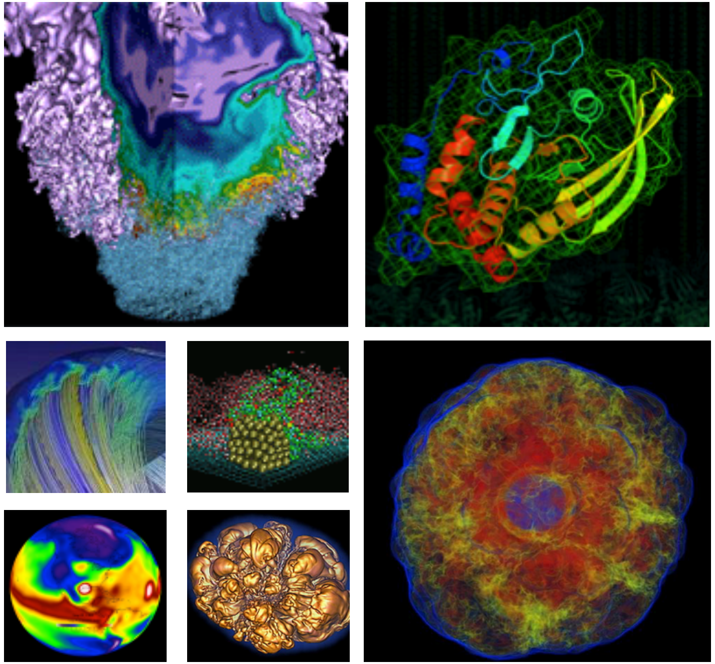
Many-Cores for the Masses: Optimizing Large Scale Science Applications on Cori



NERSC

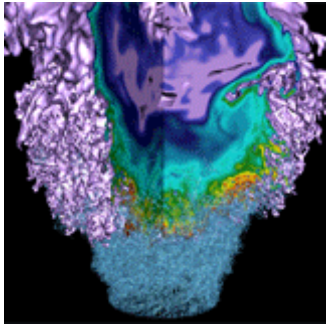
Richard Gerber
Jack Deslippe
Douglas Doerfler

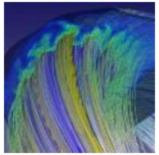
Intel HPC Developer Conference
November, 2016

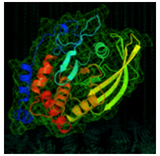
U.S. DEPARTMENT OF ENERGY | Office of Science

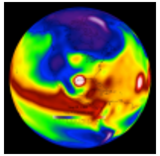
BERKELEY LAB

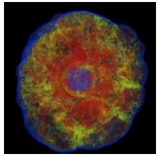
NERSC

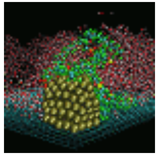








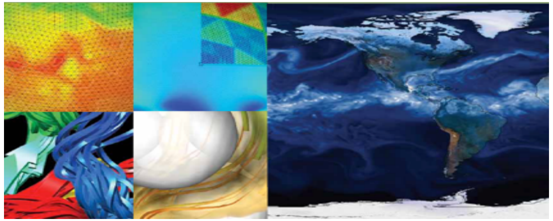




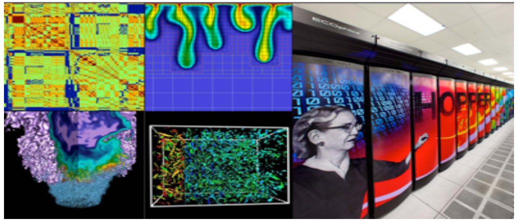
NERSC: the Mission HPC Facility for DOE Office of Science Research

U.S. DEPARTMENT OF ENERGY | Office of Science

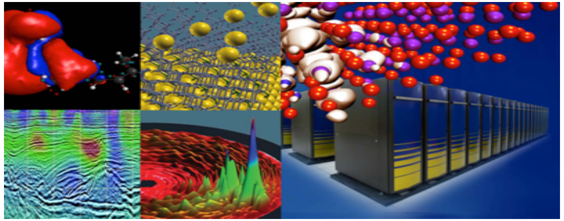
Largest funder of physical science research in U.S.



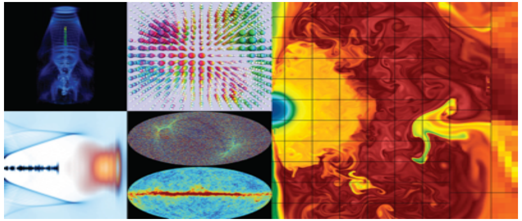
Bio Energy, Environment



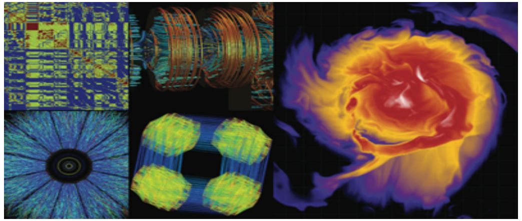
Computing



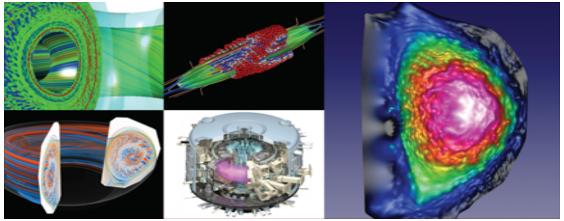
Materials, Chemistry, Geophysics



Particle Physics, Astrophysics



Nuclear Physics



Fusion Energy, Plasma Physics

6,000 users, 700 projects, 700 codes, 48 states, 40 countries, universities & national labs

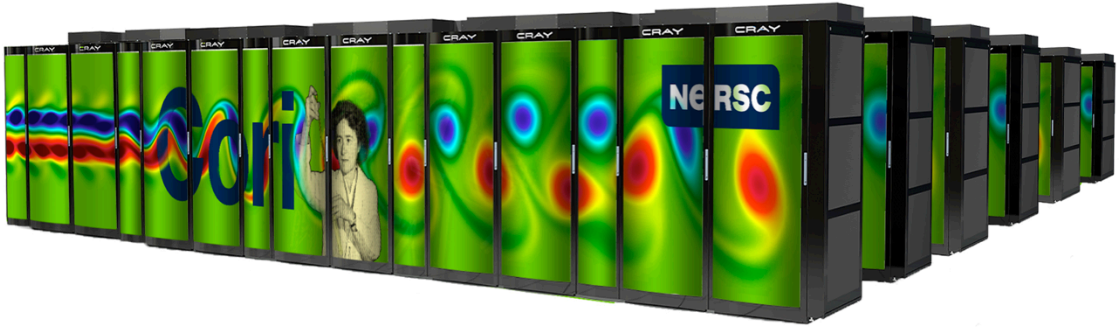
Current Production Systems

Edison

5,560 Ivy Bridge Nodes / 24 cores/node
133 K cores, 64 GB memory/node
Cray XC30 / Aries Dragonfly interconnect
6 PB Lustre Cray Sonexion scratch FS

Cori Haswell Nodes

1,900 Haswell Nodes / 32 cores/node
52 K cores, 128 GB memory/node
Cray XC40 / Aries Dragonfly interconnect
24 PB Lustre Cray Sonexion scratch FS
1.5 PB Burst Buffer



Cori Xeon Phi KNL Nodes

Cray XC40 system with 9,300 Intel Knights Landing compute nodes

68 cores / 96 GB DRAM / 16 GB HBM

Support the entire Office of Science research community

Begin to transition workload to energy efficient architectures

Data Intensive Science Support

10 Haswell processor cabinets (Phase 1)

NVRAM Burst Buffer 1.5 PB, 1.5 TB/sec

30 PB of disk, >700 GB/sec I/O bandwidth

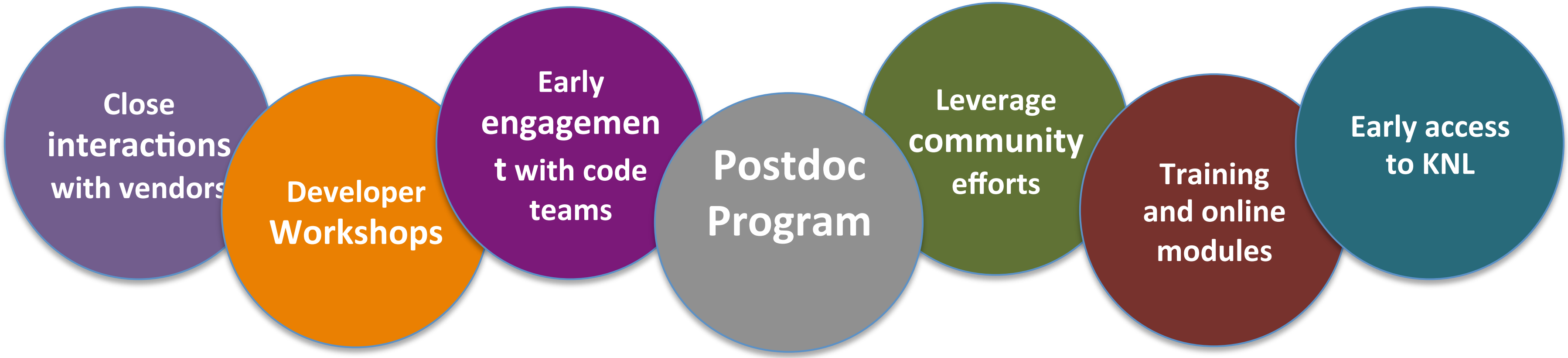
Integrated with Cori Haswell nodes on Aries network for data / simulation / analysis on one system

NERSC Exascale Scientific Application Program (NESAP)

Goal: Prepare DOE Office of Science users for many core

Partner closely with ~20 application teams and apply lessons learned to broad NERSC user community

NESAP activities include:



Close interactions with vendors

Developer Workshops

Early engagement with code teams

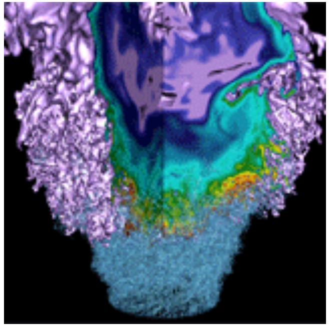
Postdoc Program

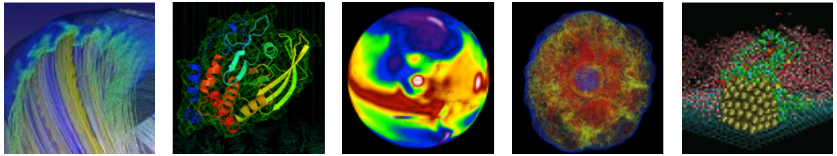
Leverage community efforts

Training and online modules

Early access to KNL

NESAP - NERSC Exascale Science Apps Program





What is different about Cori?

Edison ("Ivy Bridge):

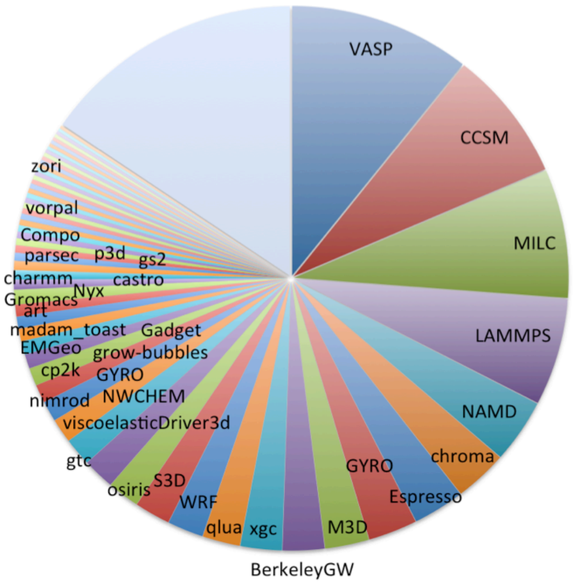
- 5576 nodes
- 12 physical cores per node
- 24 virtual cores per node
- 2.4 - 3.2 GHz
- 8 double precision ops/cycle
- 64 GB of DDR3 memory (2.5 GB per physical core)
- ~100 GB/s Memory Bandwidth

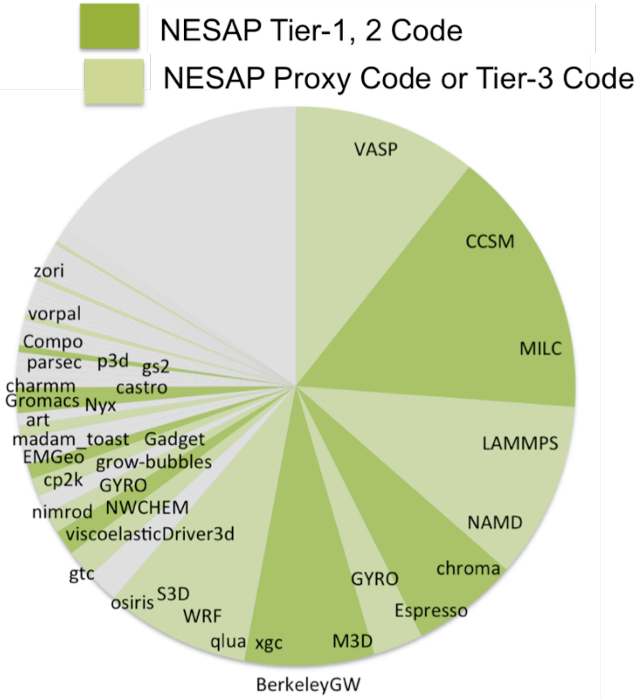
Cori ("Knights Landing"):

- 9304 nodes
- 68 physical cores per node
- 272 virtual cores per node
- 1.4 - 1.6 GHz
- 32 double precision ops/cycle
- 16 GB of fast memory
  96GB of DDR4 memory
- Fast memory has 400 - 500 GB/s

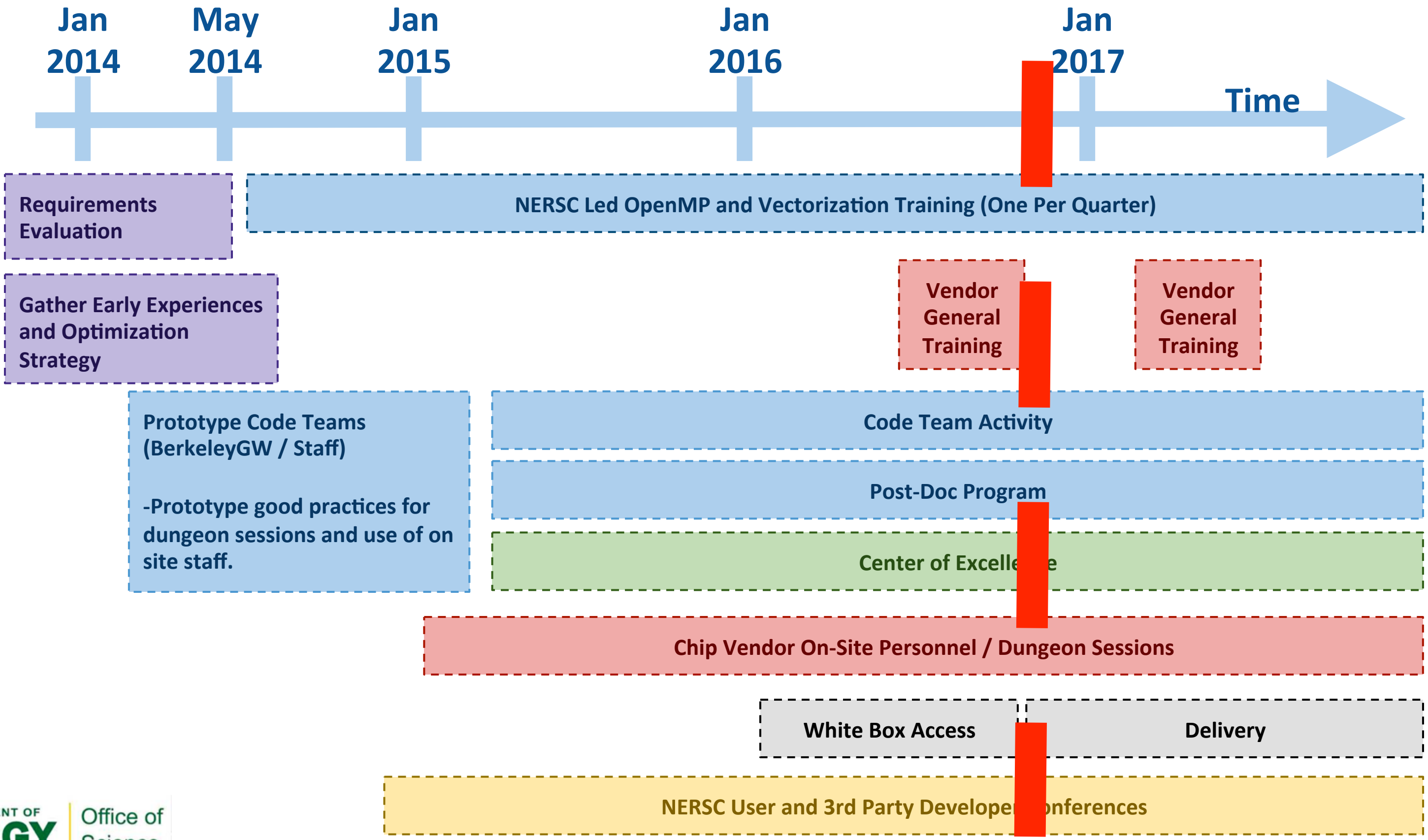
Code Coverage in NERSC App Readiness

Breakdown of Application Hours on Hopper and Edison





Timeline



Jan 2014

May 2014

Jan 2015

Jan 2016

Jan 2017

Time

Requirements Evaluation

NERSC Led OpenMP and Vectorization Training (One Per Quarter)

Gather Early Experiences and Optimization Strategy

Vendor General Training

Vendor General Training

Prototype Code Teams (BerkeleyGW / Staff)

-Prototype good practices for dungeon sessions and use of on site staff.

Code Team Activity

Post-Doc Program

Center of Excellence

Chip Vendor On-Site Personnel / Dungeon Sessions

White Box Access

Delivery

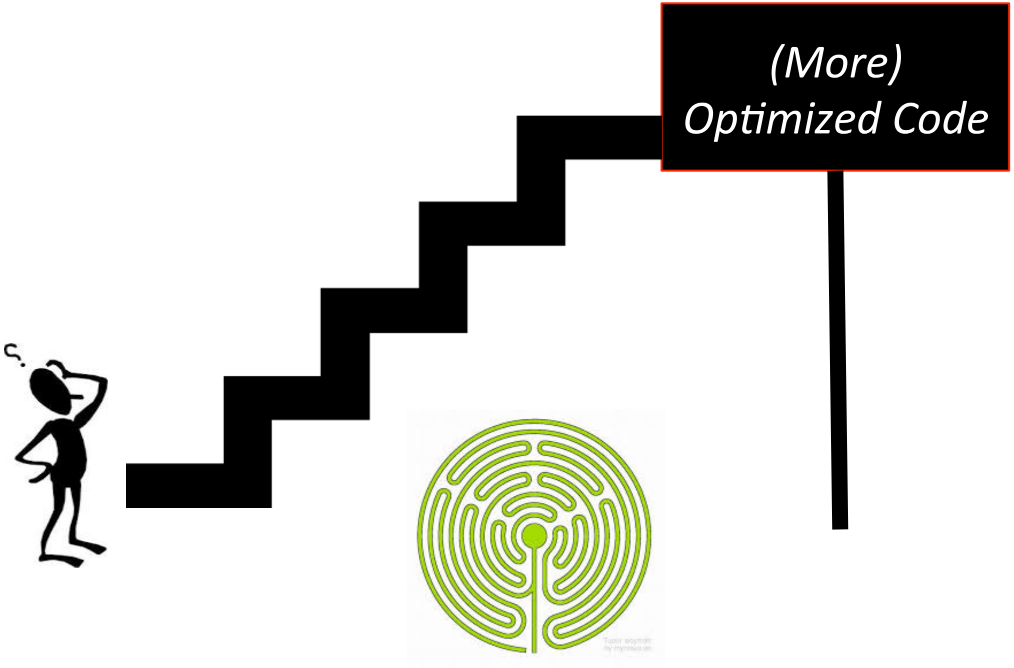
NERSC User and 3rd Party Developer Conferences

Optimizing Code For Cori is like:

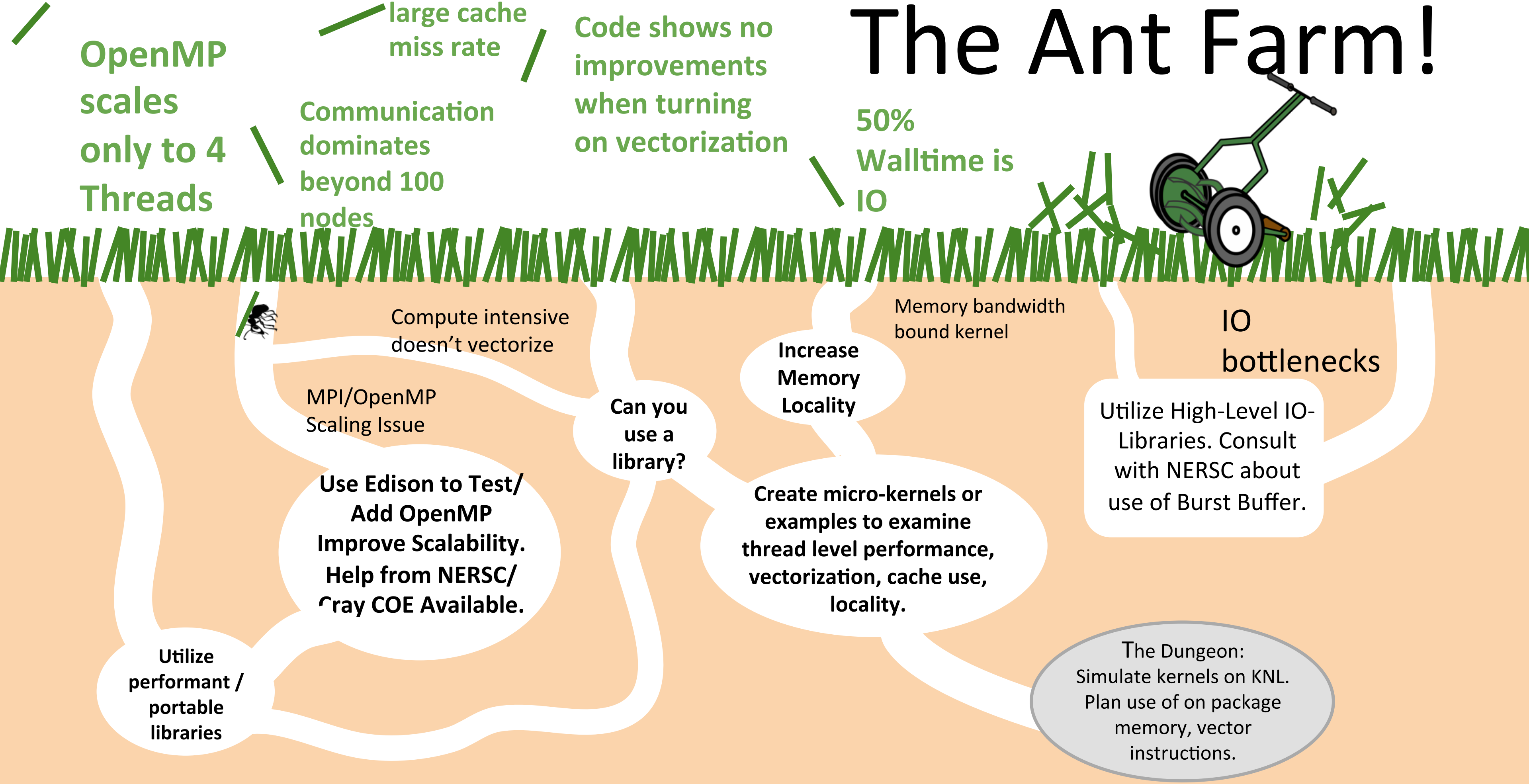
A. A Staircase ?

B. A Labyrinth ?

C. A Space Elevator?



The Ant Farm!



OpenMP scales only to 4 Threads

Communication dominates beyond 100 nodes

large cache miss rate

Code shows no improvements when turning on vectorization

50% Walltime is IO

Compute intensive doesn't vectorize

MPI/OpenMP Scaling Issue

Can you use a library?

Use Edison to Test/ Add OpenMP Improve Scalability. Help from NERSC/ Cray COE Available.

Utilize performant / portable libraries

Memory bandwidth bound kernel

Increase Memory Locality

Create micro-kernels or examples to examine thread level performance, vectorization, cache use, locality.

IO bottlenecks

Utilize High-Level IO-Libraries. Consult with NERSC about use of Burst Buffer.

The Dungeon: Simulate kernels on KNL. Plan use of on package memory, vector instructions.

Optimizing Code is hard….

It is easy to get bogged down in the weeds.

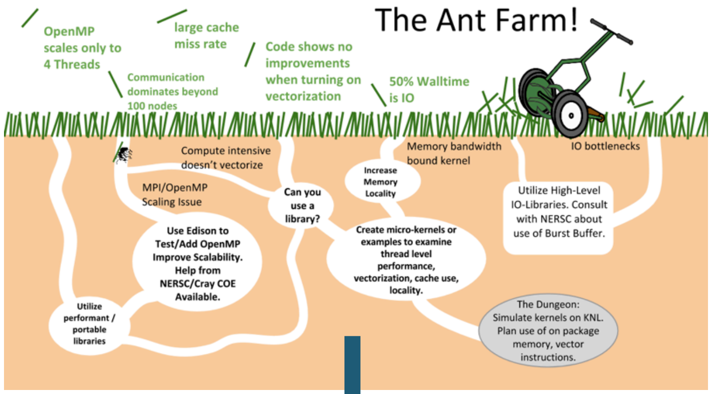
How do you know what part of an HPC system limits performance in your application?

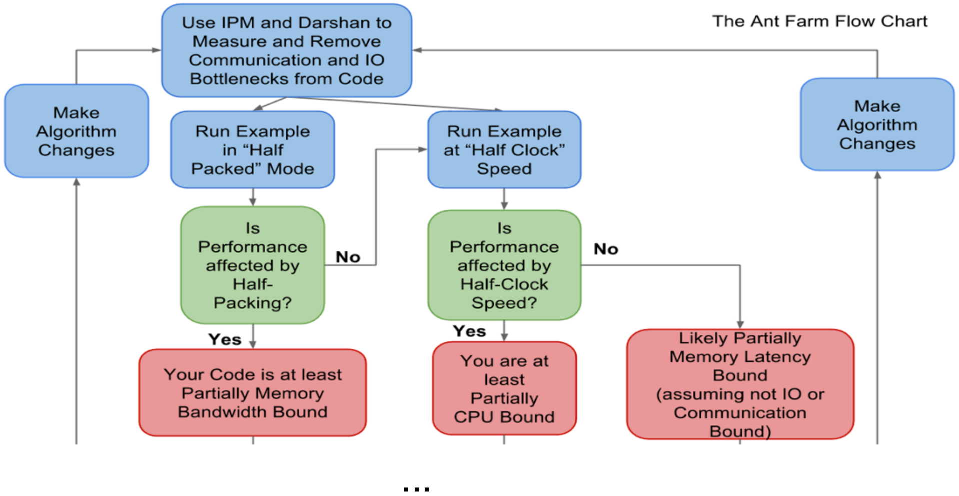
How do you know what new KNL feature to target?

Will vectorization help my performance?

NERSC distills the process for users into 3 important points for KNL:

1. Identify/exploit on-node shared-memory parallelism.
2. Identify/exploit on-core vector parallelism.
3. Understand and optimize memory bandwidth requirements with MCDRAM


The Ant Farm!
OpenMP scales only to 4 Threads
large cache miss rate
Communication dominates beyond 100 nodes
Code shows no improvements when turning on vectorization
50% Walltime is IO
Compute intensive doesn't vectorize
MPI/OpenMP Scaling Issue
Memory bandwidth bound kernel
IO bottlenecks
Increase Memory Locality
Can you use a library?
Utilize High-Level IO-Libraries. Consult with NERSC about use of Burst Buffer.
Use Edison to Test/Add OpenMP Improve Scalability. Help from NERSC/Cray COE Available.
Create micro-kernels or examples to examine thread level performance, vectorization, cache use, locality.
Utilize performant / portable libraries
The Dungeon: Simulate kernels on KNL. Plan use of on package memory, vector instructions.


The Ant Farm Flow Chart

Use IPM and Darshan to Measure and Remove Communication and IO Bottlenecks from Code

Make Algorithm Changes

Run Example in "Half Packed" Mode

Run Example at "Half Clock" Speed

Make Algorithm Changes

Is Performance affected by Half-Packing?

No

Is Performance affected by Half-Clock Speed?

No

Yes

Yes

Your Code is at least Partially Memory Bandwidth Bound

You are at least Partially CPU Bound

Likely Partially Memory Latency Bound (assuming not IO or Communication Bound)

...

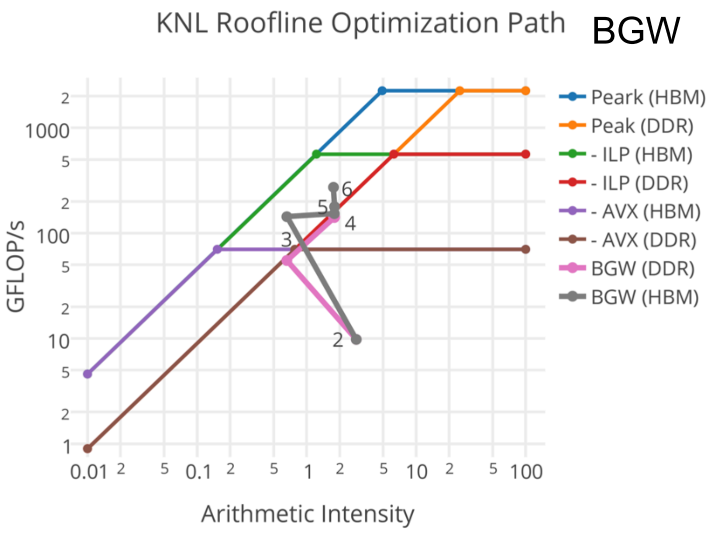
Using the Berkeley Lab Roofline Model to Frame Conversation With Users.

Interaction with Intel staff at dungeon sessions have lead to:

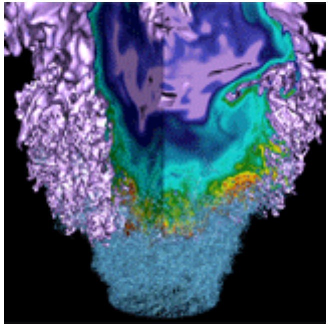
- Optimization strategy around roofline model (Tutorial presented at IXPUG).
- Advancement of Intel tools (SDE, VTune, Advisor).

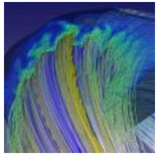
We are actively working with Intel on “co-design” of performance tools (Intel Advisor)

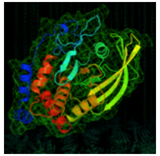
http://www.nersc.gov/users/application-performance/measuring-arithmetic-intensity/

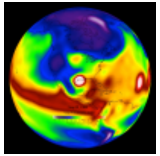


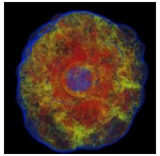
Early KNL Single Node Results

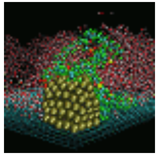






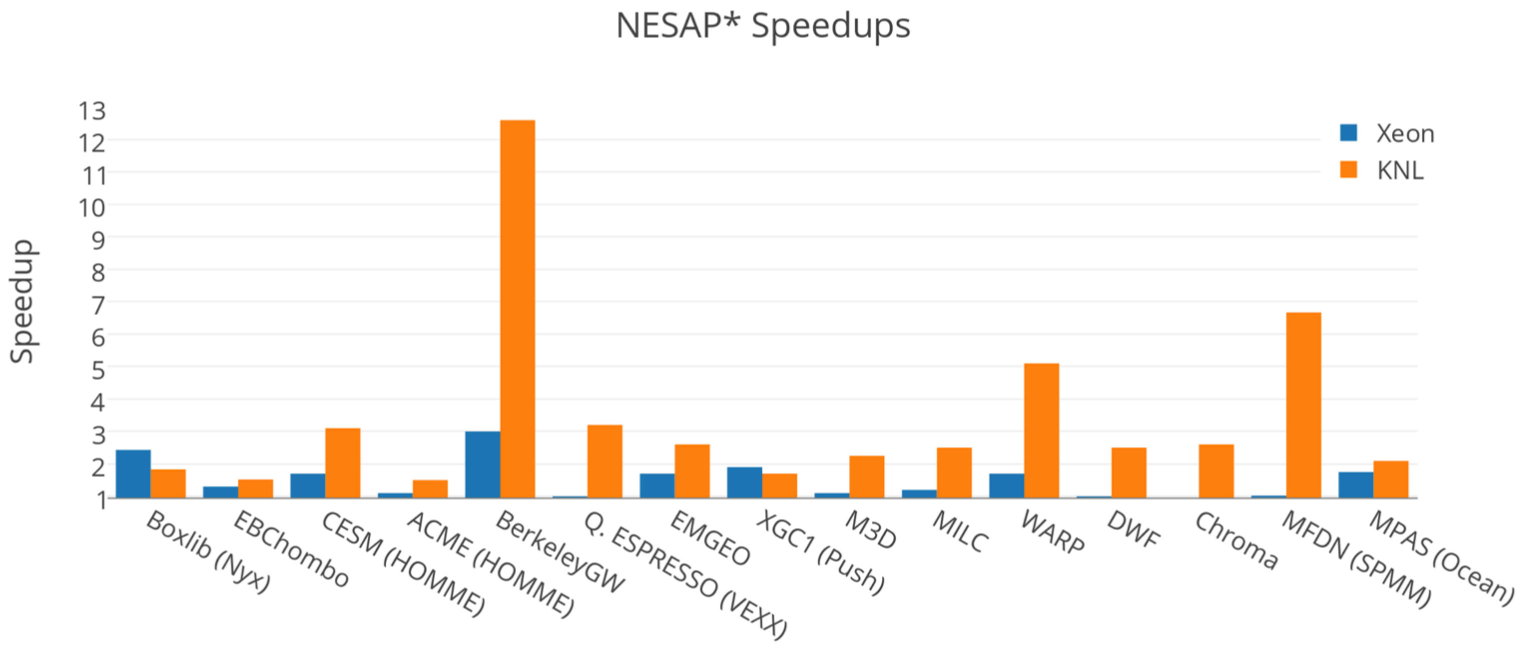




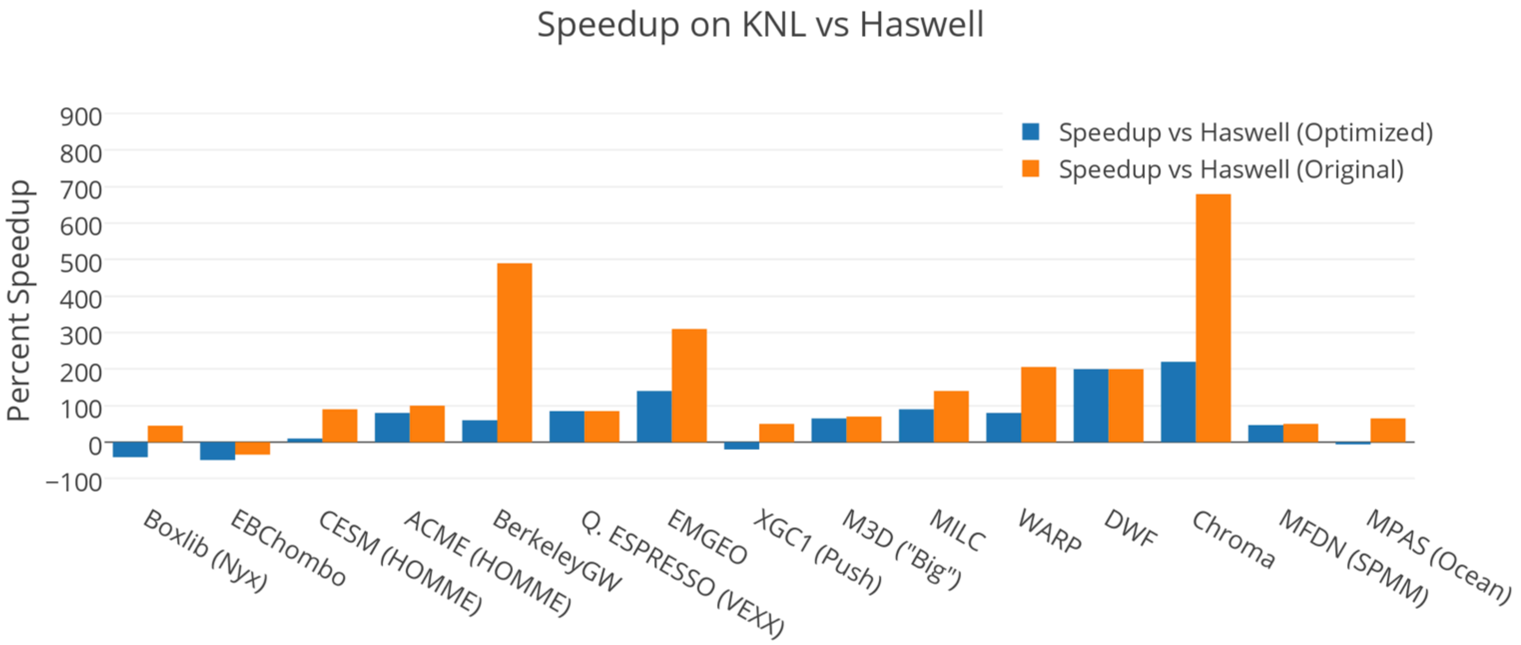


NERSC

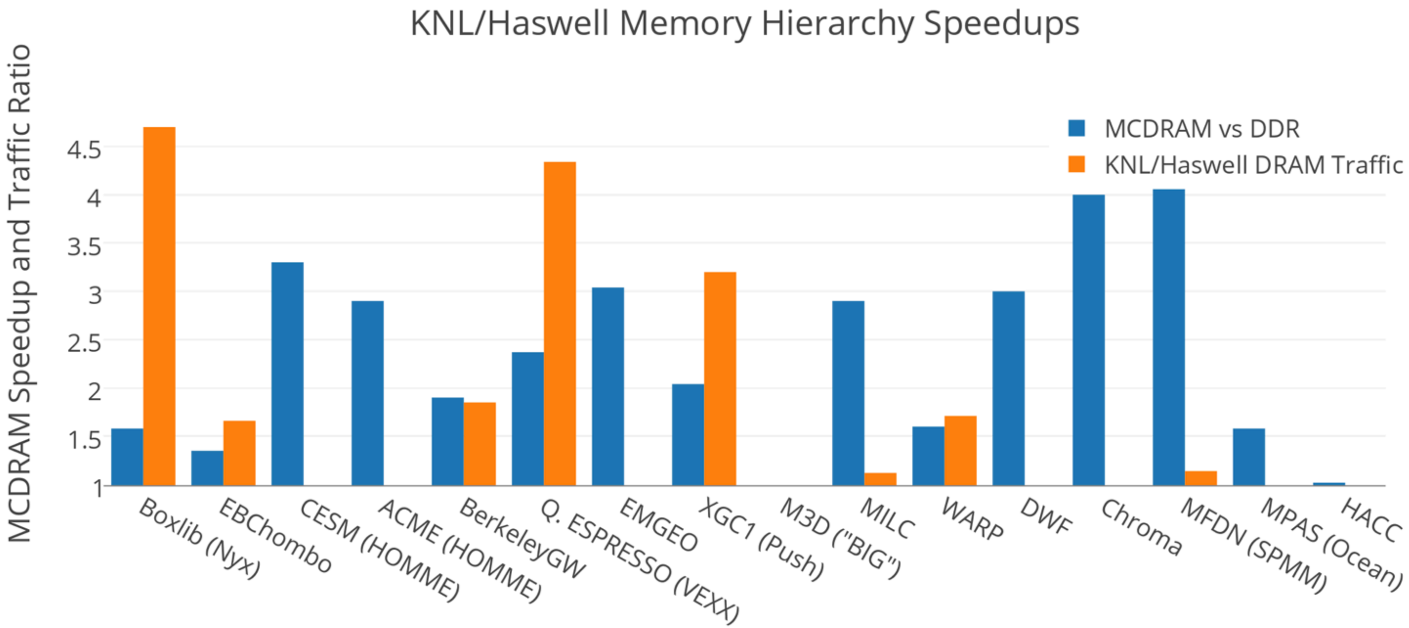
NESAP Speedups



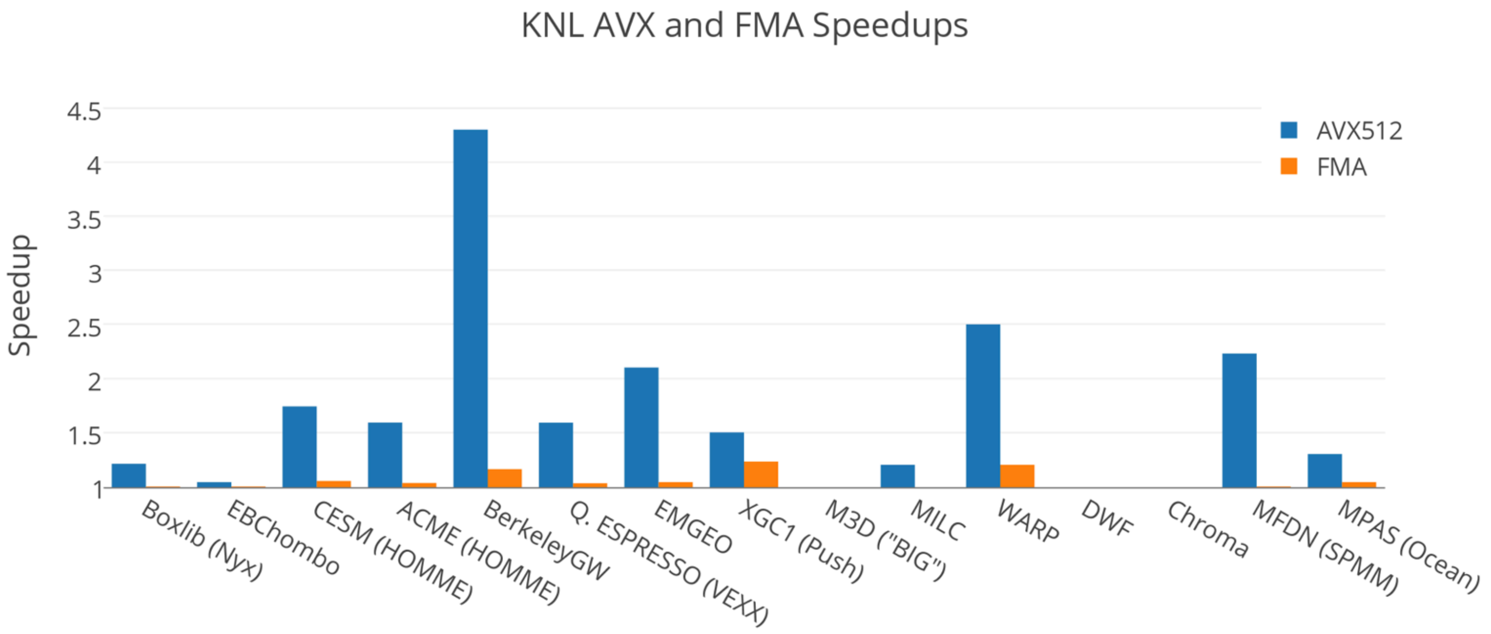
NESAP KNL vs Haswell



NESAP MCDRAM Effects



NESAP VPU Effects



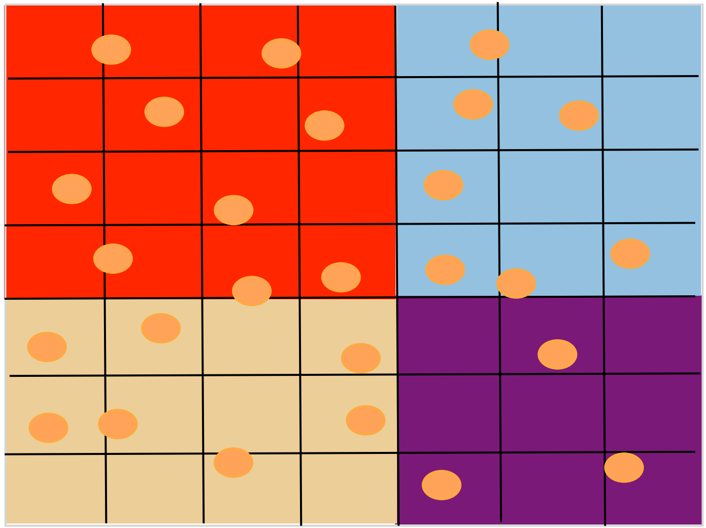
WARP Example

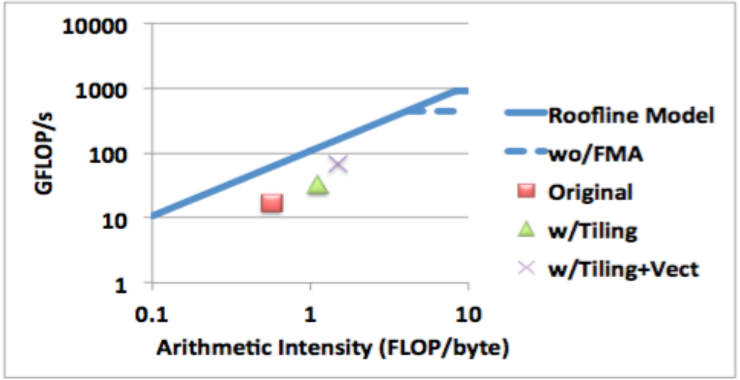
PIC Code, Current Deposition and Field Gathering

dominate cycles

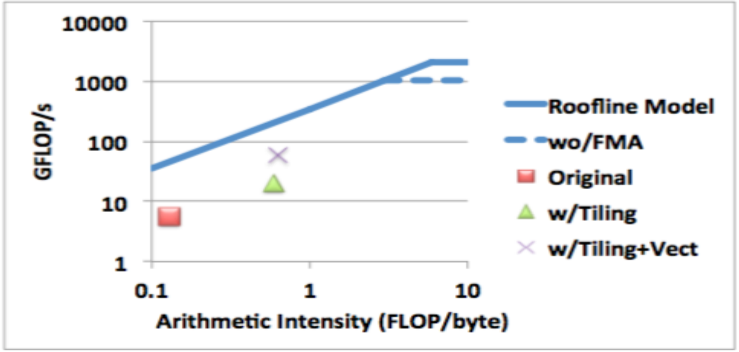
Tiling Added in Pre-Dungeon Work

Vectorization added in dungeon work





(a) Haswell Roofline



(b) KNL Roofline

MFDN Example

Use case requires all memory on node (HBM + DDR)

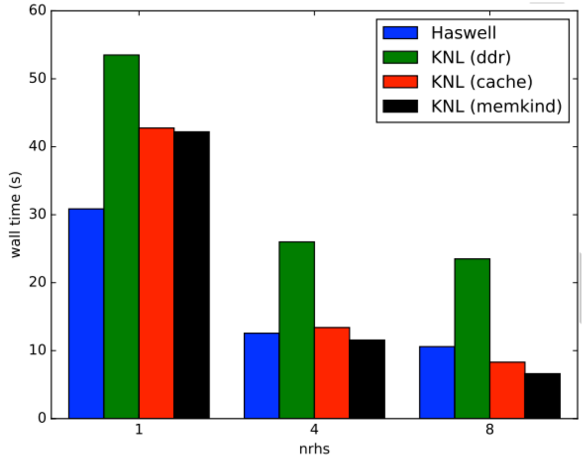
Two phases:

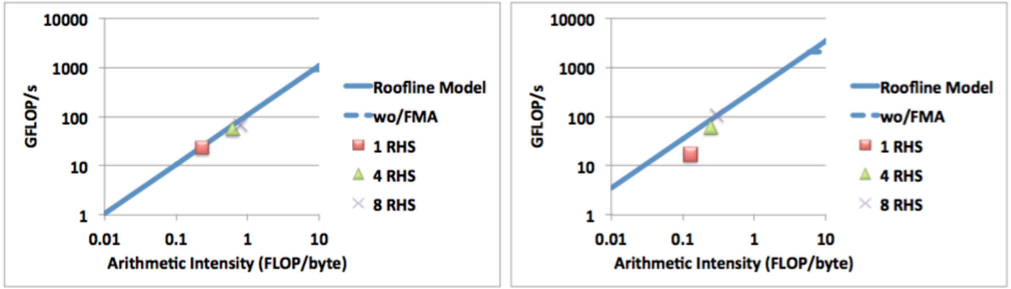
1. Sparse Matrix-Matrix or Matrix-Vector multiplies
2. Matrix Construction (Not Many FLOPs).

Major breakthrough at Dungeon Session with Intel. Code sped up by > 2x in 3 days.

Working closely with Intel and Cray staff on vector version of construction phase. For example, vector popcount.

SPMM Performance on Haswell and KNL





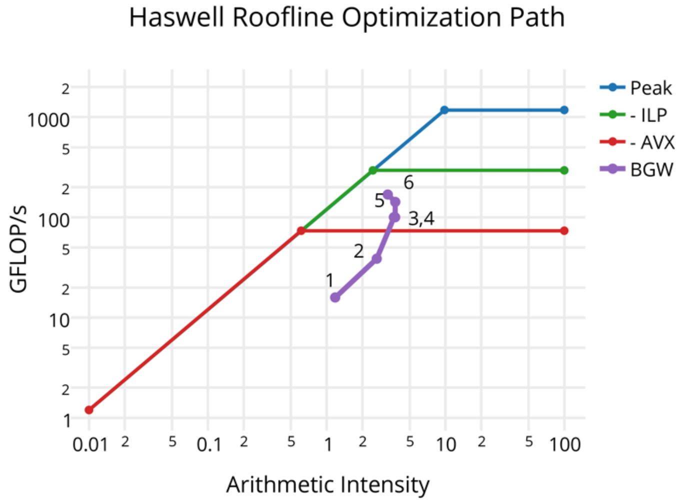
(a) Haswell Roofline

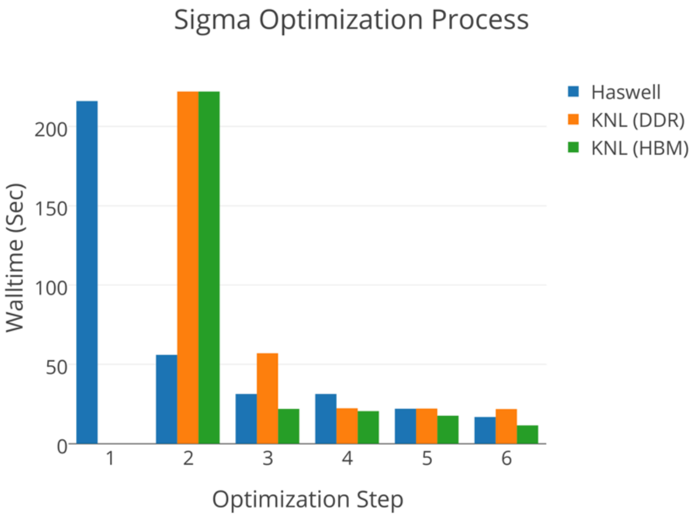
(b) KNL Roofline

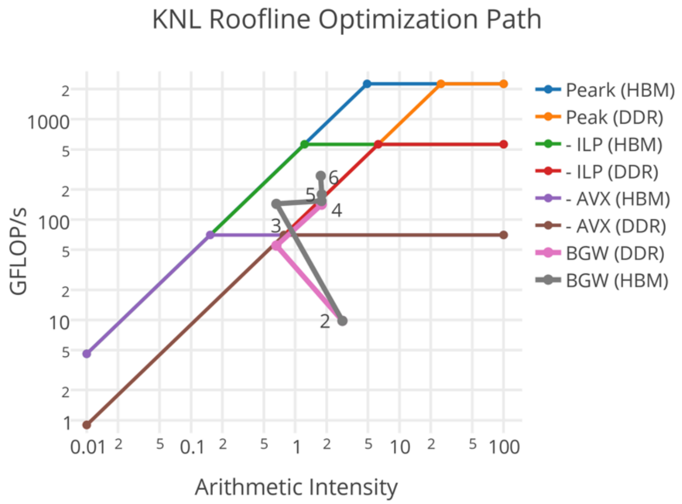
BerkeleyGW Optimization Example

Optimization process for Kernel-C (Sigma code):

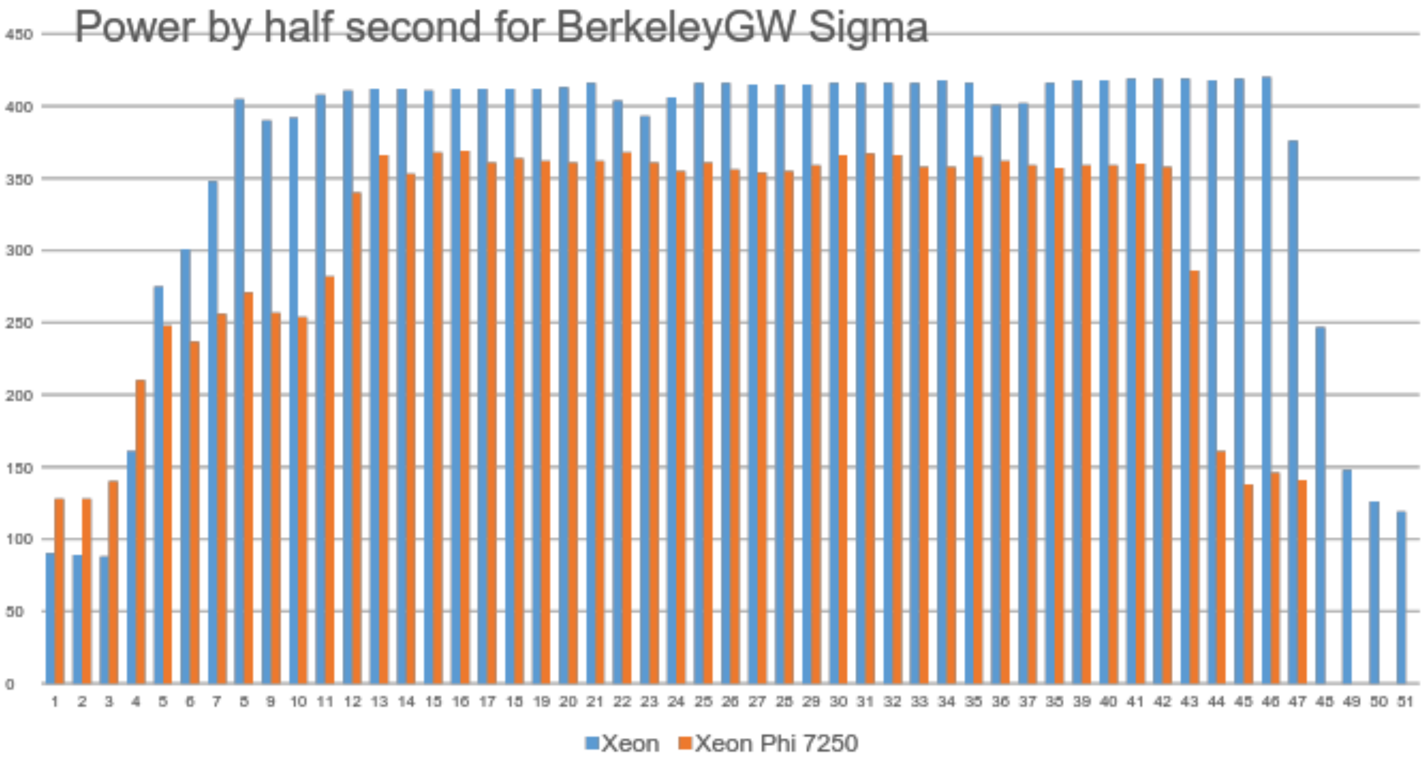
1. Refactor (3 Loops for MPI, OpenMP, Vectors)
2. Add OpenMP
3. Initial Vectorization (loop reordering, conditional removal)
4. Cache-Blocking
5. Improved Vectorization
6. Hyper-threading


Haswell Roofline Optimization Path

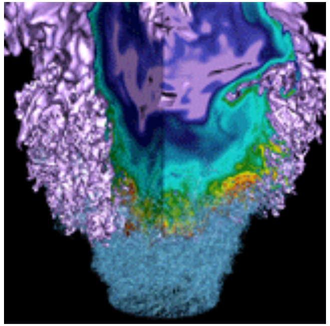

Sigma Optimization Process

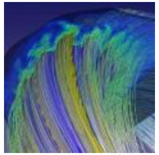

KNL Roofline Optimization Path

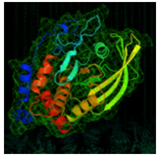
BerkeleyGW Example

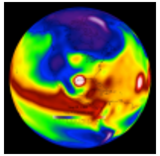


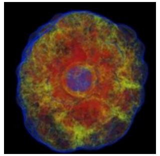
Early Cori at Scale Studies

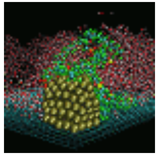












What are the issues at scale?

- Cori is based on the Cray XC architecture and uses the Aries high-speed interconnect
- KNL performance per core is ~⅓ that of a Haswell/Broadwell Xeon core
- Much of the communication stack, and in particular MPI, is done on the core
  - Cray MPI does support asynchronous progress with core specialization threads
  - Aries Block Transfer Engine also enables progress
  - Aries does have a Collective Engine to offload MPI collectives, including Barrier, small message Bcast, Allreduce, etc.
- Do we have to repartition our MPI+OpenMP codes differently when running on KNL vs Haswell?

What we are NOT going to present today

- Cori priorities are bringing Haswell to production and stabilizing the hardware after integration of HSW & KNL
- KNL NUMA cluster mode tradeoffs
  - E.g. Quad vs SNC2 vs SNC4
- KNL memory mode tradeoffs
  - E.g. MCDRAM vs Cache
- Highly optimized results
  - These are initial tests and observations

MPI Micro-benchmarks

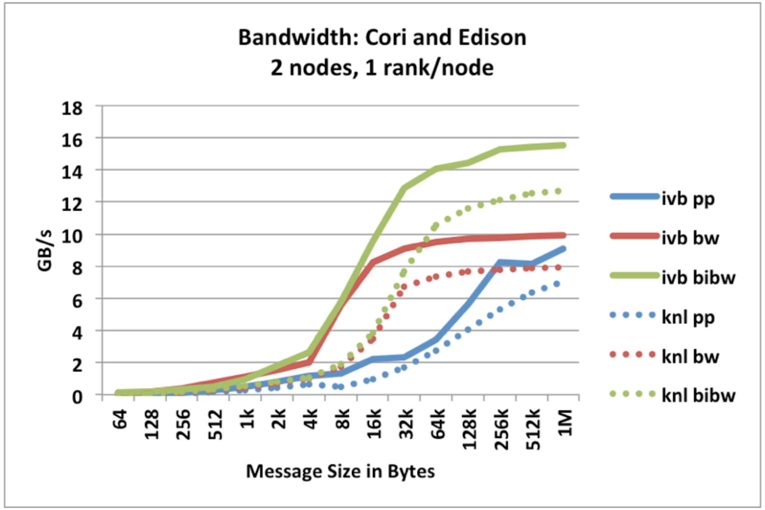
Edison (Ivy Bridge) and Cori (KNL)
2 Nodes, 1 core per node
Bandwidth
- Point-to-point (pp)
- Streaming (bw)
- Bi-directional streaming (bibw)

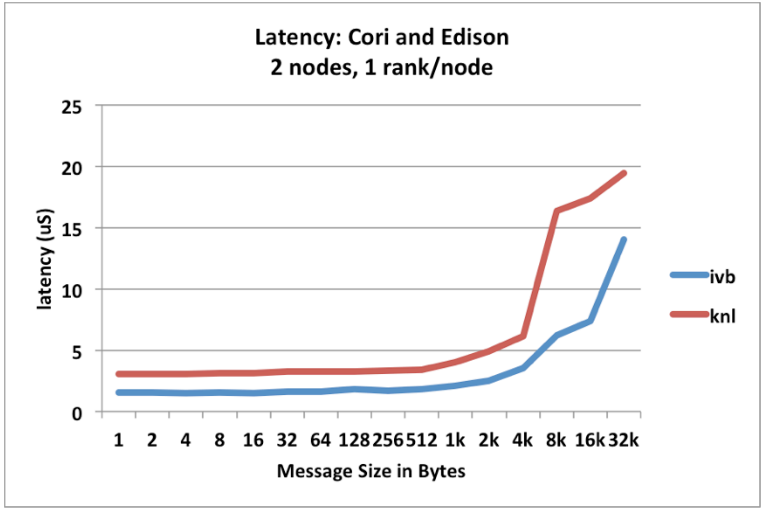
Latency
- Point-to-Point

Data collected in quad, flat mode

KNL single core:
Bandwidth is ~0.8x that of Ivy Bridge
Latency is ~2x higher

Using OSU MPI benchmark suite: http://mvapich.cse.ohio-state.edu/benchmarks/





But multi-core, high message rate traits are more interesting

In order to support many-core processors such as KNL, the NIC needs to support high message rates at all message sizes

Message rate benchmark at 64 nodes

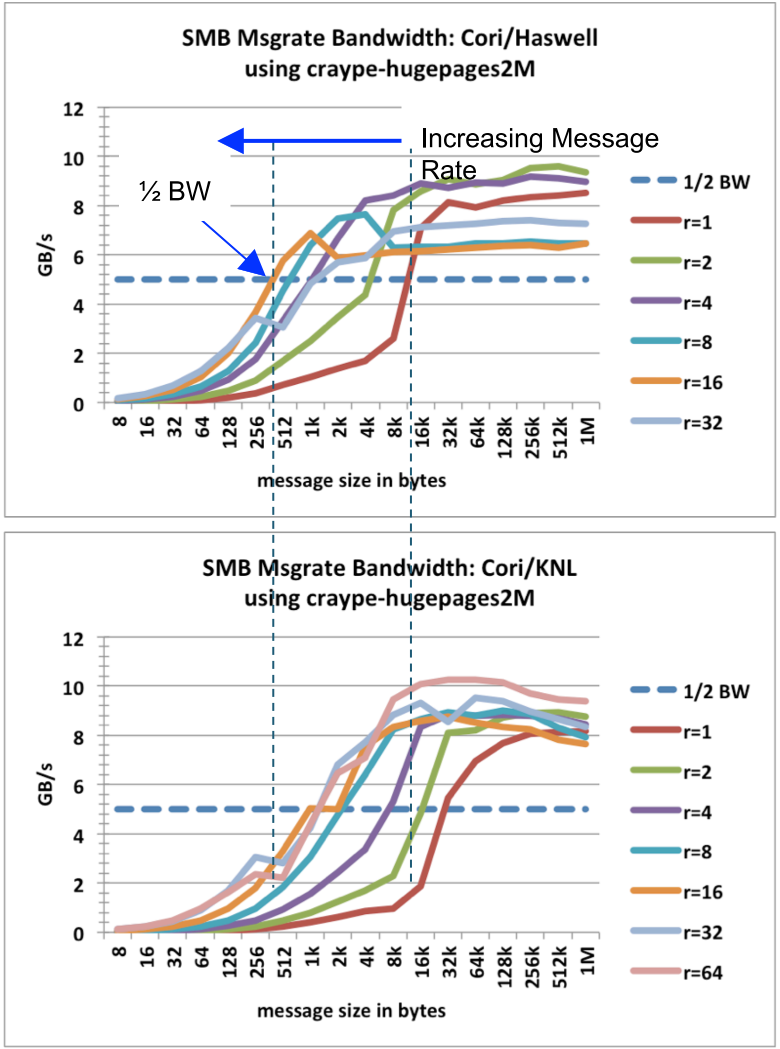
- Plotting BW, more intuitive than rate
- 3D stencil communication
- 6 Peers per rank
- ranks per node varied from 1 to 64
- Consulting with Cray, need to use hugepages to avoid Aries TLB thrashing

Data collected in quad, cache mode

KNL vs Haswell:

- Haswell reaches ½ BW with smaller msg sizes
- Still optimizing tuning to resolve lower HSW BW at high rank counts

Using Sandia MPI benchmark suite: http://www.cs.sandia.gov/smb/



How does all this translate to application performance?

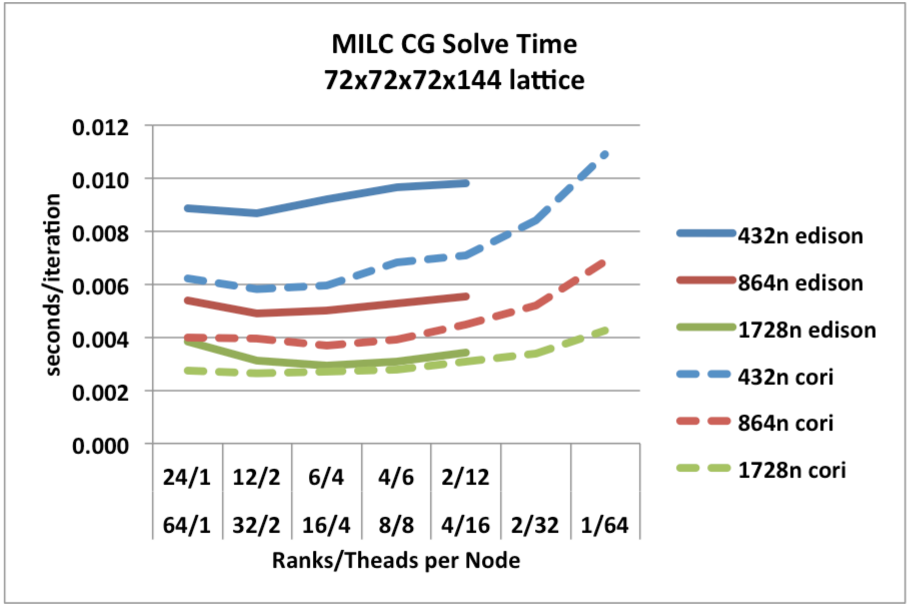
MILC - Quantum Chromodyamics (QCD) code
- #3 most used code at NERSC
- NESAP Tier 1 code
- Stresses computation and communication
- Weak and strong scaling needs

MPI vs OpenMP tradeoff study
- 3 scales, 432, 864 and 1728 nodes
- Fixed problem, 72^3x144 lattice

Cori Speedup vs Edison
- 1.5x at 432 nodes
- 1.3x at 864 nodes
- 1.1x at 1728 nodes

Data collected in quad, cache mode

Cori shows performance improvements at all scales and all decompositions



UPC Micro-benchmarks

Edison (Ivy Bridge) and Cori (KNL)
UPC uni-directional “put” Bw
“Get” characteristics are similar
Data collected in quad, flat mode

Single core:

Cori latency ~2x of Edison

Cori bandwidth ~0.8x of Edison

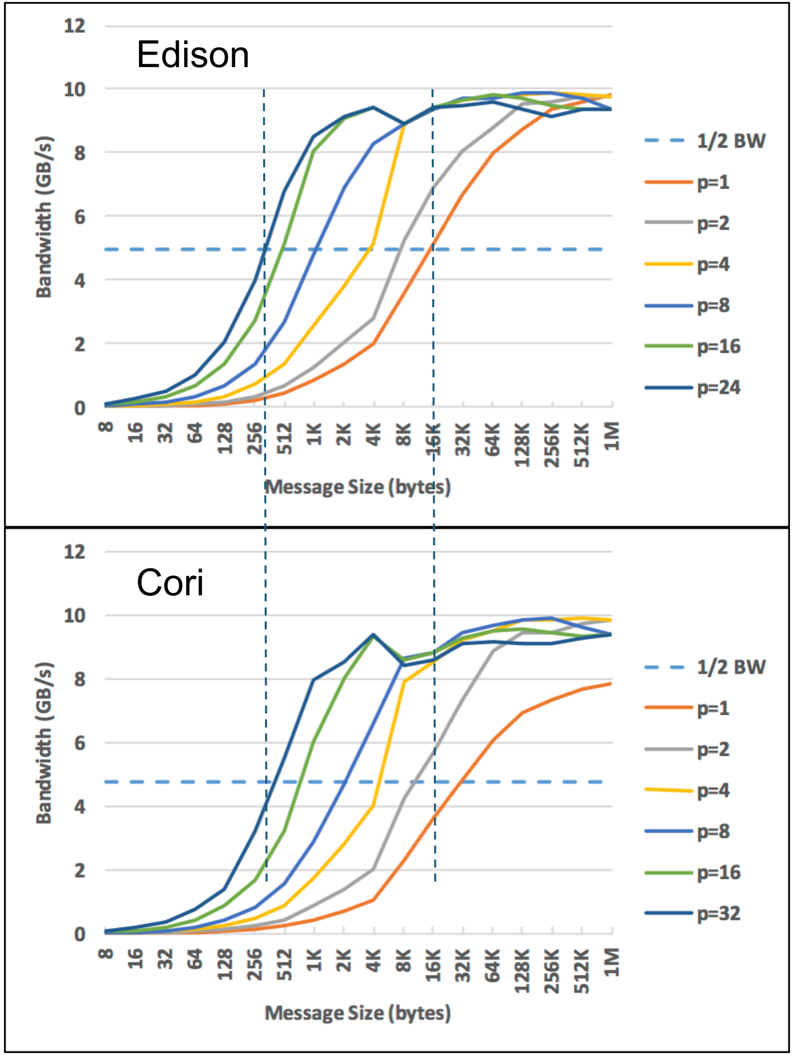
Same as MPI results

Multi core:

Similar BW profile at full core counts

Peak message rate acheived at same message size (256 bytes)

Using OSU MPI benchmark suite: http://mvapich.cse.ohio-state.edu/benchmarks/



Hipmer Micro-benchmarks

- Hipmer is a De Novo Genome Assembly Code based on UPC
- Micro-benchmarks emulate key communication operators
  - Random Get
  - Traversal
  - Construction
- Data collected in quad, cache mode

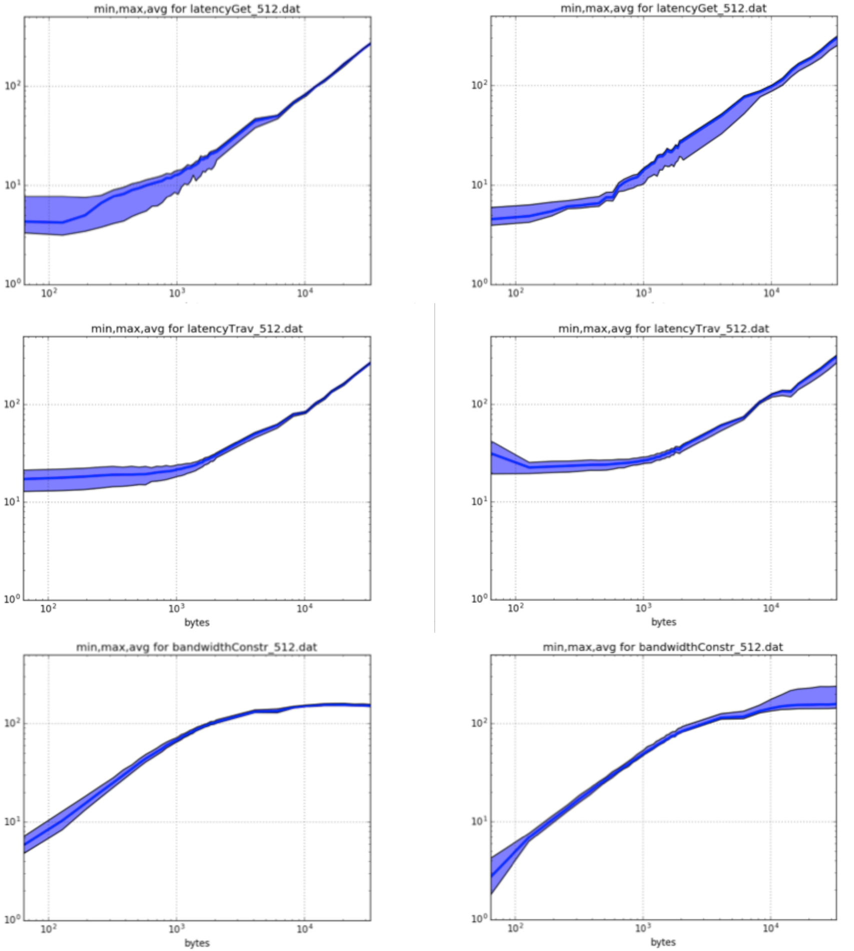
16 Nodes, 32 threads/node

- Higher avg. small message latency on KNL
- Similar avg. large message bandwidth

Random Get Latency vs Message Size

Traversal Latency vs Message Size

Construction Bandwidth vs Message Size



Cori/Haswell

Cori/KNL

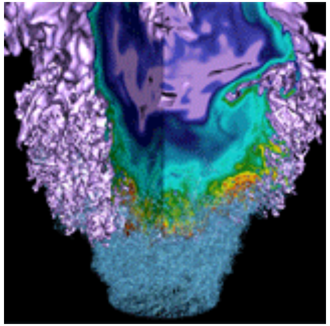
Parting observations

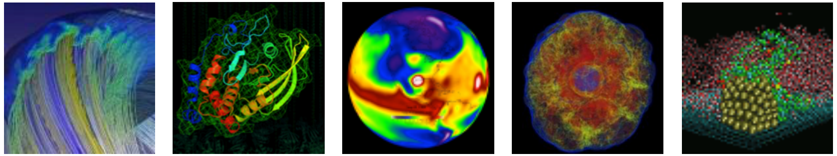
- Although we are in the early stages of bringing up Cori's KNL partition, it looks like codes will scale effectively to full size of the machine

- Individual core performance is lower, as expected, but aggregate performance is on par or better than Edison and Cori/Haswell

- Although we have not extensively compared the available clustering modes, we expect some combination of Quad/Flat and Quad/Cache to be used in early production

END, Thank you!





NERSC

U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB