MVAPICH: How a Bunch of Buckeyes Crack Tough Nuts

5th Annual MVAPICH User Group Meeting

Adam Moody
Livermore Computing

August 15, 2017

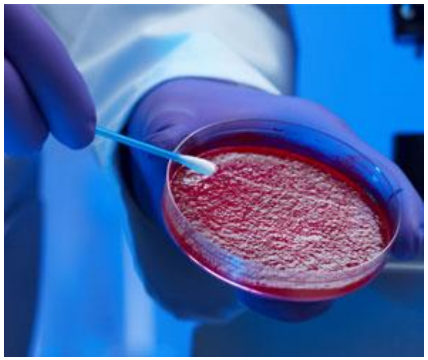
LLNL-PRES-736917
This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

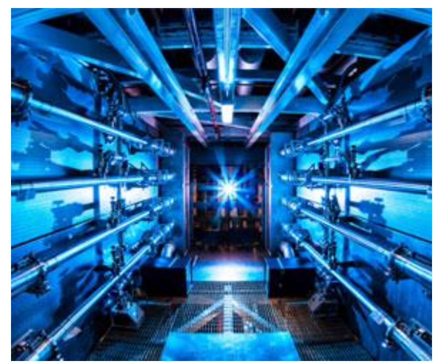
Lawrence Livermore National Laboratory

Who is LLNL?

LLNL's mission is applying world-class science, technology, and engineering to national & global problems

Bio-Security	Counterterrorism	Defense	Energy





Intelligence	Nonproliferation	Science	Weapons

https://missions.llnl.gov

LLNL systems by purpose

Capability
Capacity
Visualization
Serial

System	Top500 Rank	Program	Manufacture / Model	OS	Inter-connect	Nodes	Cores	Memory (GB)	Peak TFLOP/s	TFLOP/s (GPUs)
Unclassified Network (OCF)										
Vulcan	23	ASC+M&IC+HPCIC	IBM BGQ	RHEL/CNK	5D Torus	24,576	393,216	393,216	5,033.2	
Cab (TLCC2)		ASC+M&IC+HPCIC	Appro	TOSS	IB QDR	1,296	20,736	41,472	426.0	
Quartz	46	ASC+M&IC	Penguin	TOSS	Omni-Path	2,688	96,768	344,064	3251.4	
RZTopaz		ASC	Penguin	TOSS	Omni-Path	768	27,648	98,304	929.0	
RZManta		ASC	IBM	RHEL	IB EDR	36	720	11,520	597.6	676.8
Ray		ASC+M&IC	IBM	RHEL	IB EDR	54	1,080	17,280	896.4	1015.2
Catalyst		ASC+M&IC	Cray	TOSS	IB QDR	324	7,776	41,472	149.3	
Syrah		ASC+M&IC	Cray	TOSS	IB QDR	324	5,184	20,736	107.8	
Surface		ASC+M&IC	Cray	TOSS	IB FDR	162	2,592	41,500	451.9	451.9
Borax		ASC+M&IC	Penguin	TOSS	N/A	48	1,728	6,144	58.1	
RZTrona		ASC	Penguin	TOSS	N/A	48	1,728	6,144	58.1	
Herd		M&IC	Appro	TOSS	IB DDR	9	256	1,088	1.6	
OCF Totals	Systems	12							11,960.4	2,143.9
Classified Network (SCF)										
Pinot(TLCC2, SNSI)		M&IC	Appro	TOSS	IB QDR	162	2,592	10,368	53.9	
Sequoia	5	ASC	IBM BGQ	RHEL/CNK	5D Torus	98,304	1,572,864	1,572,864	20132.7	
Zin (TLCC2)	217	ASC	Appro	TOSS	IB QDR	2,916	46,656	93,312	961.1	
Jade+Jadeita	45	ASC	Penguin	TOSS	Omni-Path	2,688	96,768	344,064	3251.4	
Mica		ASC	Penguin	TOSS	Omni-Path	384	13,824	49,152	464.5	
Shark		ASC	IBM	RHEL	IB EDR	36	720	11,520	597.6	676.9
Max		ASC	Appro	TOSS	IB FDR	324	5,184	82,944	107.8	52.4
Agate		ASC	Penguin	TOSS	N/A	48	1,728	6,144	58.1	
SCF Totals	Systems	8							25,627.1	729.3
Combined Totals		20							37,587.5	2,873.2

MVAPICH

System	Top500 Rank	Program	Manufacture / Model	OS	Inter-connect	Nodes	Cores	Memory (GB)	Peak TFLOP/s	TFLOP/s (GPUs)
Unclassified Network (OCF)										
Vulcan	23	ASC+M&IC+HPCIC	IBM BGQ	RHEL/CNK	5D Torus	24,576	393,216	393,216	5,033.2	
Cab (TLCC2)		ASC+M&IC+HPCIC	Appro	TOSS	IB QDR	1,296	20,736	41,472	426.0	
Quartz	46	ASC+M&IC	Penguin	TOSS	Omni-Path	2,688	96,768	344,064	3251.4	
RZTopaz		ASC	Penguin	TOSS	Omni-Path	768	27,648	98,304	929.0	
RZManta		ASC	IBM	RHEL	IB EDR	36	720	11,520	597.6	676.8
Ray		ASC+M&IC	IBM	RHEL	IB EDR	54	1,080	17,280	896.4	1015.2
Catalyst		ASC+M&IC	Cray	TOSS	IB QDR	324	7,776	41,472	149.3	
Syrah		ASC+M&IC	Cray	TOSS	IB QDR	324	5,184	20,736	107.8	
Surface		ASC+M&IC	Cray	TOSS	IB FDR	162	2,592	41,500	451.9	451.9
Borax		ASC+M&IC	Penguin	TOSS	N/A	48	1,728	6,144	58.1	
RZTrona		ASC	Penguin	TOSS	N/A	48	1,728	6,144	58.1	
Herd		M&IC	Appro	TOSS	IB DDR	9	256	1,088	1.6	
OCF Totals	**Systems**	**12**							**11,960.4**	**2,143.9**
Classified Network (SCF)										
Pinot(TLCC2, SNSI)		M&IC	Appro	TOSS	IB QDR	162	2,592	10,368	53.9	
Sequoia	5	ASC	IBM BGQ	RHEL/CNK	5D Torus	98,304	1,572,864	1,572,864	20132.7	
Zin (TLCC2)	217	ASC	Appro	TOSS	IB QDR	2,916	46,656	93,312	961.1	
Jade+Jadeita	45	ASC	Penguin	TOSS	Omni-Path	2,688	96,768	344,064	3251.4	
Mica		ASC	Penguin	TOSS	Omni-Path	384	13,824	49,152	464.5	
Shark		ASC	IBM	RHEL	IB EDR	36	720	11,520	597.6	676.9
Max		ASC	Appro	TOSS	IB FDR	324	5,184	82,944	107.8	52.4
Agate		ASC	Penguin	TOSS	N/A	48	1,728	6,144	58.1	
SCF Totals	**Systems**	**8**							**25,627.1**	**729.3**
Combined Totals		**20**							**37,587.5**	**2,873.2**

Why MVAPICH?

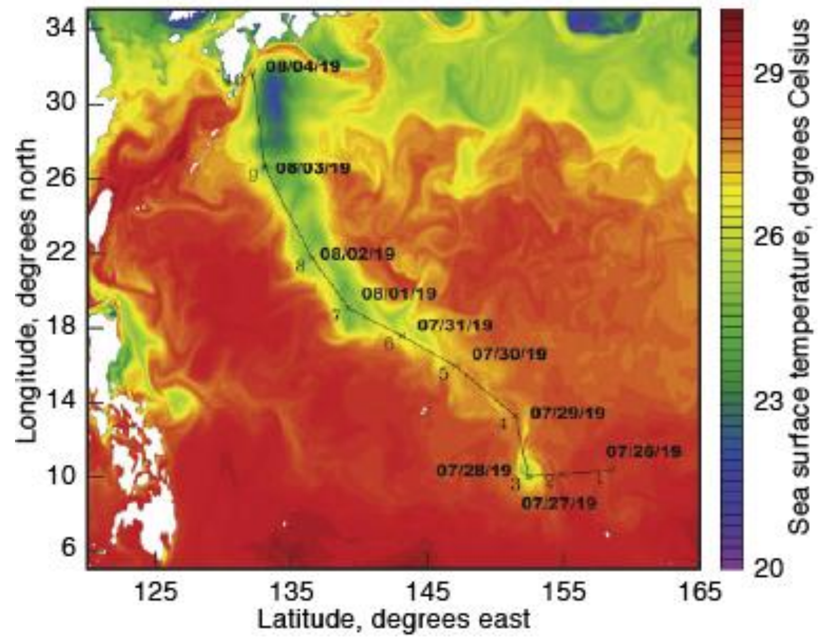
- First MPI available for IB, enabled IB for HPC
- Reliable and proven
- Fastest for many users
- Familiarity with MPICH code base
- Acceptance of feedback and patches
- Good ties and communication with OSU

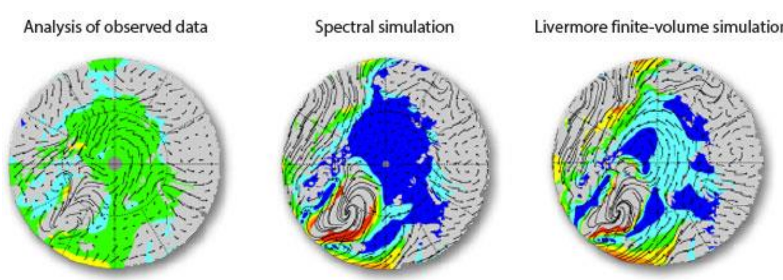
Science with MVAPICH

Climate Change

Climate scientist Ben Santer adds, “We know that Earth’s climate system is going to experience profound changes, such as large-scale warming and moistening of the atmosphere, rising sea levels, retreat of snow and sea-ice cover, and increases in the frequency and intensity of heat waves, but the regional and seasonal details of these changes are much fuzzier.”

Predicting these details with precision and confidence and delivering information that can help countries and communities make resource-planning decisions will require enhanced models and exaflop-scale computing capabilities.





https://str.llnl.gov/september-2015/carnes UCRL-TR-52000-15-9

Carbon Capture

- Public toolset to evaluate different carbon capture technologies
- The CCSI toolset addresses key industry challenges such as gaining a better understanding of sources of error (or uncertainty) in process-simulation results, quantifiying and reducing that uncertainty, and assessing the risks of scaling up a particular technology.

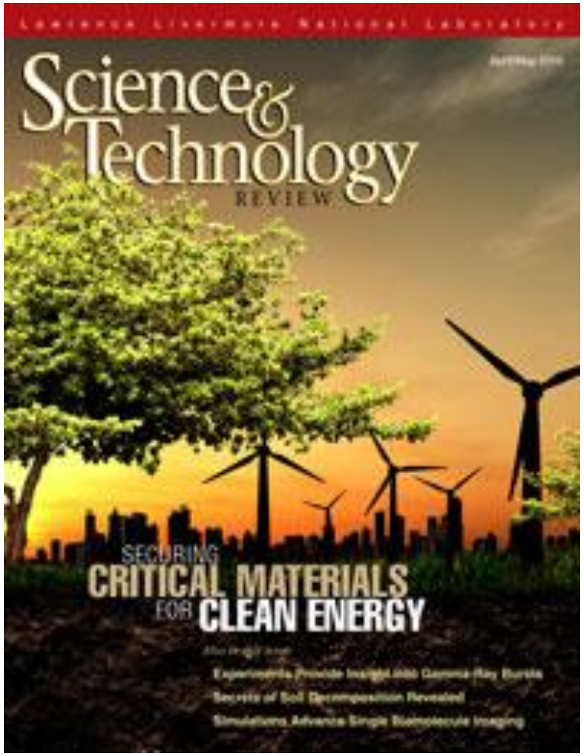
https://str.llnl.gov/january-2017/tong UCRL-TR-52000-17-1/2

Rare Earth Elements for Renewable Energy

The Department of Energy's Critical Materials Institute (CMI) is working to ensure the nation has adequate supplies of certain scarce materials that are essential to the U.S. clean-energy industry.

These materials are found in a wide array of products, including magnets, catalysts, metallurgical additives, phosphors, polishing powders, and ceramics.

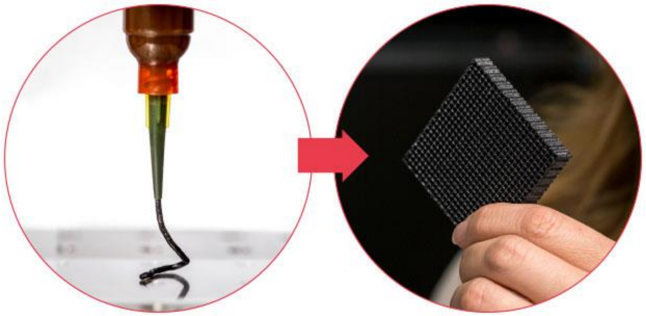
Livermore scientists are supporting CMI objectives by developing new alloys and substitute materials that reduce the need for rare-earth elements in high-efficiency motors, magnets, and fluorescent lightbulbs, as well as producing novel methods to reuse and recycle existing materials.



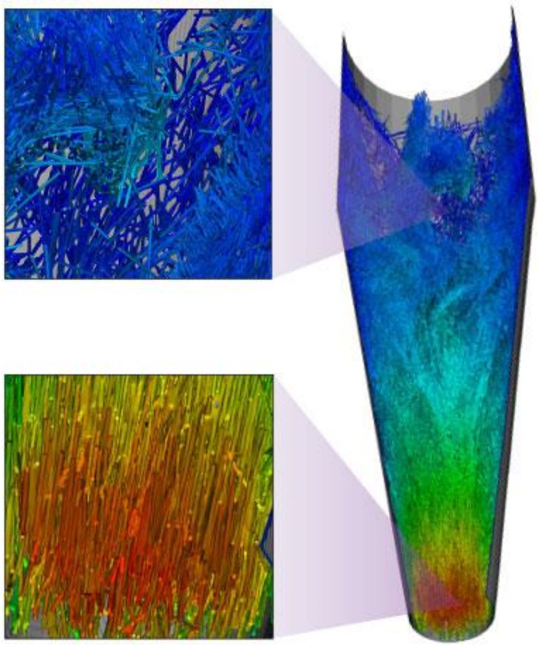
https://str.llnl.gov/april-2016/schwegler UCRL-TR-52000-16-4/5

Additive Manufacturing

Livermore chemist James Lewicki says, “Carbon fiber is the ultimate structural material. If we could make everything out of carbon fiber, we probably would, but it’s been waiting in the wings for years because it’s so difficult to make in complex shapes.”



Fluid analyst Yuliya Kanarska adds, “With our code, we can simulate the evolution of the fiber orientations in 3D under different printing conditions to find the optimal fiber length and optimal performance.”



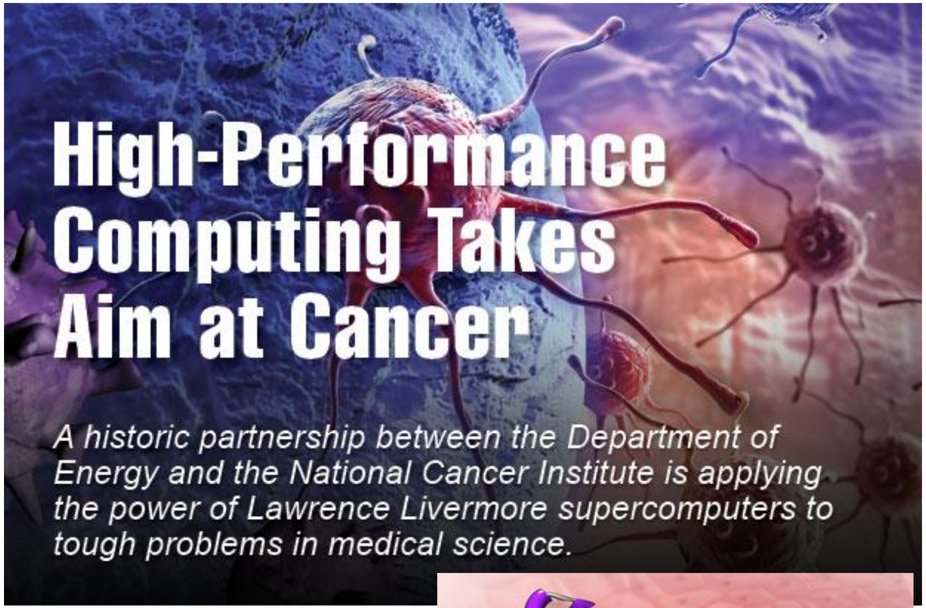
Simulation results both validated and explained what was observed experimentally—that with the right ingredients in the right ratio and the right nozzle size and shape, the resin can efficiently deliver carbon fibers without clogging the printer.

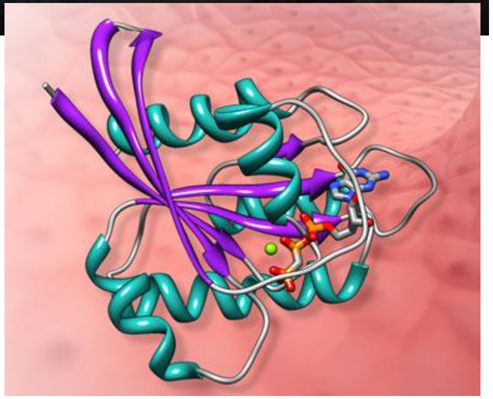
https://str.llnl.gov/june-2017/lewicki UCRL-TR-52000-17-6

Cancer Research

A historic partnership between the Department of Energy (DOE) and the National Cancer Institute (NCI) is applying the formidable computing resources at Livermore and other DOE national laboratories to advance cancer research and treatment.

Announced in late 2015, the effort will help researchers and physicians better understand the complexity of cancer, choose the best treatment options for every patient, and reveal possible patterns hidden in vast patient and experimental data sets.

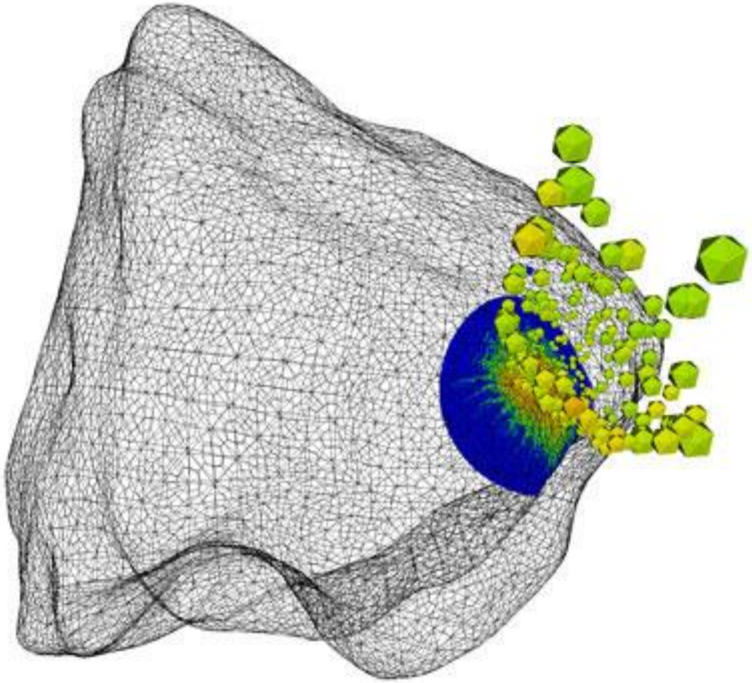


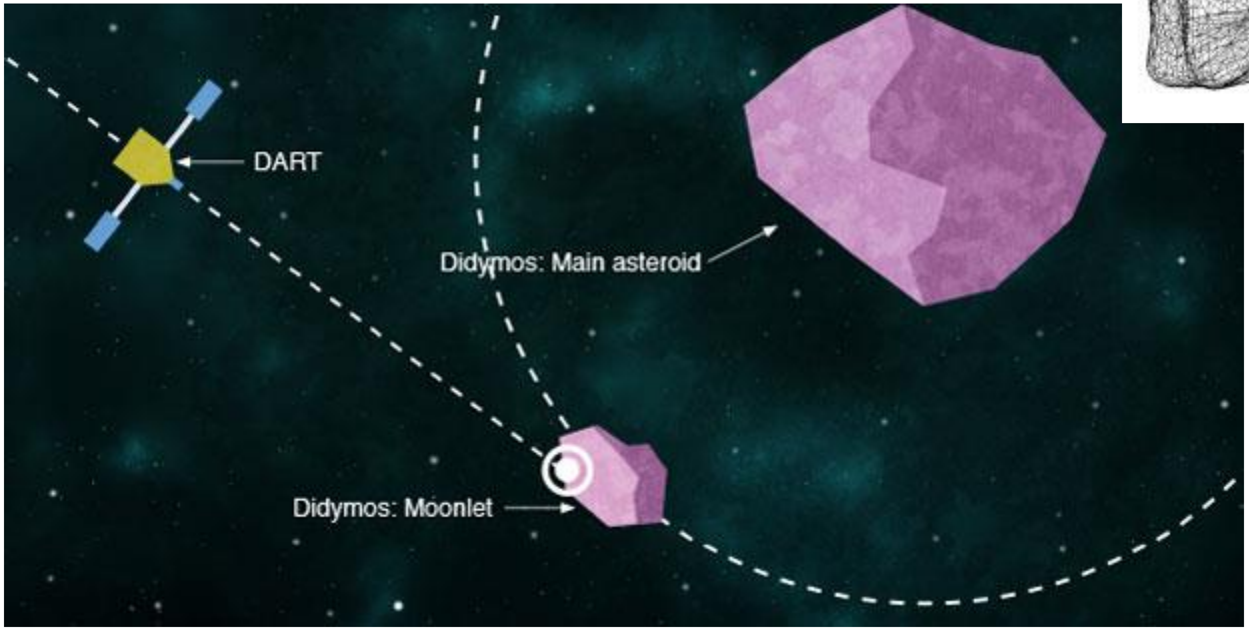


https://str.llnl.gov/november-2016/streitz UCRL-TR-52000-16-10/11

Asteroid Deflection

- Model how asteroid trajectory changes after impact from spacecraft
- DART: Joint mission with JHU APL to launch a test in 2020





http://youtu.be/xXCxMeZ-yQo

https://str.llnl.gov/december-2016/syal UCRL-TR-52000-16-12

What’s next at LLNL?

Commodity Technology Systems (CTS-1) PSM2 in spades with MVAPICH

- Upgraded Linux Systems
- Delivery from 2016 through 2018
- ~6600 total nodes so far and counting
  - Dual socket, 18-core Intel Xeon E5-2695, 2.10GHz
  - Intel Omni-Path (PSM2) in tapered fat-tree
- MVAPICH2-2.2 selected as default MPI
- Systems to last 5+ years
- Many more years with MVAPICH

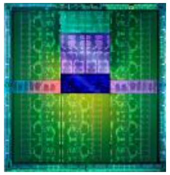
https://www.llnl.gov/news/labs-tap-silicon-valley-bolster-computing

The Sierra system that will replace Sequoia features a GPU-accelerated architecture

Components

IBM POWER9
- NVLink

NVIDIA Volta
- HBM2
- NVLink



Compute Node

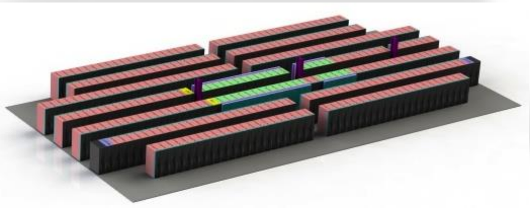
2 IBM POWER9 CPUs
4 or 6 NVIDIA Volta GPUs
NVMe-compatible PCIe 800GB SSD
512 GB DDR4
Globally addressable HBM2 associated with GPUs
Coherent Shared Memory

Compute Rack

Standard 19”
Warm water cooling

Compute System

3400 -4200 nodes
2.1 – 2.7 PB Memory
120 -150 PFLOPS
10 MW



Mellanox Interconnect
Dual-rail EDR Infiniband

GPFS File System

182 PB usable storage
2.5/1.8 TB/s R/W bandwidth

GPUS are coming… big time… finally!

- Sierra system will be loaded with GPUS
- Codes are porting to use GPUs now
  - Some doing direct CUDA programming
  - Many using programming abstractions like RAJA
- After porting, we expect high demand for GPUs on future systems

Needs More Parallel

RAJA is a C++ abstraction layer enabling portability with small disruption to application programming styles

The main goal of RAJA is to balance *performance*…

- Augment compiler's ability to optimize C++ code
  - Enable work-arounds when performance is not what's expected
- Support various forms of fine-grained (on-node) parallelism
  - Facilitate use of common programming models (OpenMP, CUDA, TBB, OpenACC, …)

… and *developer productivity*

- Applications maintain single-source kernels
  - Don't bind an application to a particular programming model
  - Easy integration with application data structures and algorithms
- Clear separation of responsibilities
  - **RAJA:** Execute loop iterations, encapsulate hardware & programming model details
  - **Application:** Select loop iteration patterns and execution policies with RAJA API

RAJA development is currently driven by the needs of ATDM/ASC applications at LLNL and ECP collaborators

RAJA concepts “orthogonalize” and encapsulate loop execution details

C-style for-loop

```
double* x ; double* y ;
double a, tsum = 0, tmin = MYMAX;

for ( int i = begin; i < end; ++i ) {
    y[i] += a * x[i] ;
    tsum += y[i] ;
    if ( y[i] < tmin ) tmin = y[i];
}
```

- **RAJA decouples loop iteration and loop body**
  - Iterations are “tasks” – aggregate, reorder, etc.
- **RAJA Concepts:**
  - **Patterns:** forall, forallN, reduce, scan
  - **Policies:** sequential, simd, openmp, cuda, ...
  - **Index:** iterations – aggregate, reorder, tile, ...

RAJA-style loop

```
double* x ; double* y ;
double a ;
RAJA::SumReduction<reduce_policy, double> tsum(0);
RAJA::MinReduction<reduce_policy, double> tmin(MYMAX);

RAJA::forall<exec_policy > ( IndexSet , [=] (int i) {
    y[i] += a * x[i] ;
    tsum += y[i];
    tmin.min( y[i] );
} );
```

Execution patterns & policies
(scheduling, PM choice, etc.)

IndexSets
(iteration space, ordering, etc.)

Portable Reduction types

https://github.com/LLNL/RAJA

Loop body is mostly unchanged (C++ lambda function).

Why we prefer RAJA over alternatives

- "Light touch"
  - Works with our existing application data structures & algorithms – loop bodies require little change if any
- "Low barrier to entry"
  - Add parallelism selectively & incrementally without changing the way existing algorithms appear in source code
- "Application-facing design philosophy"
  - Concepts are easy to grasp for (non-CS) application developers
  - Constructs map naturally to apps and are easy to customize
- "Performance"
  - RAJA does well with "streaming" kernels that are prevalent in LLNL codes
  - Designed for coarse-grained synchronization – can greatly reduce resource contention and memory synchronization vs. finer-grained techniques

RAJA performance on CoMD across platforms (from 2015)

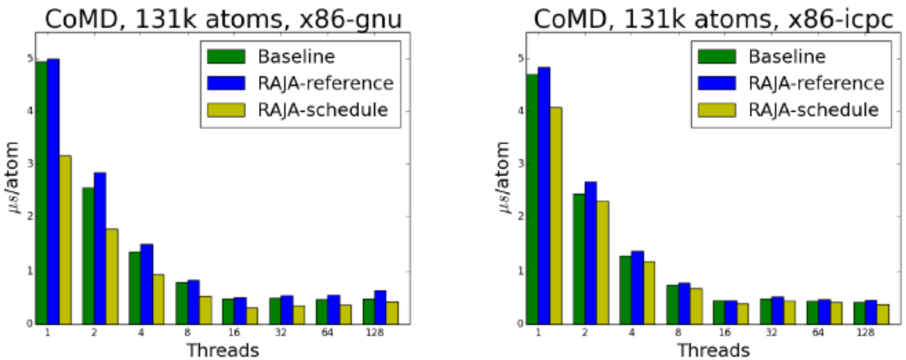


Figure 1.11: CoMD on x86, EAM force computation

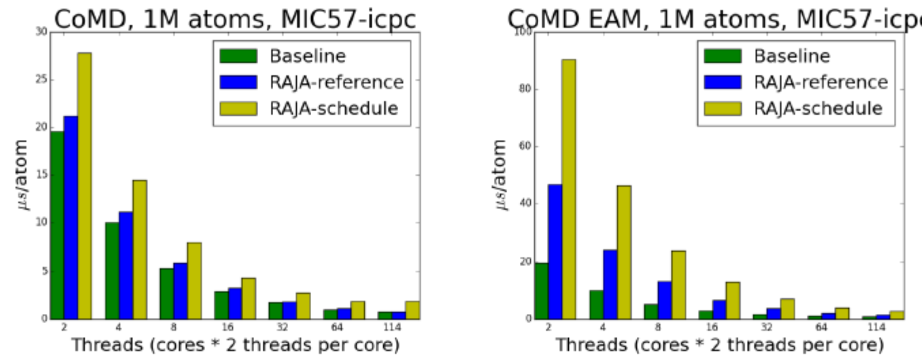


Figure 1.13: CoMD on MIC with Intel compiler, LJ and EAM forces. Problem size 1M atoms.

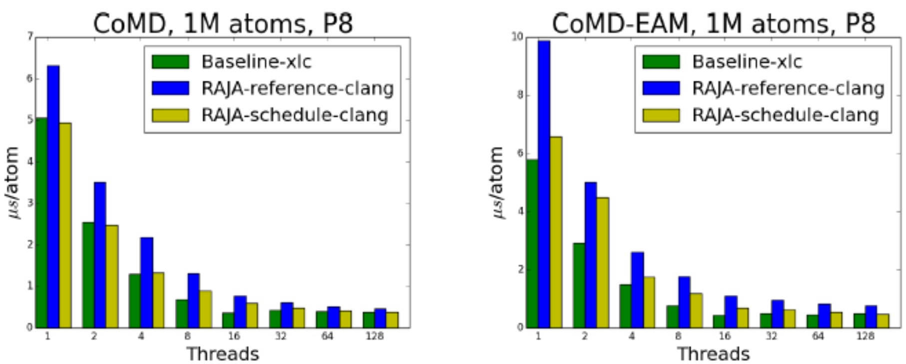


Figure 1.12: CoMD on Power8 with xlC and Clang compilers, LJ and EAM forces

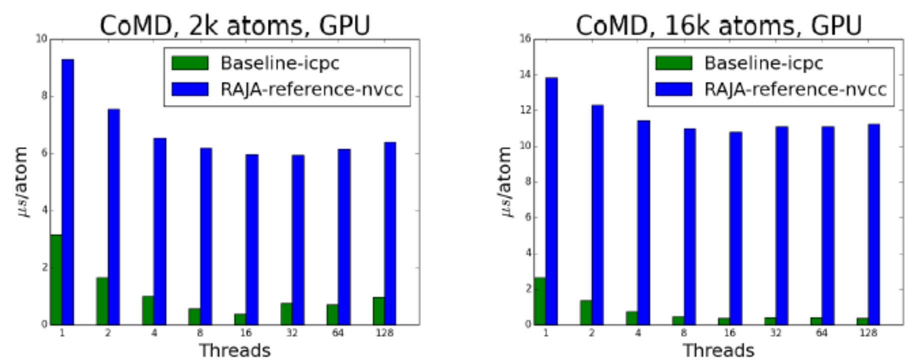


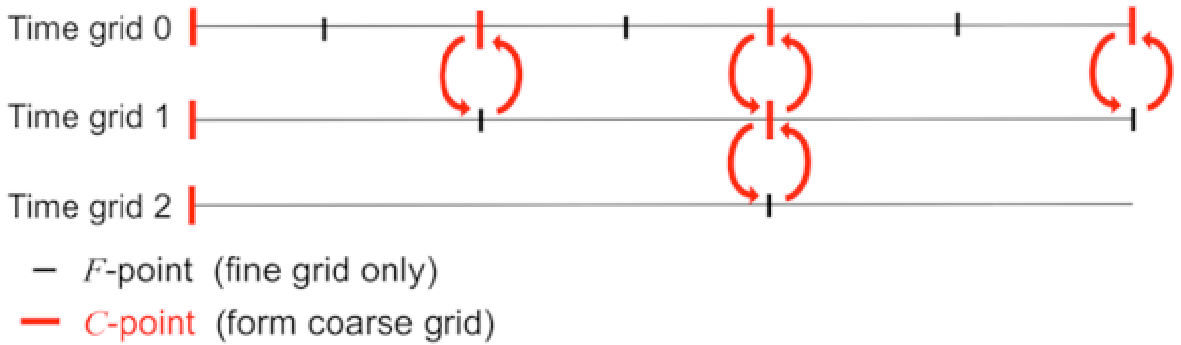
Figure 1.14: CoMD on GPU with NVIDIA compiler

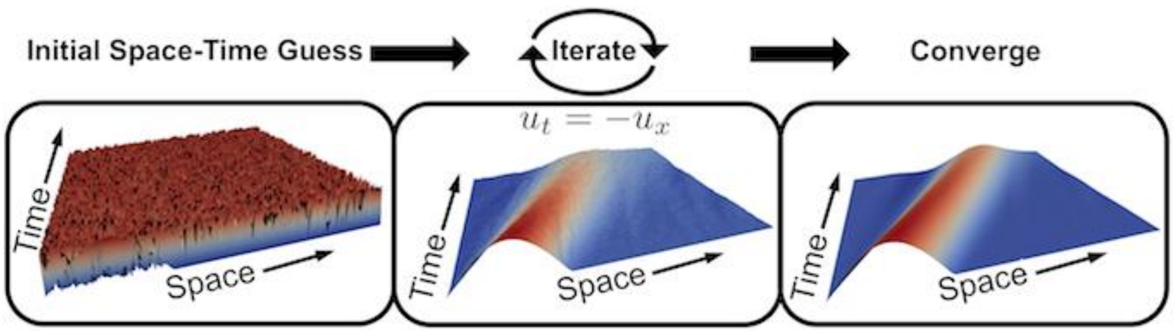
https://software.llnl.gov/RAJA/_static/RAJAOverview-Trilab-09.2015_LLNL-TR-677453.pdf

Xbraid – Parallelizing across time and space

Developing algorithms that account for known architectural trends such as reducing data movement and allowing for many more actions to happen in parallel, or simultaneously.

The latter is particularly critical because future speedups for applications will likely happen only through greater parallelism.





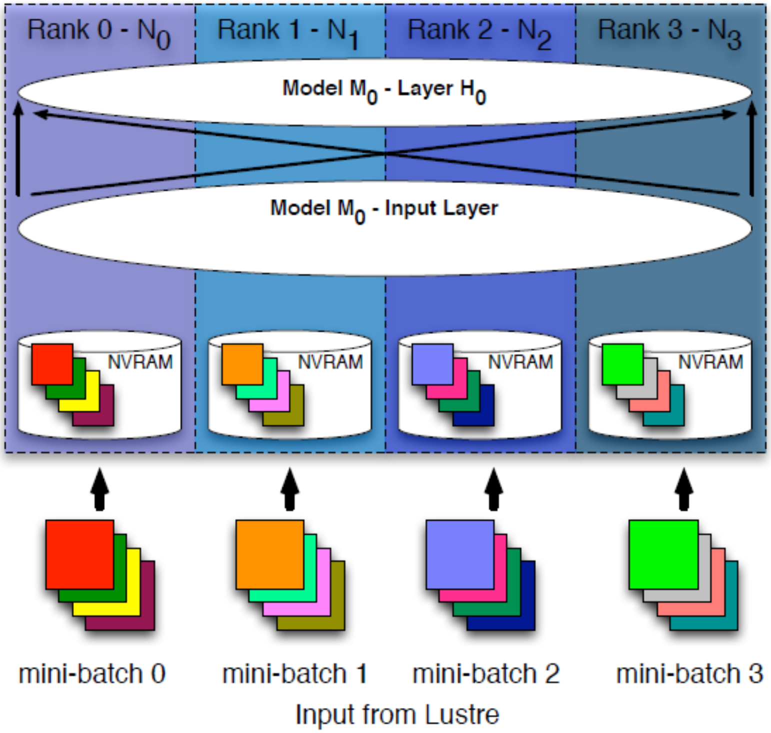
https://computation.llnl.gov/projects/parallel-time-integration-multigrid

https://str.llnl.gov/september-2016/diachin UCRL-TR-52000-16-9

Applying HPC to Deep Learning

Applying HPC to Deep Learning: Livermore Big Artificial Neural Network (LBANN)

- Framework for training deep neural networks on HPC systems using MPI
- Supports
  - data-parallel
  - model-parallel
  - multiple models
- Distributed Matrix Multiply with Elemental Linear Algebra Library
- CPUs and/or GPUs
- Parallel I/O and data augmentation
  - Uses node-local storage if available
  - Optimized for Lustre

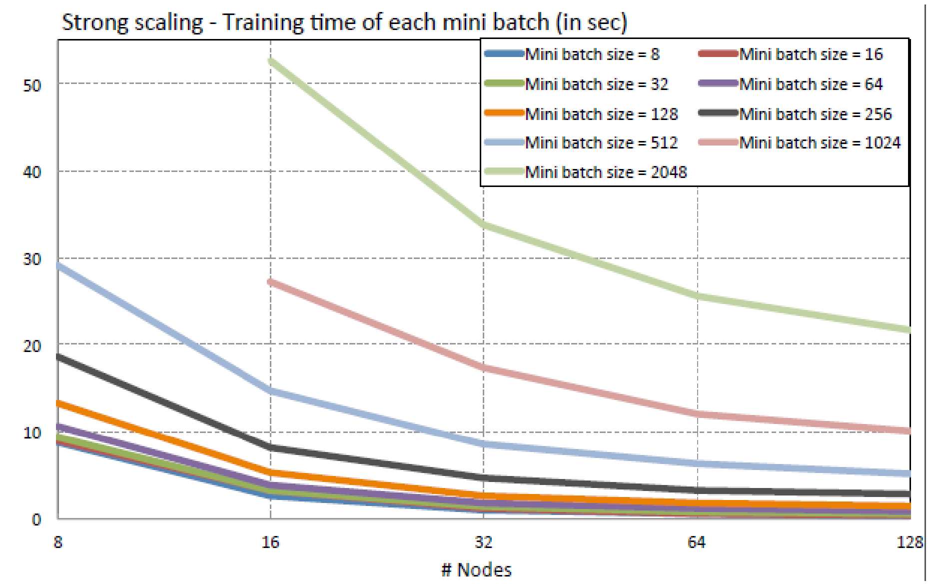


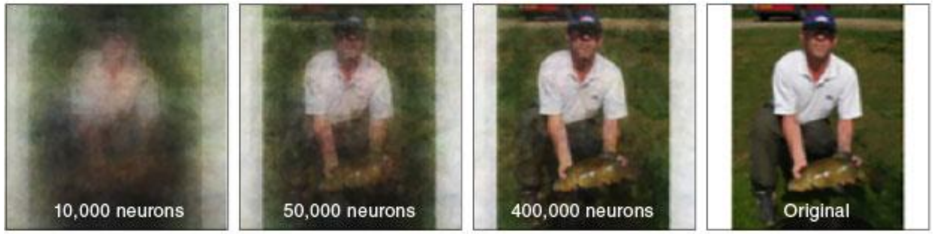
https://github.com/LLNL/lbann

https://str.llnl.gov/june-2016/chen UCRL-TR-52000-16-6

LBANN strong scaling (distributed matrix multiply with MPI)

- # nodes versus mini-batch training time
- Processing multiple images per step
- Test
  - 50K neurons
  - 8-128 nodes, 12 ranks per node
  - Mini-batch sizes from 8-2048 images
- Large mini-batches benefit greatly from additional nodes
- Smaller mini-batches have limited improvement beyond 16 or 32 nodes
  - Insufficient work to effectively amortize communication overheads





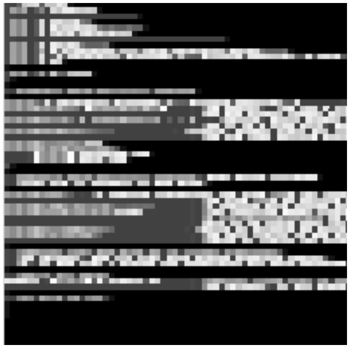
Applying Deep Learning To HPC

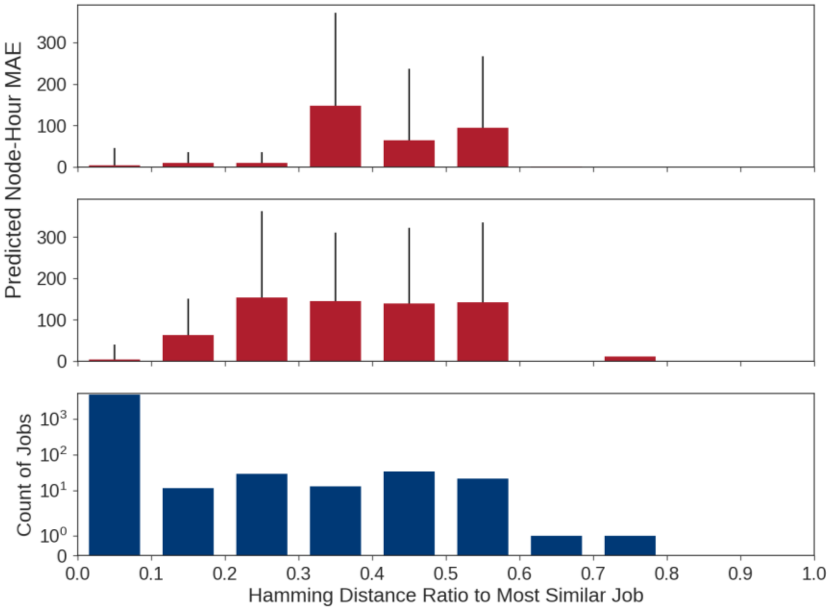
Applying Deep Learning to HPC: 1. Predicting HPC Job Behavior

- Apply machine learning to predict HPC job behavior like run time, IO, networking, and power
  - Better backfill
  - Schedule jobs for IO and power
- Use DNN w/ CNN on inputs like job scripts, job environment, input deck
- Michael R. Wyatt II, Michela Taufer (Advisor) University of Delaware

```
#!/bin/bash
#PBS -l partition=cab
#PBS -l nodes=4
#PBS -l walltime=16:00:00
#PBS -q pbatch

cd $HOME/project_A
srun -n 64 ./GEOS -i
Prop_bx_2a.xml
```

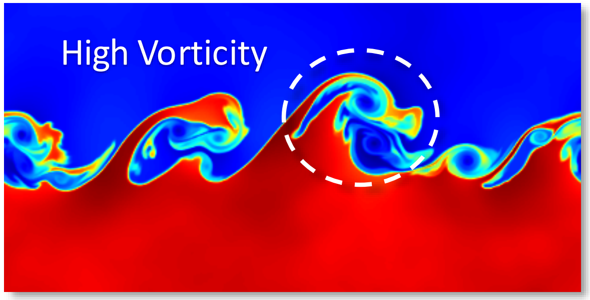




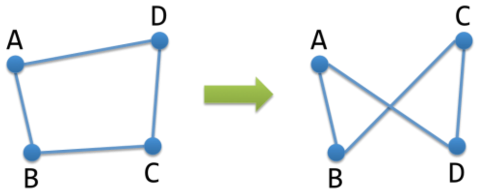
Applying Deep Learning to HPC:
2. Self-driving codes

ALE simulations use dynamic meshes to simulate complex dynamics

- They fail frequently
- Mesh geometry: *mesh zone tangling*
- Physical quantities: *anomalous hot spots*



Goal: Apply machine learning to predict simulation failures and proactively avoid them



Mesh zone tangling

Feasibility demonstration: *Successfully predicted and automatically avoided* different mesh tangling conditions using 3 test cases – Helium bubble, shock tube, simple hohlraum

Challenges to MVAPICH

System	Top500 Rank	Program	Manufacture / Model	OS	Inter-connect	Nodes	Cores	Memory (GB)	Peak TFLOP/s	TFLOP/s (GPUs)
Unclassified Network (OCF)										
Vulcan	23	ASC+M&IC+HPCIC	IBM BGQ	RHEL/CNK	5D Torus	24,576	393,216	393,216	5,033.2	
Cab (TLCC2)		ASC+M&IC+HPCIC	Appro	TOSS	IB QDR	1,296	20,736	41,472	426.0	
Quartz	46	ASC+M&IC	Penguin	TOSS	Omni-Path	2,688	96,768	344,064	3251.4	
RZTopaz		ASC	Penguin	TOSS	Omni-Path	768	27,648	98,304	929.0	
RZManta		ASC	IBM	RHEL	IB EDR	36	720	11,520	597.6	676.8
Ray		ASC+M&IC	IBM	RHEL	IB EDR	54	1,080	17,280	896.4	1015.2
Catalyst		ASC+M&IC	Cray	TOSS	IB QDR	324	7,776	41,472	149.3	
Syrah		ASC+M&IC	Cray	TOSS	IB QDR	324	5,184	20,736	107.8	
Surface		ASC+M&IC	Cray	TOSS	IB FDR	162	2,592	41,500	451.9	451.9
Borax		ASC+M&IC	Penguin	TOSS	N/A	48	1,728	6,144	58.1	
RZTrona		ASC	Penguin	TOSS	N/A	48	1,728	6,144	58.1	
Herd		M&IC	Appro	TOSS	IB DDR	9	256	1,088	1.6	
OCF Totals	Systems	12							11,960.4	2,143.9
Classified Network (SCF)										
Pinot(TLCC2, SNSI)		M&IC	Appro	TOSS	IB QDR	162	2,592	10,368	53.9	
Sequoia	5	ASC	IBM BGQ	RHEL/CNK	5D Torus	98,304	1,572,864	1,572,864	20132.7	
Zin (TLCC2)	217	ASC	Appro	TOSS	IB QDR	2,916	46,656	93,312	961.1	
Jade+Jadeita	45	ASC	Penguin	TOSS	Omni-Path	2,688	96,768	344,064	3251.4	
Mica		ASC	Penguin	TOSS	Omni-Path	384	13,824	49,152	464.5	
Shark		ASC	IBM	RHEL	IB EDR	36	720	11,520	597.6	676.9
Max		ASC	Appro	TOSS	IB FDR	324	5,184	82,944	107.8	52.4
Agate		ASC	Penguin	TOSS	N/A	48	1,728	6,144	58.1	
SCF Totals	Systems	8							25,627.1	729.3
Combined Totals		20							37,587.5	2,873.2

- Request to port MVAPICH to CORAL
- CORAL EA
  - POWER8
  - Dual Mellanox EDR
  - NVIDIA Pascal
- CORAL (Sierra)
  - POWER9
  - Dual Mellanox EDR
  - NVIDIA Volta

MVAPICH diversity within LC

- RPM needed for each color
- Careful to install correct RPM on each machine
- Static linking: need to relink app for each machine (and use correct one)
- Similar problem for apps using Spack
- Lots of builds and difficult to manage

PSM
PSM2
Mellanox + NVIDIA
Shared Mem
POWER + Mellanox + NVIDIA

System	Top500 Rank	Program	Manufacture / Model	OS	Inter-connect	Nodes	Cores	Memory (GB)	Peak TFLOP/s	TFLOP/s (GPUs)
Unclassified Network (OCF)										
Vulcan	23	ASC+M&IC+HPCIC	IBM BGQ	RHEL/CNK	5D Torus	24,576	393,216	393,216	5,033.2	
Cab (TLCC2)		ASC+M&IC+HPCIC	Appro	TOSS	IB QDR	1,296	20,736	41,472	426.0	
Quartz	46	ASC+M&IC	Penguin	TOSS	Omni-Path	2,688	96,768	344,064	3251.4	
RZTopaz		ASC	Penguin	TOSS	Omni-Path	768	27,648	98,304	929.0	
RZManta		ASC	IBM	RHEL	IB EDR	36	720	11,520	597.6	676.8
Ray		ASC+M&IC	IBM	RHEL	IB EDR	54	1,080	17,280	896.4	1015.2
Catalyst		ASC+M&IC	Cray	TOSS	IB QDR	324	7,776	41,472	149.3	
Syrah		ASC+M&IC	Cray	TOSS	IB QDR	324	5,184	20,736	107.8	
Surface		ASC+M&IC	Cray	TOSS	IB FDR	162	2,592	41,500	451.9	451.9
Borax		ASC+M&IC	Penguin	TOSS	N/A	48	1,728	6,144	58.1	
RZTrona		ASC	Penguin	TOSS	N/A	48	1,728	6,144	58.1	
Herd		M&IC	Appro	TOSS	IB DDR	9	256	1,088	1.6	
OCF Totals	Systems	12							11,960.4	2,143.9
Classified Network (SCF)										
Pinot(TLCC2, SNSI)		M&IC	Appro	TOSS	IB QDR	162	2,592	10,368	53.9	
Sequoia	5	ASC	IBM BGQ	RHEL/CNK	5D Torus	98,304	1,572,864	1,572,864	20132.7	
Zin (TLCC2)	217	ASC	Appro	TOSS	IB QDR	2,916	46,656	93,312	961.1	
Jade+Jadeita	45	ASC	Penguin	TOSS	Omni-Path	2,688	96,768	344,064	3251.4	
Mica		ASC	Penguin	TOSS	Omni-Path	384	13,824	49,152	464.5	
Shark		ASC	IBM	RHEL	IB EDR	36	720	11,520	597.6	676.9
Max		ASC	Appro	TOSS	IB FDR	324	5,184	82,944	107.8	52.4
Agate		ASC	Penguin	TOSS	N/A	48	1,728	6,144	58.1	
SCF Totals	Systems	8							25,627.1	729.3
Combined Totals		20							37,587.5	2,873.2

Building MPI: A Nightmare of Permutations

- Multiple compilers
  - GNU, Intel, PGI
  - several versions of each
- Multiple MPI implementations
  - MVAPICH, MVAPICH2, Open MPI
  - 2-3 versions of each
  - normal + debug
- Multiple system types

MPI	Open MPI	MVAPICH2
Compilers	3	3
x MPI Versions	3	3
x (Normal + Debug)	2	2
x Platforms	1	4
= Total	18	72 !!!

Compiler and Library Observations

	GNU MVAPICH	GNU OpenMPI	Intel-MPI	Intel MVAPICH	Intel* OpenMPI	PGI OpenMPI
Allreduce (36ppn)	✓		✗	✓	✗	✗
MPI_Send MPI_Recv	✓			✓		
RMA Get	✓	✓	✗	✓	✓	✓
RMA Put (low PPN)	✗	✗	✓	✗	✗	✗
RMA Put (high PPN)	✓	✓	✗	✓	✓	✓
Consistent Put & Get			✓			

empty cells are neither good nor bad

HPC apps needs from MPI

- Thread multiple becoming more common
- Non-blocking collectives w/ async progress
- Network offload where possible
  - Collectives
  - Tag matching
  - Rendezvous handshake
- Reduce pt2pt latency (lots of latency bound apps)

Deep learning needs from MPI

- Improve Allreduce algorithms for user-defined datatypes/ops
  - Allreduce on compressed data
- Improve support for large-bandwidth messages
- Support for non-blocking Allreduce and pt-2-pt
  - Overlap messages with backprop steps
- Higher precision accumulate for low-precision inputs
  - e.g., 32-bit Allreduce on 16-bit data
- NCCL-like performance from MPI collectives

Thanks to MVAPICH and to the NOWLAB Nutcrackers!