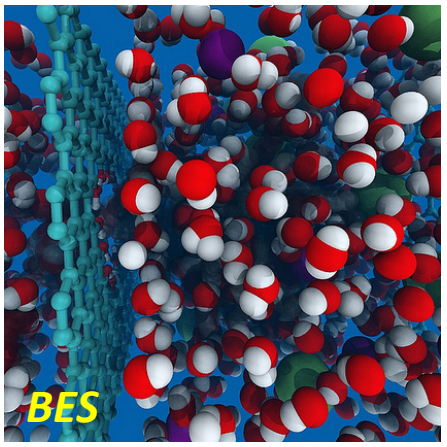
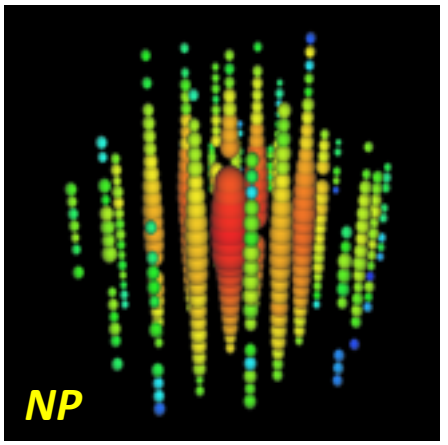
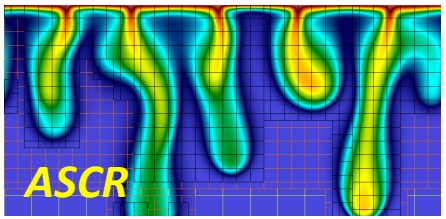
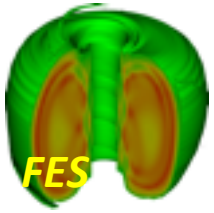
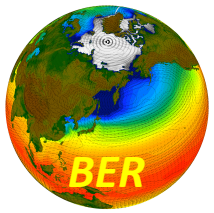
Transitioning to NERSC-8 and Beyond

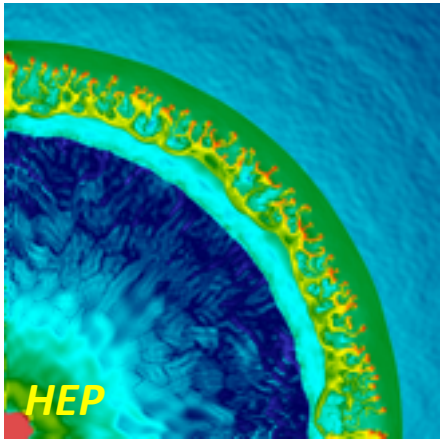












NERSC 40 YEARS at the FOREFRONT 1974-2014

Harvey Wasserman

NP Requirements Review
April 29, 2014

U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB
Lawrence Berkeley National Laboratory

What This Talk is About

- **NERSC will announce the NERSC-8 architecture soon.**
- **This talk concerns preparation for the system.**
- **Although we cannot disclose the architecture for NERSC-8, trend for all 2015-2016 systems is manycore**

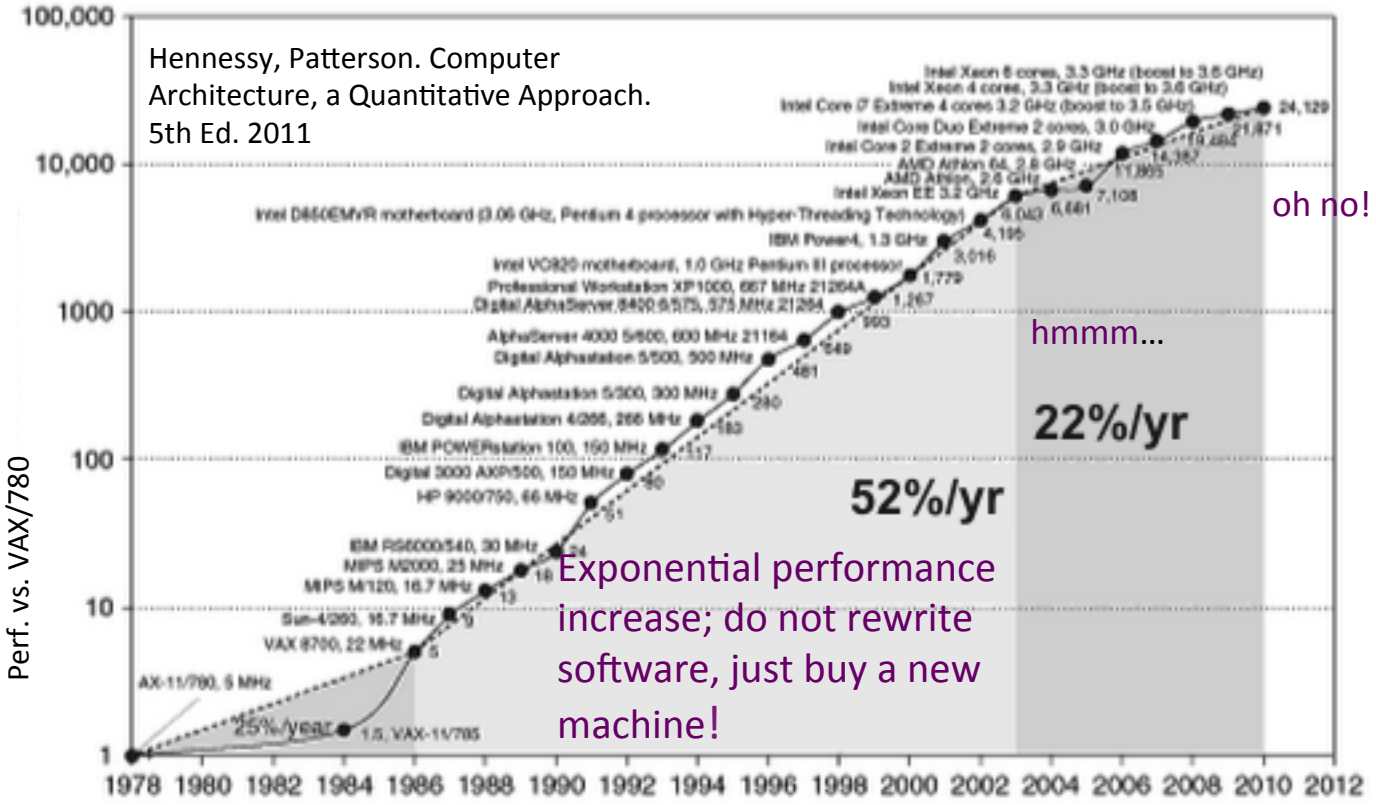
NERSC-8 Project Goals

- **NERSC directly supports DOE's science mission; we focus on the scientific impact of our users**
- **Need to provide a significant increase in computational capabilities for DOE SC computational research; at least 10x increase in sustained performance over NERSC-6 (Hopper)**
- **Begin transitioning user code base to energy-efficient manycore architectures and programming environments –**
  - only way to continue to provide compute speed improvements that meet user need; attempt to do this only once
- **Integrate the system into the NERSC environment, enabling user productivity**

What is "Manycore?"

- No precise definition
- Multicore (heavyweight): slow evolution from 2–12 cores per chip; core performance matters more than core count
- Manycore (lightweight): jump to 30–60+ cores per chip; core count essentially matters more than individual core performance
- Manycore: Relatively simplified, lower-frequency computational cores; less instruction-level parallelism (ILP); engineered for lower power consumption and heat dissipation

Why Now?



To sustain historic performance growth, the DOE community must prepare for new architectures

"Moore's Law" will continue.

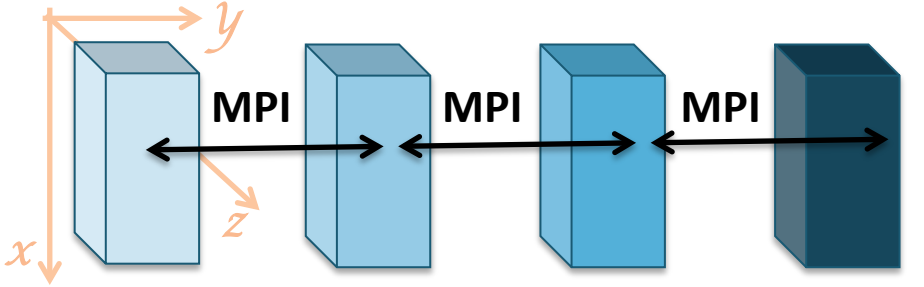
- Memory wall
- Power wall
- ILP wall

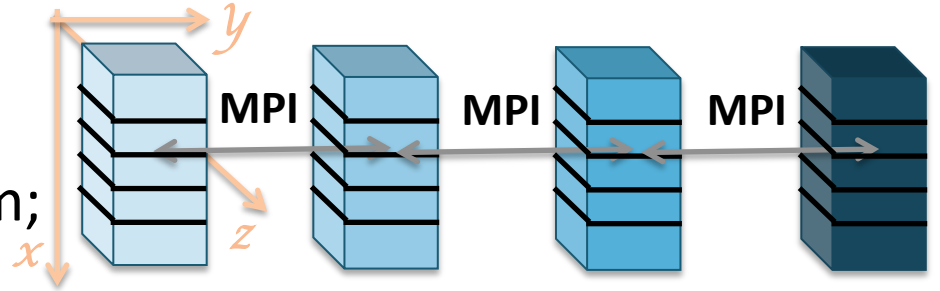
What Does it Mean for Users?

- **Look at sources of parallelism: new and old**

Sources of Parallelism

- **Domain Parallelism**
  - independent program units; explicit
- **Thread Parallelism**
  - independent execution units within the program; generally explicit
- **Data Parallelism**
  - Same operation on multiple elements
- **Instruction Level Parallelism**
  - Between independent instructions in sequential program





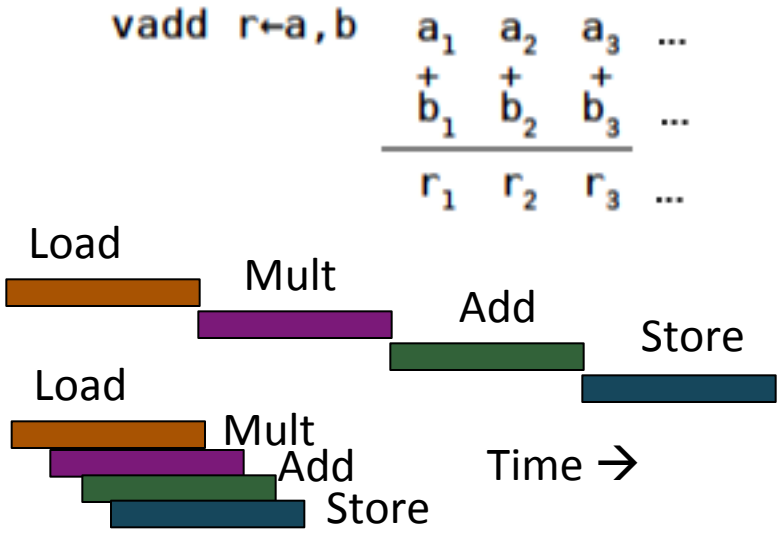

```
|--> DO I = 1, N
|        R(I) = B(I) + A(I)
|--> ENDDO
```

```
add r3 ← r1, r2
mul r0 ← r0, r1
sub r1 ← r3, r0
```
Parallel

What's All This About Vectorization (1/2)?

- **Vectorization is another form of on-node parallelism**

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



- Potentially more energy efficient because a single instruction will launch many operations on different data;
- Most commonly, multiple iterations of a loop can be done "concurrently." (Simpler control logic, too.)

What's All This About Vectorization (2/2)?

- **Vectorization analysis is performed by the compiler.**
- **Vector analysis problem: algorithms and/or programming styles can obscure or inhibit vectorization**

Loop exit (will not vectorize):

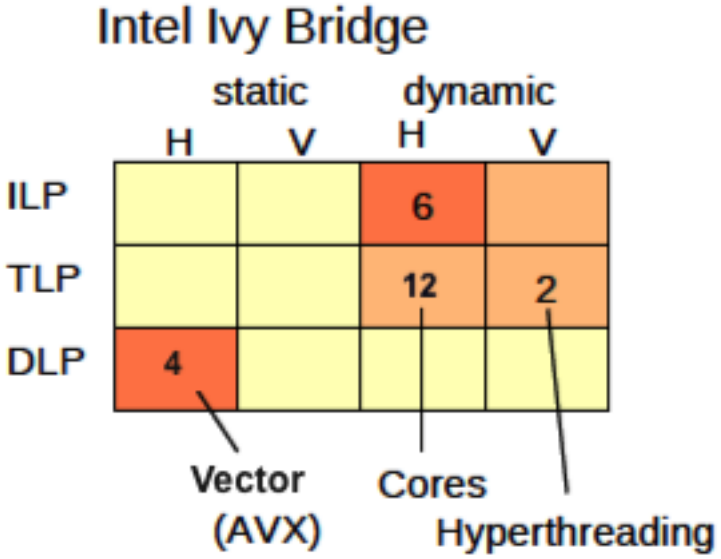
```
DO I = 1, N
     IF (A(I) < X) CYCLE
     ...
ENDDO
```

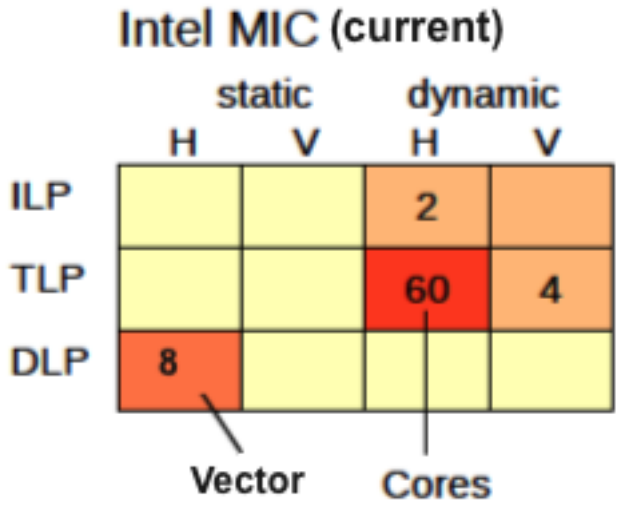
Loop dependency (will not vectorize):

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

- Application performance will be very sensitive to:
  - Percentage of code that can use the vector unit
  - Length of loops that vectorize

Multicore vs. Manycore Parallelism





- **ILP: Instruction Level Parallelism**
- **TLP: Thread Level Parallelism**
- **DLP: Data Level Parallelism**

What Does it Mean for Users?

- Codes will probably run on NERSC-8 without any changes.
- To take advantage of these emerging architectures, applications must
  - Exploit more parallelism
  - Express thread-level parallelism
  - Exploit data level parallelism
  - Manage data placement and movement
  - Accommodate less memory per process space

What is NERSC Doing About This?

- **NERSC-7 was not the best time to make the transition**

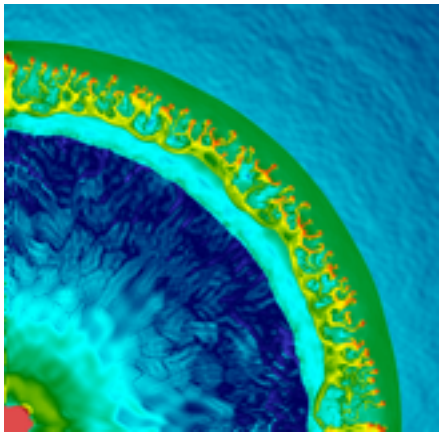
- **Transition of the workload to energy-efficient, manycore architectures is essential to success of NERSC-8.**

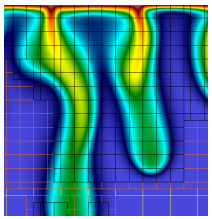
- **We have an "Application Readiness" effort since 2011**

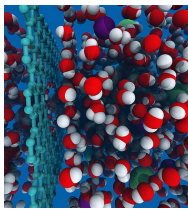
Who Else Cares About This?

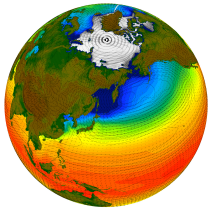
- **Transition of workloads to energy-efficient, manycore architectures is essential to success of exascale computing.**
- **In fact, these changes will affect computing at *all* scales. Energy efficiency is necessary for computer performance growth.**
- **A DOE-wide community is actively engaged. NERSC will build upon prior efforts, leverage lessons learned, and contribute.**
- **The effort is vital for the success of the computer industry.**

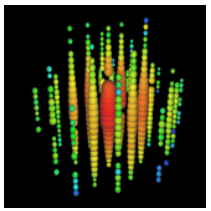
NERSC Application Readiness Effort











NERSC 40 YEARS at the FOREFRONT 1974-2014

U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB
Lawrence Berkeley National Laboratory

Programming Models Strategy

- **Necessary characteristics for broad adoption of a new programming model are**
  - Performance: At least 10x-50x performance improvement
  - Portability: Code performs well on multiple platforms; cannot be a proprietary solution
  - Durability: Solution must be good for a decade or more
- **Our near-term strategy is**
  - Smooth progression to exascale from user's point of view
  - Support for legacy code, albeit at less than optimal performance
  - Support for a variety of programming models
  - Support optimized libraries

App Readiness Approach

- Start early; profile workload; set user expectations
- Technology-agnostic approach to maintain investment going forward.
- Enable a significant fraction of NERSC workload to run on N8; NERSC-7 for codes that cannot.
- Transfer lessons learned to/from broader DOE community; Collaborate with ACES/ALCF/OLCF
- Leverage NERSC visibility to ensure vendor involvement
- Early test beds to learn about the issues now

Application Readiness Team

- Katie Antypas: FLASH (explicit hydro) (proxy: Castro, MAESTRO, S3D, AMR)
- Harvey Wasserman: SNAP (S_N Discrete Ordinates Transport)
- Jack Deslippe: BerkeleyGW (proxy: VASP, Abinit, Quantum Espresso)
- Woo-Sun Yang: Multigrid solver
- Matt Cordery: MPAS (Scalable Ocean Model) (proxy: CESM)
- Lenny Oliker: GTC (PIC – Fusion) (proxy: GTS, XGC, Osiris, g2s)

- Brian Austin: Zori (QMC) (proxy: qwalk)
- Kjiersten Fagnan: BLAST, Allpaths (Bioinformatics)
- Burlen Loring: Vis, Iso-surface
- Aaron Collier: MADAM-toast (CMB), Gyro (Fusion)
- Hongzhang Shan: NWChem (proxy: qchem, Gamess)
- Helen He: WRF (Regional Climate)
- Hari Krishnan: Vis, Iso-surface

- **Phase 1: NERSC/CRD team convened in 2011 to study thread parallelism and vectorization in key codes on local test beds.**
  - **Result: compendium of case studies reflecting porting effort, performance results, best practices, common issues on real codes**

Some App Readiness Initial General Findings

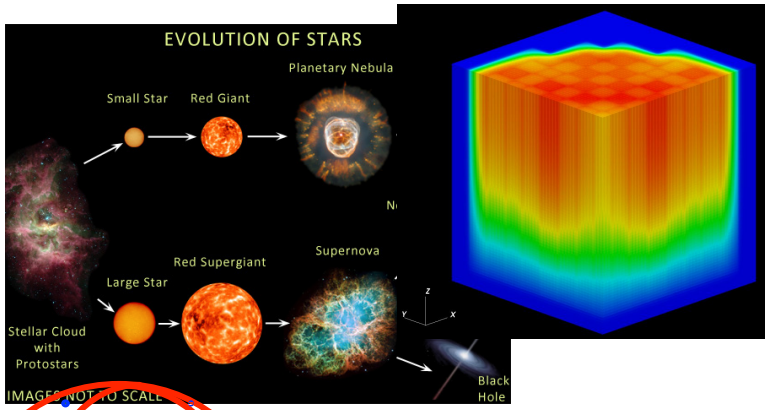
- Improving a code for advanced architectures can improve performance on traditional architectures.
- Inclusion of OpenMP may be needed just to get the code to run (or to fit within memory)
- Some codes may need significant rewrite or refactoring; others gain significantly just adding OpenMP and vectorization
- Profiling/debugging tools and optimized libraries will be essential
- Vectorization important for performance

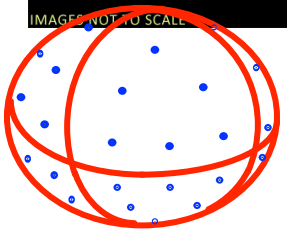
Some App Readiness Initial Specific Findings

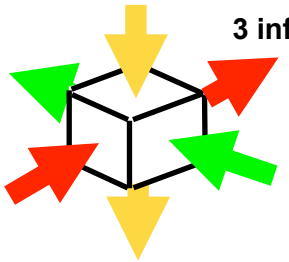
- BerkeleyGW (electronic structure): MPI-only code refactored to have a 3-loop form: 1) a large MPI-parallel loop, 2) a large OpenMP-threaded loop, and 3) a large inner loop that can be vectorized; then required addition of OMP directives and vectorization enhancement on a loop-by-loop basis
- FLASH (hydrodynamics): minimal opportunity for vectorization because subroutines update fluid state *one cell at a time*; requires extensive rewrite; original data layout is array of structures (which is backwards)

SNAP: S_N Transport Application Proxy

- **Proxy for the performance of a modern "discrete ordinates" (S_N) neutral particle transport code**
- **S_N codes solve the Boltzmann transport equation to give the flux of particles (positions, energies,& directions) as a function of time**
- **Examples: neutrons/photons moving through some material(s)**
- **Application in stellar evolution, reactor design, medical radiation treatment, high-energy-density lasers**

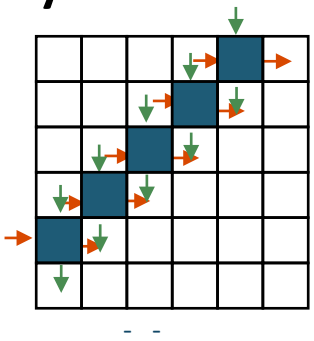




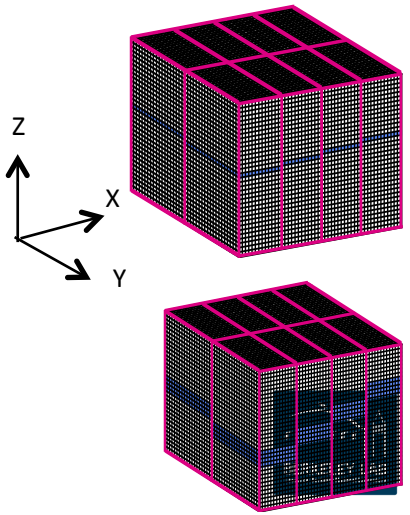


3 inflows & 3 outflows and a cell balance equation

1D



2-D diagonal sweep – idea carried to MPI parallel



U.S. DEPARTMENT OF ENERGY | Office of Science

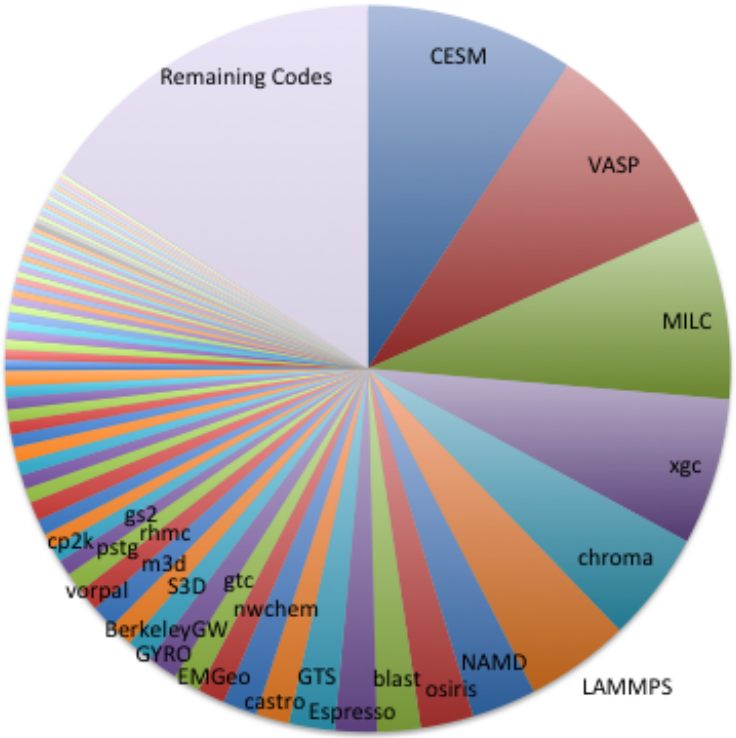
S_N: Changes in Parallelization for MIC

- Complete rewrite required; only one explicit OpenMP parallel loop in old code, which ran over (variable) length of diagonal (between 1 and i*j) as simulation progressed; led to poor OMP performance, no vectorization.

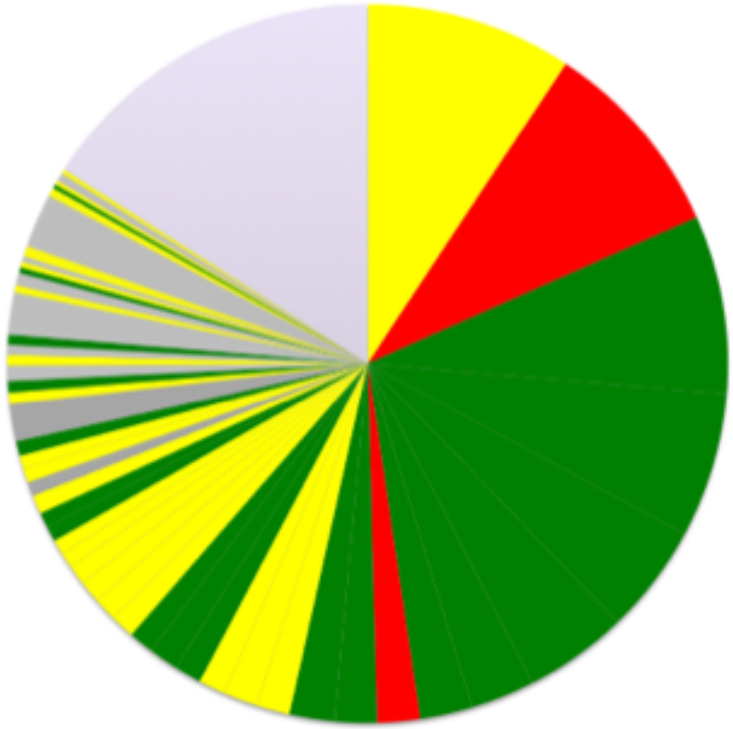
- New code: Invert entire code loop structure; add threads that process energy groups in parallel using OpenMP at outermost level; entire sweep is now a parallel region; innermost loop (over discrete ordinate angles) vectorizes completely

Application Readiness by Code

Workload by Code



Initial Readiness Estimate



Includes 23 key applications with the potential to be ready on Day One

Advanced user; will likely transition via early access to testbeds and training

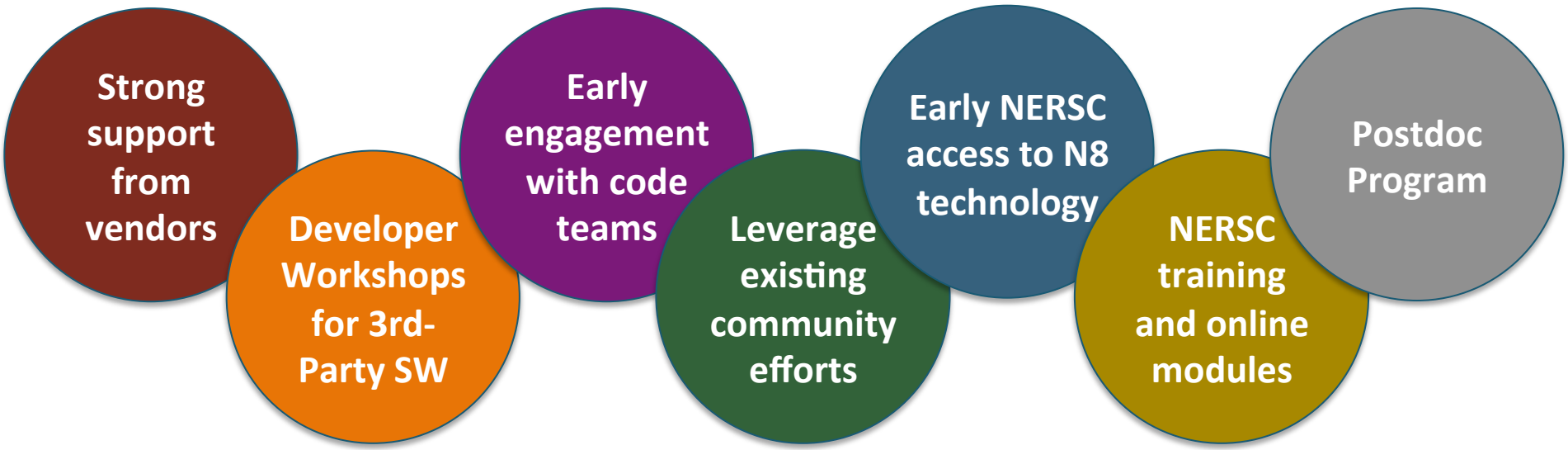
Needs work

More complicated. Offer training, testbeds, but may take longer to port.

Uncertain status

NERSC App Readiness Components

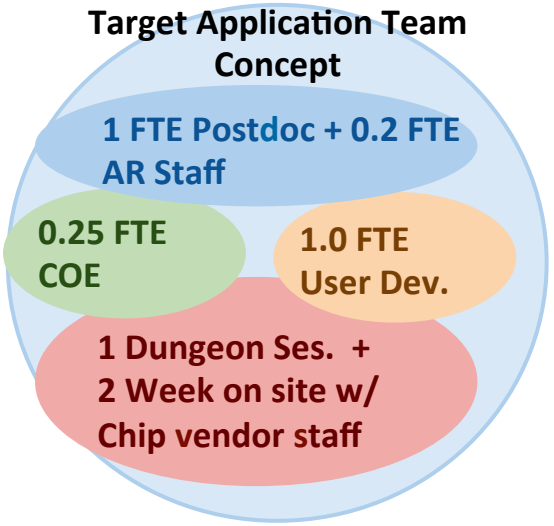
- **Phase 2: Several approaches will prepare our diverse user community for the N8 architecture:**



NERSC is committed to helping our users make this transition

Postdoctoral Research Program: Important for Success of our NERSC-8 System

- We will embed postdocs in application teams at Berkeley Lab for applications that may be more challenging for NERSC-8
- Funding secured from DOE
  - Function: Research basic and applied computer and domain science issues with NERSC PI along with NERSC and vendor staff.
- Wide variety of science domains



What All This Means for Users

- **NERSC will urge users to *begin preparing for N8 now*, using Edison and the "Babbage" Intel MIC test bed system**
- **Users can profile codes; examine vectorization levels and loop lengths; begin to transform loops**
- **Can also examine OpenMP parallelism**
- **Difficult to estimate performance effects, though**
  - If code performs well on Babbage, it will probably perform well on N8

Summary

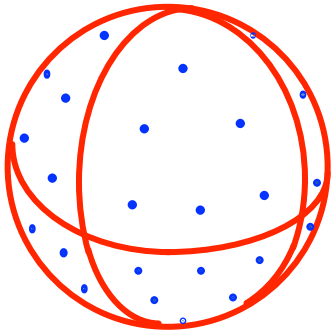
- **Potentially-disruptive programming model changes are coming and will affect computing at all levels; not just about preparing for NERSC-8; is really about preparing for Exascale computing**
- **NERSC's goal is to provide *usable Exascale computing***
- **NERSC's application readiness program will enable users of NERSC-8 to make this transition**

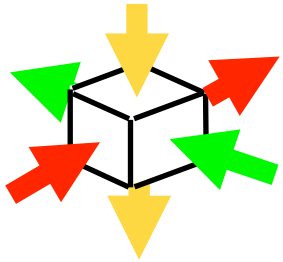
Thank you

A few extras…

S_N Algorithm

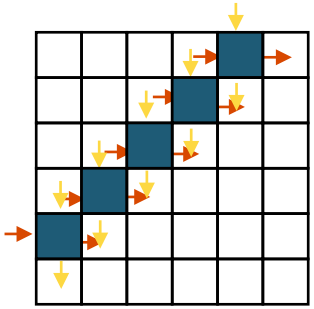
- **Evaluate the flux through a set of discrete directions ("ordinates") $\underline{\Omega}_m$ (e.g., S_6 has 6 per octant)**
- **Also discretize space (3D), energy("groups"), and time: N($\underline{x}$, E, $\underline{\Omega}$, t)**
- **Nested iterative scheme:**
  - Multi-group outer iteration per timestep
  - Source -> transport sweep -> update source per timestep
- **"Sweeps" dominate**
  - solution for a mesh cell depends on flows into that cell from upstream neighbors.
- **Cell-by-cell dependence creates wavefront structure**





3 inflows & 3 outflows and a cell balance equation

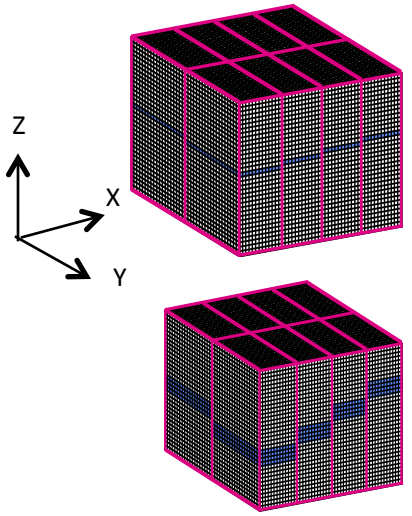
1D



2-D diagonal sweep – idea carried to MPI parallel

S_N Parallelization

- 2-D parallel domain decomposition with MPI
  - 3rd dimension is serial; stack several planes to decrease interprocessor communications
  - Multiple wavefronts active at any one time to increase parallel efficiency
  - Add threading to MPI, not as substitute for it
  - Weak-scale spatial grid; two performance metrics: "Solve" time (should remain constant) and "Grind" time (time per cell/angle/group; should decrease)



Blocking in "z" leads to tradeoff: Parallel Efficiency vs. Communication Intensity

Problem Sizes

Desired ranges for real work:		This study on KNC testbed:	
Cells:	$O(10^6 \rightarrow 10^9)$	Cells:	200 - 18,000
Groups:	$O(10^1 \rightarrow 10^2)$	Groups:	32
Angles:	$O(10^2 \rightarrow 10^3)$	Angles:	50 - 150
Time:	$O(10^3 \rightarrow 10^4)$	Time:	10

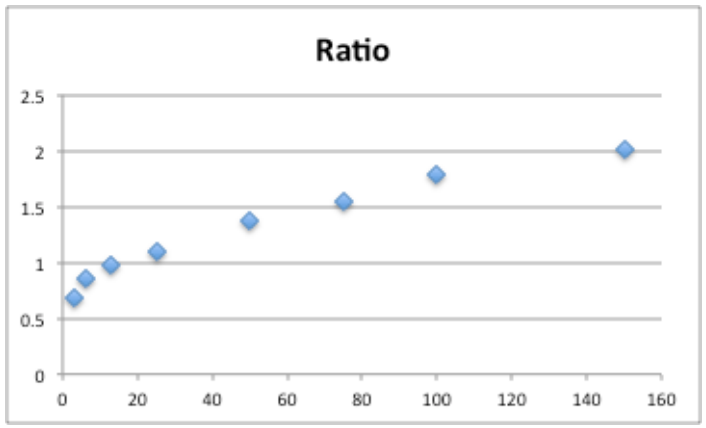
S_N Vectorization Analysis

- Key ingredient in new code: innermost loop is over angles, vectorizes completely; eliminates recursion found in old code

Schematic view of recursion in old code: flux in cell ***i*** depends on ***i*** **– 1; compiler says, "`sweep.f (401): (col. 10) remark: loop was not vectorized: existence of vector dependence on phiir;`" this dependence is eliminated in new code due to complete restructuring**

- The following shows the improvement afforded by vectorization on KNC as a function of number of angles (Na); N.B. these results obtained BEFORE obtaining best sequential performance
- **Ratios range from 0.69 (vectorization is worse) to 2.0**
- Problem is memory capacity:
  8 x 4 x 6 x 32 x 50 problem exhausts KNC memory at 192 cores

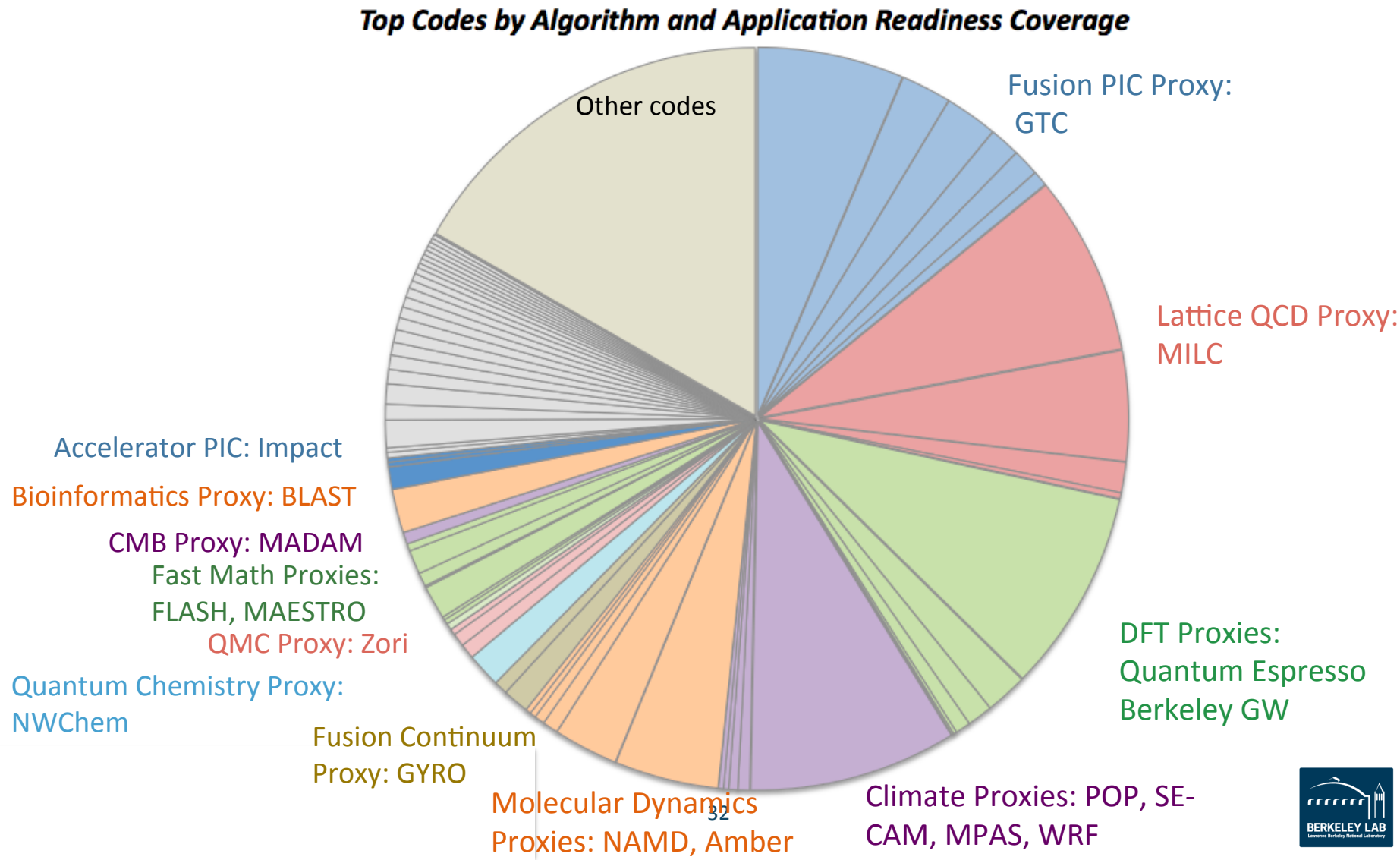
Improvement Factor
Due to Vectorization



Number of angles (length of vector loops)

- Compile with `-no-vec -no-simd` to see vectorization effect

The App Readiness team proxies cover almost 75% of the workload



Application	Assessment of readiness	Comments on advanced architecture application readiness	% Cum
CESM		CamSE port to Titan, vendor interest, large complex code	9%
VASP		GPU solver for exchange correlation, closed code	18%
MILC		GPU version, MIC version in progress, BG/Q port, OpenMP enabled	26%
XGC		SciDAC funded project, vendor interest	33%
Chroma		GPU Version, MIC version, SciDAC project	38%
LAMMPS		GPU version, Part of Titan App Readiness	42%
NAMD		GPU Version, MIC version, BG/Q port, highly engaged team	45%
Osiris		2D GPU version, 3D in progress, OpenMP enabled, BG/Q port, SciDAC funded	48%
BLAST		Engaged NERSC staff, vendor interest but high throughput application	50%
Espresso		Funded NERSC App Readiness staff, Ported to GPU, MIC port in progress	51%
GTS		Running on BG/Q, SciDac project, OpenMP Enabled, GPU port	53%
CASTRO		OpenMP enabled	55%
NWChem		SciDAC project, NERSC funded App Readiness staff, Intel MIC port, vendor interest	57%
EMGeo		GPU implementation	58%
GTC		OpenMP enabled, GPU version, Starting MIC port, BG/Q port	59%
GYRO		NERSC funded app readiness code, OpenACC directives added, BG/Q port, SciDAC funded	60%
BerkeleyGW		NERSC funded app readiness code, OpenMP enabled, GPU enabled, MIC port	61%

Survey results of readiness of top codes for manycore architectures

- **Green**
  - Application currently running on GPUs, MICs, highly-threaded BG/Q or;
  - Active ***and*** funded development underway to transition algorithms, or;
  - Highly engaged application developer community
- **Yellow**
  - Some activity begun to transition application, or;
  - Activity underway, but highly complicated code, or;
  - No known development underway, but similar application/algorithm to code that has been transitioned
- **Red**
  - Concerns about readiness of application
  - Closed code teams, or;
  - Application not well aligned to many-core architectures

NERSC workload is highly concentrated making transition less daunting

- **NERSC workload is unevenly distributed**
  - 10 codes make up 50% of workload
  - 25 codes make up 66% of workload
  - 75 codes make up 85% of workload
- **How well are these code prepared to take advantage of future architectures?**

Top Application Codes by Hours Used

Hopper System Jan – Nov 2012