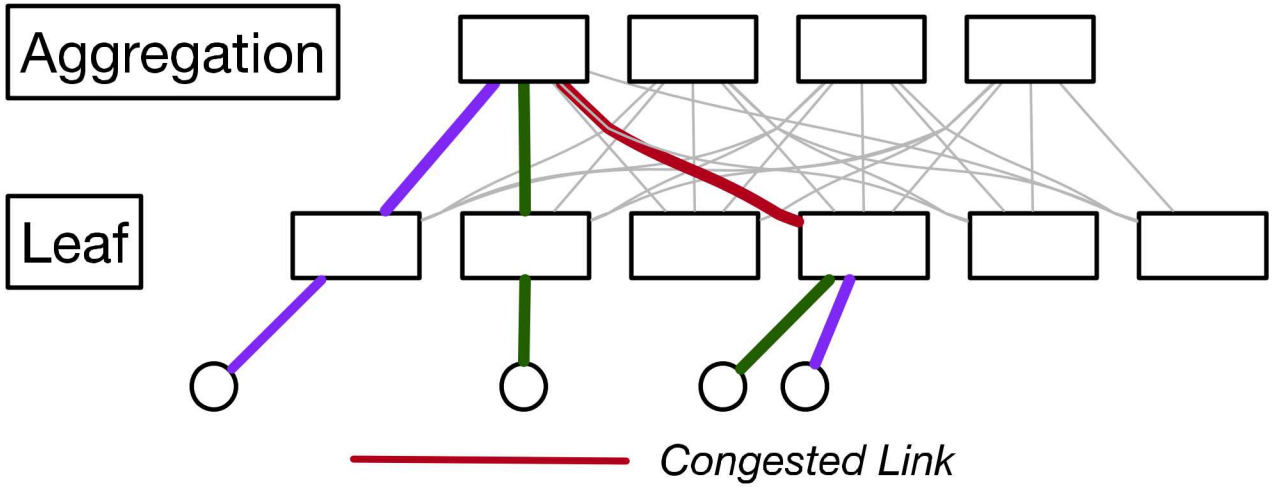
Actual

HPCG Compute Times



Despite approximations, traffic pattern and compute times are reasonably reproduced

The simulator is an analysis tool that can diagnose causes and test cures across the entire network stack



Goal for interconnects is simple:
Get data onto the network as quickly as possible. Avoid contention on the network!

Figure: Isolated network hotspot affects performance

Symptom	Cause	Cure	Where Fixed
Large MPI Time	Mismatched send/recv	Load balance, non-blocking calls	App
Large MPI Time	MPI Memory copies	Change communication protocol	Runtime
Large MPI Time	A few “hotspot” links	Adaptive routing	Hardware
Large MPI Time	Latency-sensitive messages	Quality-of-service	Runtime + Hardware
Large MPI Time	Heavy traffic, many “hotspots”	Bigger network, bandwidth steering	Runtime + Hardware