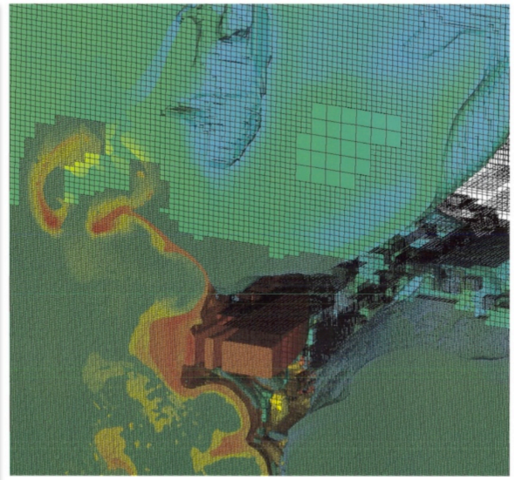
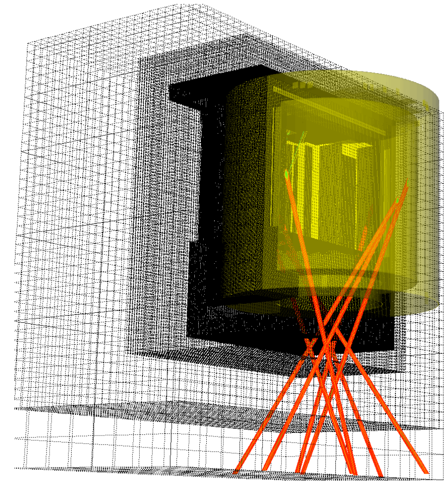
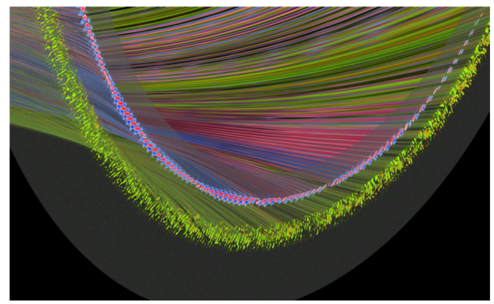
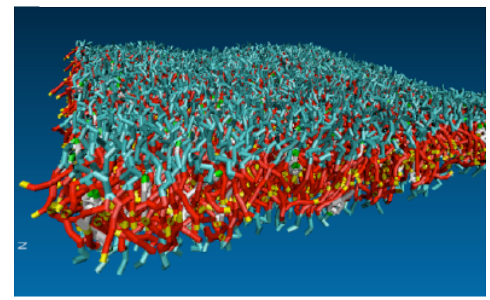
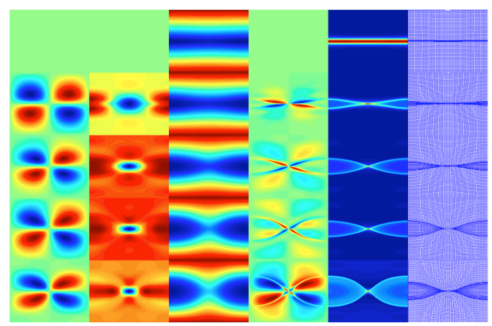
Research on the Path to Exascale at NERSC

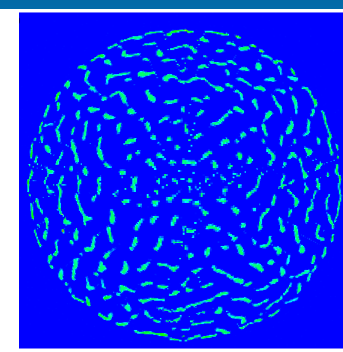












Alice E. Koniges

Sustainable Research Pathways Workshop

December 8, 2015

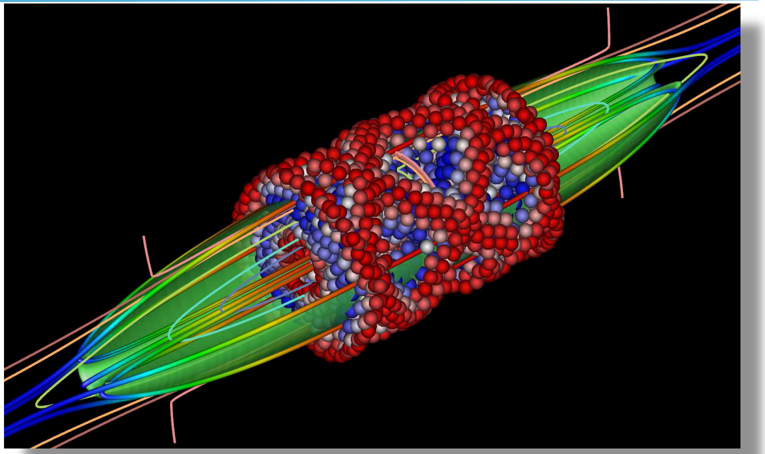
U.S. DEPARTMENT OF ENERGY | Office of Science

NERSC

BERKELEY LAB
Lawrence Berkeley National Laboratory

NERSC

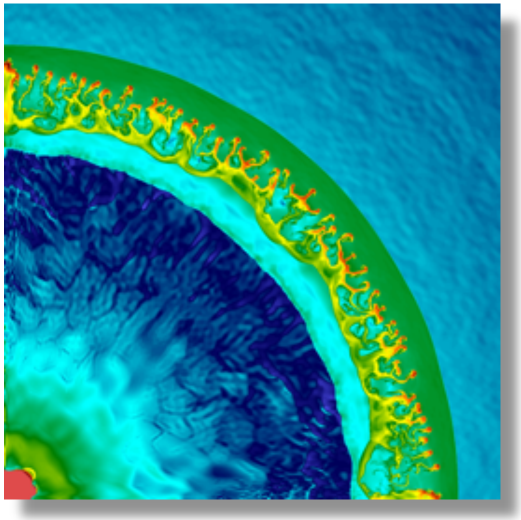
- **National Energy Research Scientific Computing Center**
  - Established 1974, first unclassified supercomputer center
  - Original mission: to enable computational science as a complement to magnetically controlled plasma experiment
- Today's mission: ***Accelerate scientific discovery at the DOE Office of Science through high performance computing and extreme data analysis***
- A national user facility



Trajectory of an energetic ion in a Field Reverse Configuration (FRC) magnetic field. Magnetic separatrix denoted by green surface. Spheres are colored by azimuthal velocity. Image courtesy of Charlson Kim, U. of Washington; NERSC repos m487, mp21, m1552

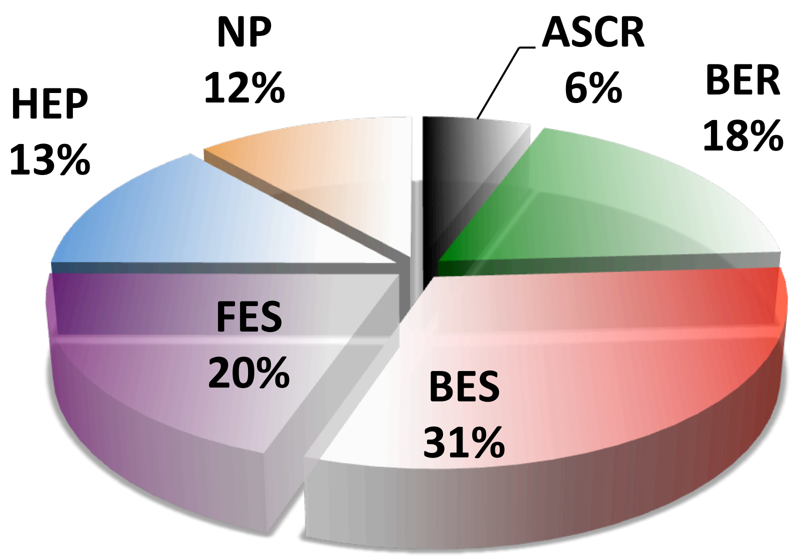
NERSC: Mission Science Computing for the DOE Office of Science

- **Diverse workload:**
  - 4,500 users, 700+ projects
  - 700 codes; 100s of users daily
- **Allocations controlled primarily by DOE**
  - 80% DOE Annual Production awards (ERCAP):
    - From 10K hour to ~10M hour
    - Proposal-based; DOE chooses
  - 10% DOE ASCR Leadership Computing Challenge
  - 10% NERSC reserve
    - NISE, NESAP



Collision between two shells of matter ejected in two supernova eruptions, showing a slice through a corner of the event. Colors represent gas density (red is highest, dark blue is lowest). Image courtesy of Ke-Jung Chen, School of Physics and Astronomy, Univ. Minnesota. Repo m1400

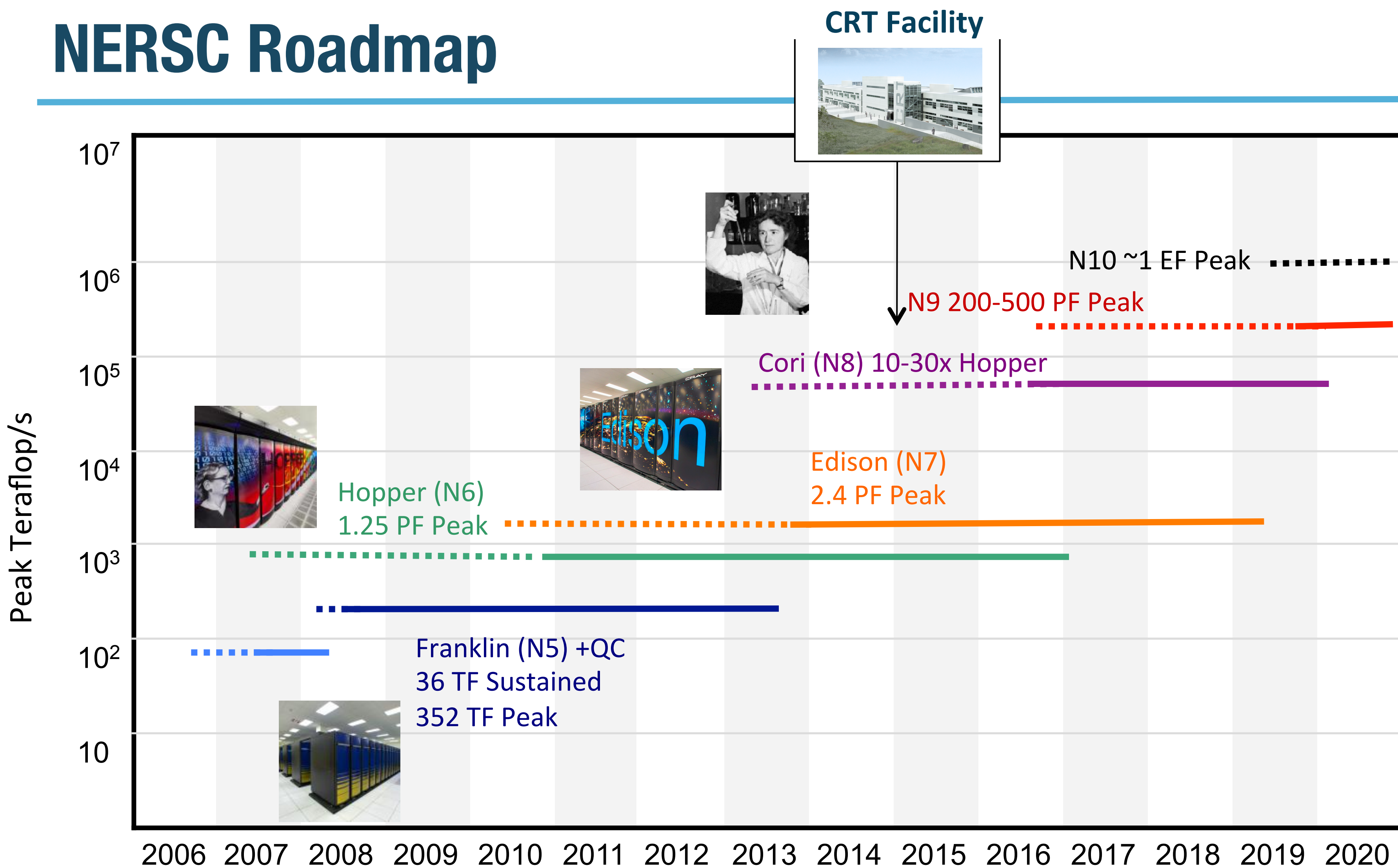
DOE View of Workload



NERSC Allocations By DOE Office

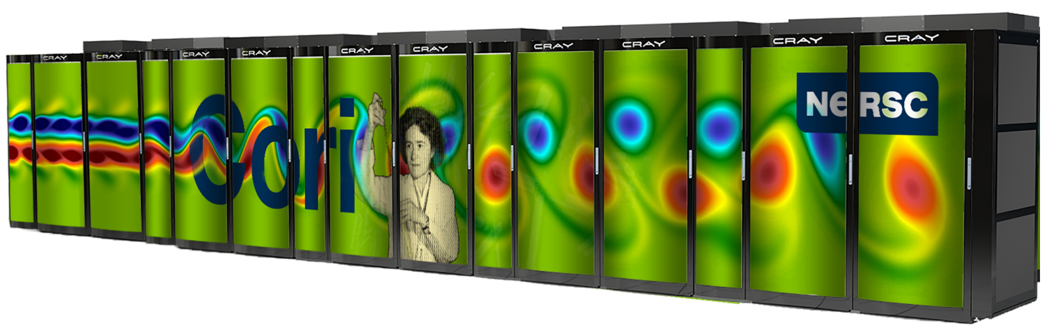
ASCR	Advanced Scientific Computing Research
BER	Biological & Environmental Research
BES	Basic Energy Sciences
FES	Fusion Energy Sciences
HEP	High Energy Physics
NP	Nuclear Physics

NERSC Roadmap



The Big Picture: KNL is Coming to NERSC

- **The next large NERSC production system "Cori" will be Intel Xeon Phi KNL (Knights Landing) architecture**
  - Energy efficient manycore system
  - > 9300 single socket nodes, multiple NUMA domains
  - Self-hosted, not an accelerator
  - 72 cores/node, 4 hardware threads per core. Total of 288 threads per node
  - AVX512, larger vector length of 512 bits (8 double-precision elements)
  - On package high-bandwidth memory (HBM)
  - Burst Buffer



Edison / Cori Quick Comparison

Edison (Ivy-Bridge)

NERSC Cray XC30 system

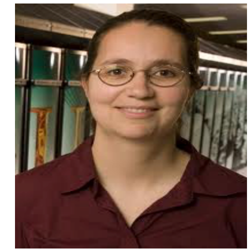
- 12 Cores Per CPU
- 24 Logical Cores Per CPU
- 2.4-3.2 GHz
- Vector length of 256 bits, 4 Double Precision Ops per Cycle (+ multiply/add)
- 2.5 GB of Memory Per Core
- ~100 GB/s Memory Bandwidth

Cori (Knights-Landing)

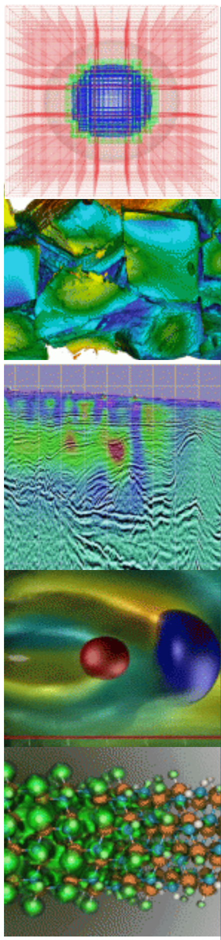
- 72 Physical Cores Per CPU
- 288 Logical Cores Per CPU
- Much slower GHz
- Vector length of 512 bits, 8 Double Precision Ops per Cycle (+ multiply/add)
- < 0.3 GB of Fast Memory Per Core
- < 2 GB of Slow Memory Per Core
- Fast memory has ~ 5x DDR4 bandwidth
- Burst Buffer for fast IO

NESAP: NERSC Exascale Science Application Program

Jack Deslippe



Rebecca Hartman-Baker



<u>Advanced Scientific Computing Research</u>

BoxLib AMR Framework

Chombo-crunch

<u>High Energy Physics</u>

WARP & IMPACT

MILC

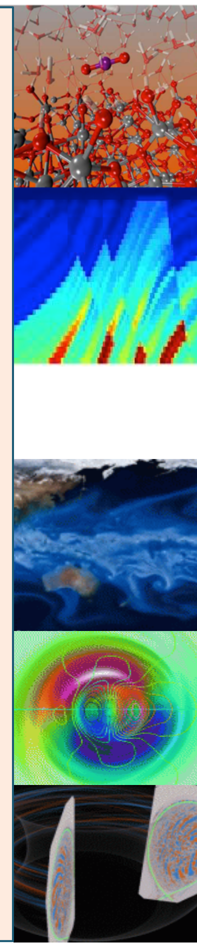
HACC

<u>Nuclear Physics</u>

MFDn

Chroma

DWF/HISQ



<u>Basic Energy Sciences</u>

Quantum Espresso

BerkeleyGW

PARSEC

NWChem

EMGeo

<u>Biological and Environmental Research</u>

Gromacs

Meraculous

MPAS-O

ACME

CESM

<u>Fusion Energy Sciences</u>

M3D

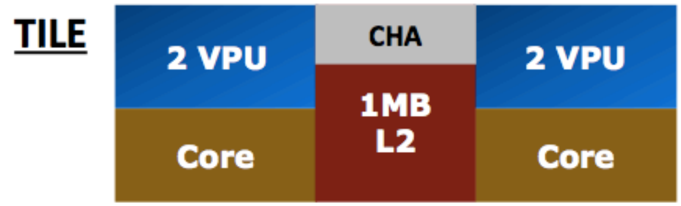
XGC1

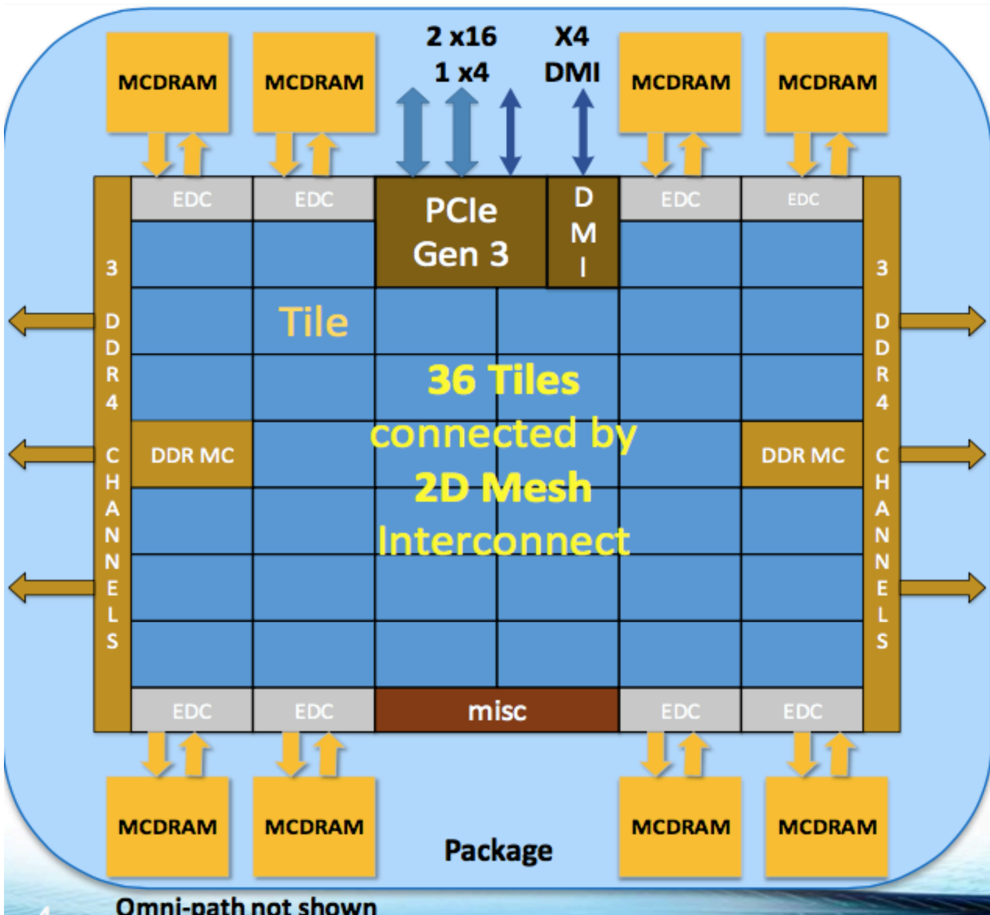
Programming Considerations: Running on Cori

- Application is very likely to run on KNL with simple porting, but high performance is harder to achieve.
- Applications need to explore more on-node parallelism with thread scaling and vectorization, also to utilize HBM and burst buffer options.
- Many applications will not fit into the memory of a KNL node using pure MPI across all HW cores and threads because of the memory overhead for each MPI task.
- Hybrid MPI/OpenMP is the recommended programming model, to achieve scaling capability and code portability.
- Current NERSC systems (Edison/Hopper and Babbage) can help prepare codes for Cori.

Lots of cores with in package memory

Knights Landing Overview





Chip: 36 Tiles interconnected by **2D Mesh**
Tile: 2 Cores + 2 VPU/core + 1 MB L2

Memory: MCDRAM: 16 GB on-package; High BW
DDR4: 6 channels @ 2400 up to 384GB
IO: 36 lanes PCIe Gen3. 4 lanes of DMI for chipset
Node: 1-Socket only
Fabric: Omni-Path on-package (not shown)

Vector Peak Perf: 3+TF DP and 6+TF SP Flops
Scalar Perf: ~3x over Knights Corner
Streams Triad (GB/s): MCDRAM : 400+; DDR: 90+

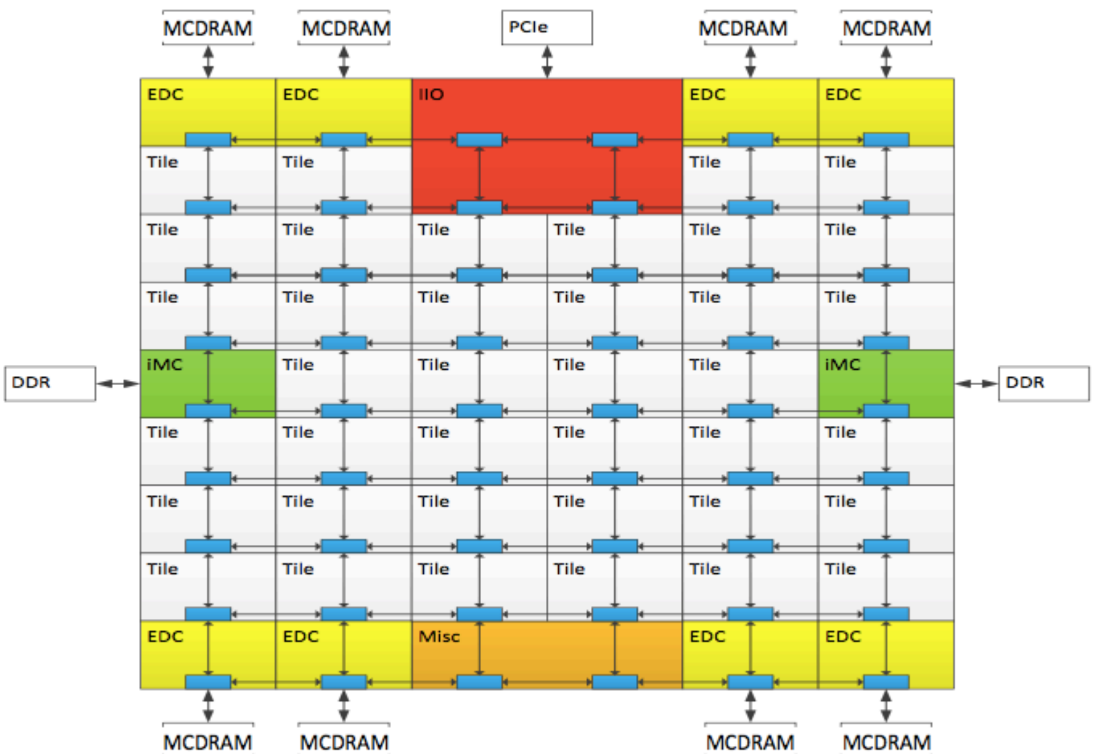
Source Intel: All products, computer systems, dates and figures specified are preliminary based on current expectations, and are subject to change without notice. KNL data are preliminary based on current expectations and are subject to change without notice. 1Binary Compatible with Intel Xeon processors using Haswell Instruction Set (except TSX). 2Bandwidth numbers are based on STREAM-like memory access pattern when MCDRAM used as flat memory. Results have been estimated based on internal Intel analysis and are provided for informational purposes only. Any difference in system hardware or software design or configuration may affect actual performance.

4

Source: Avinash Sodani, Hot Chips 2015 KNL talk

Connecting tiles

KNL Mesh Interconnect



Mesh of Rings

- Every row and column is a (half) ring
- YX routing: Go in Y → Turn → Go in X
- Messages arbitrate at injection and on turn

Cache Coherent Interconnect

- MESIF protocol (F = Forward)
- Distributed directory to filter snoops

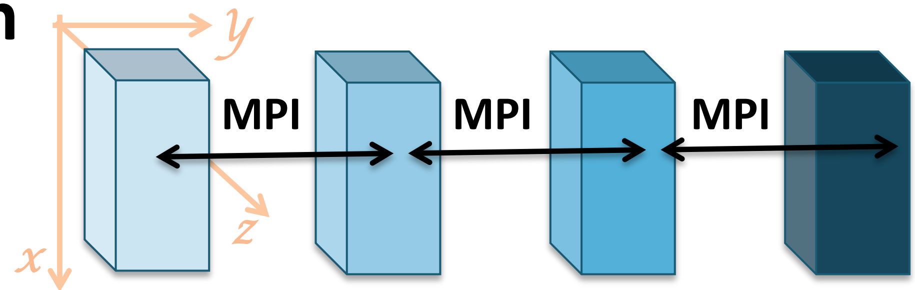
Three Cluster Modes

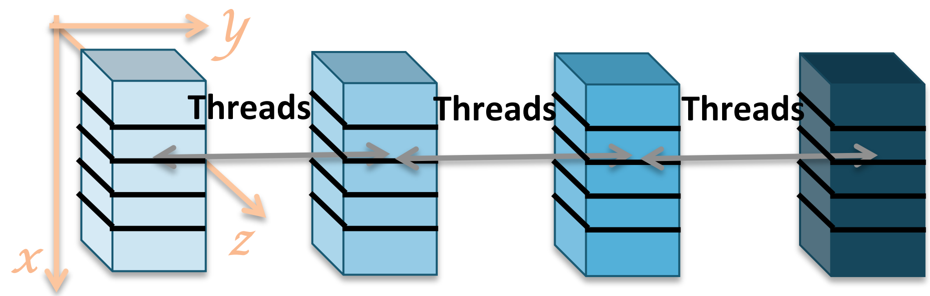
(1) All-to-All (2) Quadrant (3) Sub-NUMA Clustering

Source: Avinash Sodani, Hot Chips 2015 KNL talk

To run effectively on Cori users will have to:

- **Manage Domain Parallelism**
  - independent program units; explicit
- **Increase Node Parallelism**
  - independent execution units within the program; generally explicit
- **Exploit Data Parallelism**
  - Same operation on multiple elements
- **Improve data locality**
  - Cache blocking; Use on-package memory

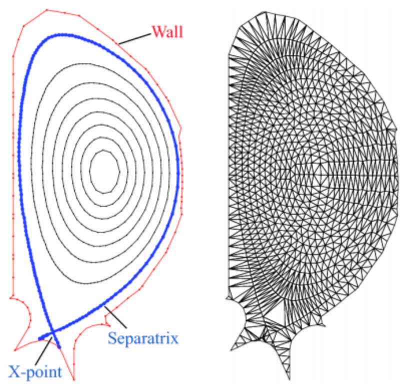




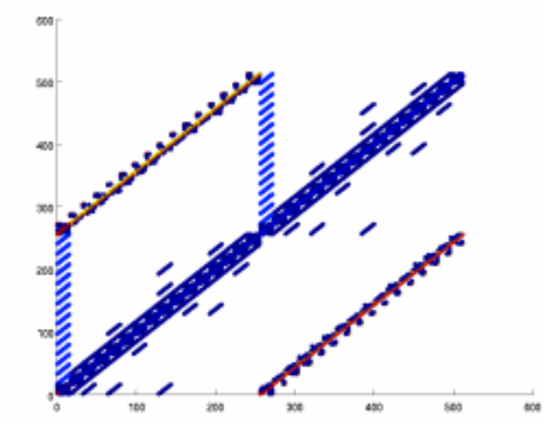

```
|--> DO I = 1, N
|        R(I) = B(I) + A(I)
|--> ENDDO
```

CASE STUDY: XGC1 PIC Fusion Code

- **Particle-in-cell code used to study turbulent transport in magnetic confinement fusion plasmas.**
- **Uses fixed unstructured grid. Hybrid MPI/OpenMP for both spatial grid and particle data. (plus PGI CUDA Fortran, OpenACC)**
- **Excellent overall MPI scalability**
- **Internal profiling timer borrowed from CESM**
- **Uses PETSc Poisson Solver** (separate NESAP effort)
- **60k+ lines of Fortran90 codes.**
- **For each time step:**
  - Deposit charges on grid
  - Solve elliptic equation to obtain electro-magnetic potential
  - Push particles to follow trajectories using forces computed from background potential (~50-70% of time)
  - Account for collision and boundary effects on velocity grid
- **Most time spent in Particle Push and Charge Deposition**



Unstructured triangular mesh grid due to complicated edge geometry



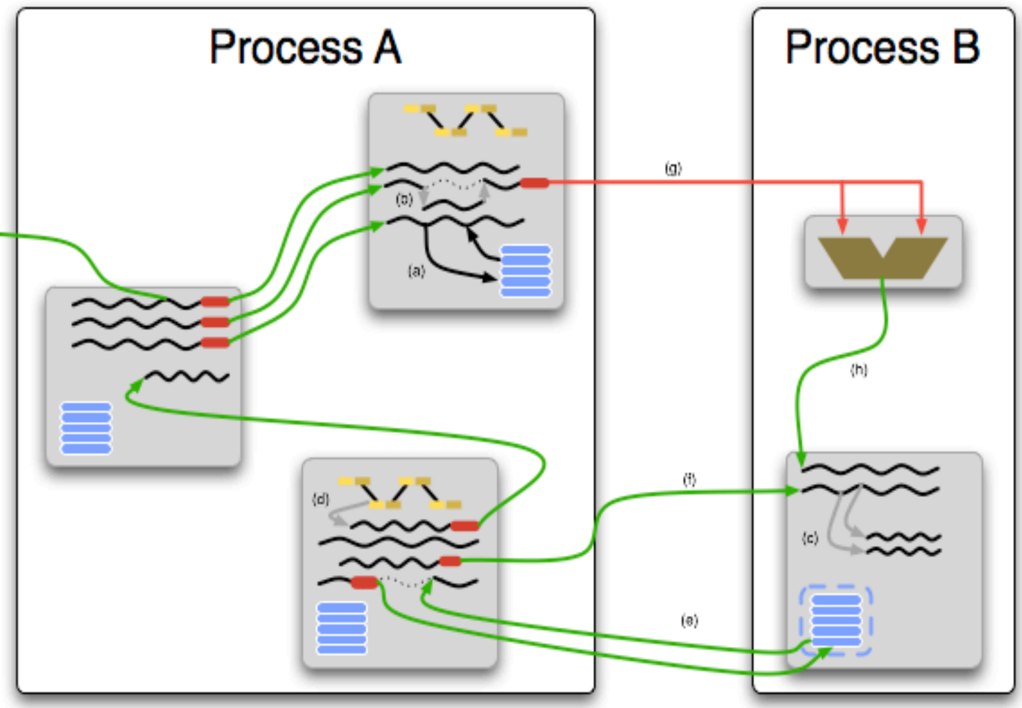
Sample Matrix of communication volume

Programming Portability

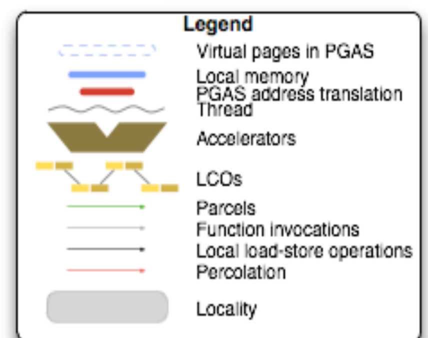
- Currently XGC1 runs on many platforms
- Part of NESAP and ORNL CAAR programs
- Applied for ANL Theta program
- Previously used PGI CUDA Fortran for accelerators
- Exploring OpenMP 4.0 target directives and OpenACC.
- Have #ifdef _OpenACC and #ifdef _OpenMP in code.
- Hope to have as fewer compiler dependent directives as possible.
- Nested OpenMP is used
- Needs thread safe PSPLIB and PETSc libraries.

New algorithms should work in concert with new exascale operating systems: ParalleX Execution Model

- **Lightweight multi-threading**
  - Divides work into smaller tasks
  - Increases concurrency
- **Message-driven computation**
  - Move work to data
  - Keeps work local, stops blocking
- **Constraint-based synchronization**
  - Declarative criteria for work
  - Event driven
  - Eliminates global barriers
- **Data-directed execution**
  - Merger of flow control & data structure
- **Shared name space**
  - Global address space
  - Simplifies random gathers



(a) Local data access
(b) Local thread invocation (co-routine)
(c) Local thread invocation (concurrent threads)
(d) LCO spawning a thread
(e) Remote atomic memory operation through parcels
(f) Remote thread invocation through parcels
(g) Percolation
(h) Thread creation as result of continuation action



HPX and Related Application Development

- Explore app development alternative to “traditional MPI+X”.
- Question: Can a qualitatively different approach (Parallex-based):
  - Exploit untapped and new parallelism?
  - Improve expressability?
  - Improve productivity?
  - Get us to Exascale and beyond?
- Broad sampling of app domains & algorithms:
  - Plasma physics, Many-body & particle-in-cell (PIC)
  - Nuclear engineering & finite volume/eigensolvers.
  - Shock physics & finite element/explicit time integration.
  - Computational mechanics & implicit sparse solvers.
- Full team effort involving app designers, XPRESS team, HPX and ParalleX developers, and compiler and tools developers

Application Perspective

- **Use of Futures:**
  - Exploit previously inaccessible, fine-grain dynamic parallelism.
  - Natural framework for expressing data-driven parallelism.
- **Better than MPI:**
  - Beyond functional mimic of MPI.
  - Really use Active Global Address Space (AGAS)
  - Take advantage of fine-grained parallelism using a generalized concept of threads
- **Overarching Application Team goal:**
  - Demonstrate that Parallex-based approaches work
  - Superior to MPI+X in one or more metric:
    - Performance: Extracting latent parallelism.
    - Portability: Performance obtained from system's underlying runtime.
    - Productivity: Easier to write, understand, maintain.

The Ideal HPX Model from an Application Perspective

- **Functionalize – figure out what is a quantum of work**
- **Determine data dependencies**
- **Create a data flow structure**
- **Feed into data task manager**
- **Applications**
  – Sweet spot between grain sizes for various task granularities.
  – Strong scaling improves as you added extra levels of refinement. Opposite of what you see with MPI, giving more usable work to the simulation
  – Lots new results at SC15

NERSC is a great place for research on applications and programming models

- Access to latest hardware
- Application experts in a variety of domains
- NESAP program with post-docs and staff
- Participates in OpenMP and MPI language standards
- Cross-DOE projects on next generation runtimes and programming models