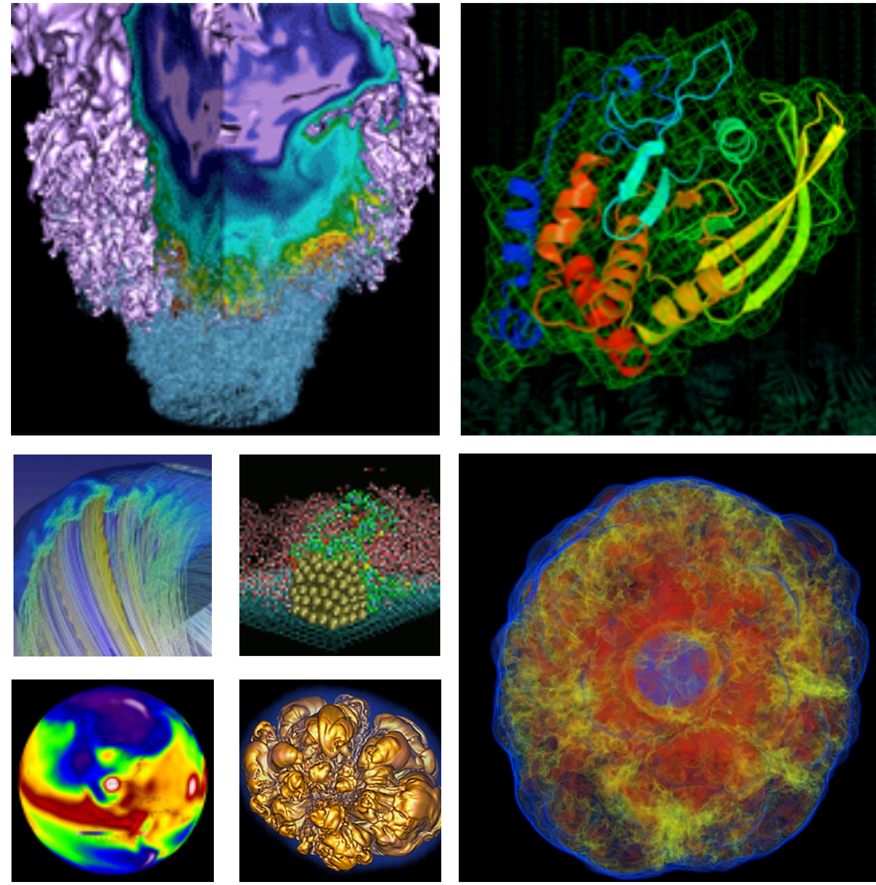
NUG Monthly Meeting



NERSC 40 YEARS at the FOREFRONT 1974-2014

Richard Gerber, Lisa Gerhardt, Harvey Wasserman, Helen He, Scott French, Zhengji Zhao

NUG Monthly Meeting
September 11, 2014

U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB
Lawrence Berkeley National Laboratory

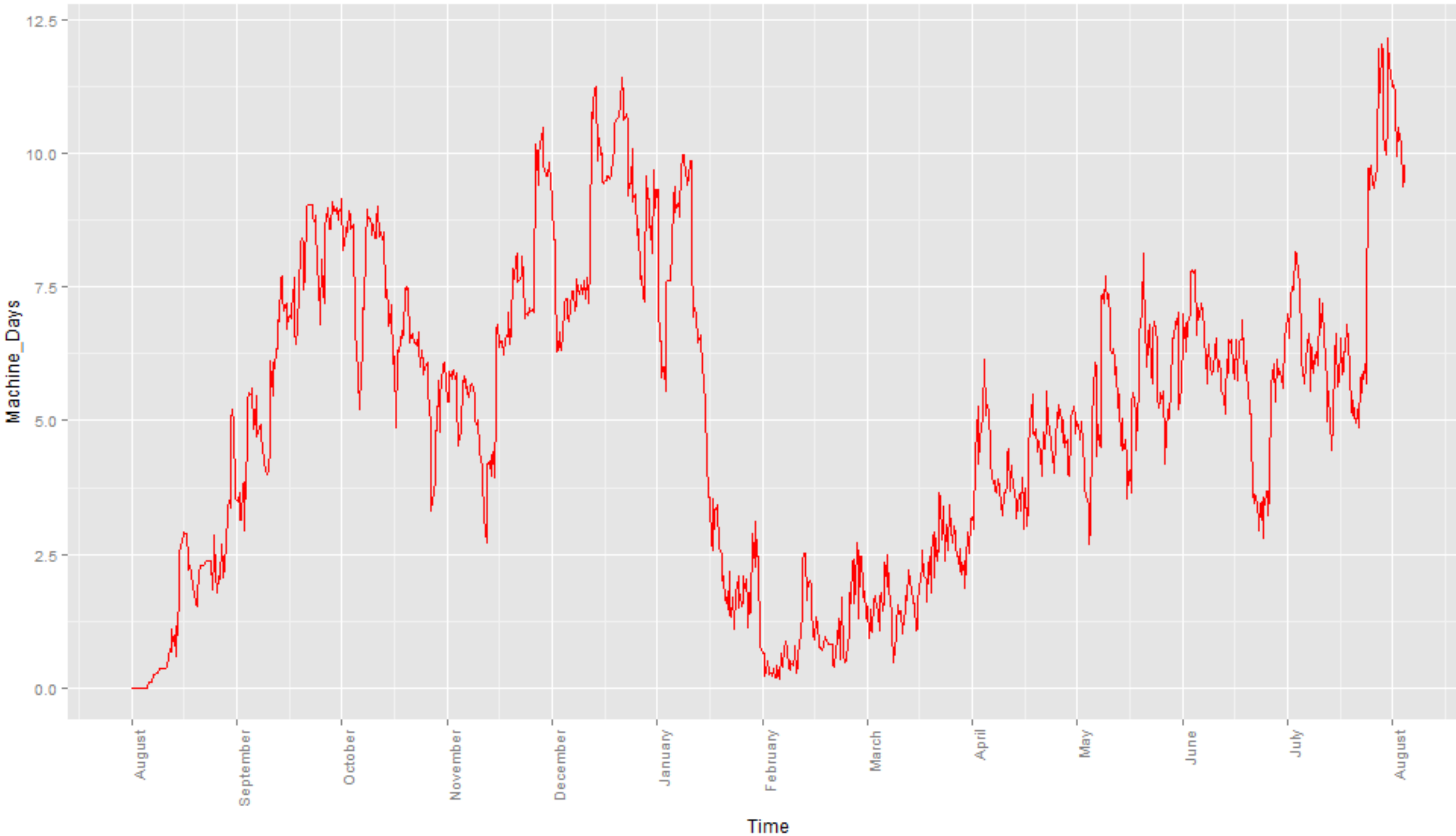
Agenda

- Hopper and Edison Utilization, Backlog, and Queue Waits
- Edison memory replacement: downtime 9/25/14-9/29/14
- Carver SL6 OS upgrade and CHOS
- Hopper apsched errors
- Update on the NESAP program and NERSC Application Readiness for Cori (NERSC-8)
- Dirac and Carver retirement reminder
- NUGEX Elections
- Mini-Seminar: Programming for high-level and fine-grained parallelism with MPI, OpenMP, & UPC

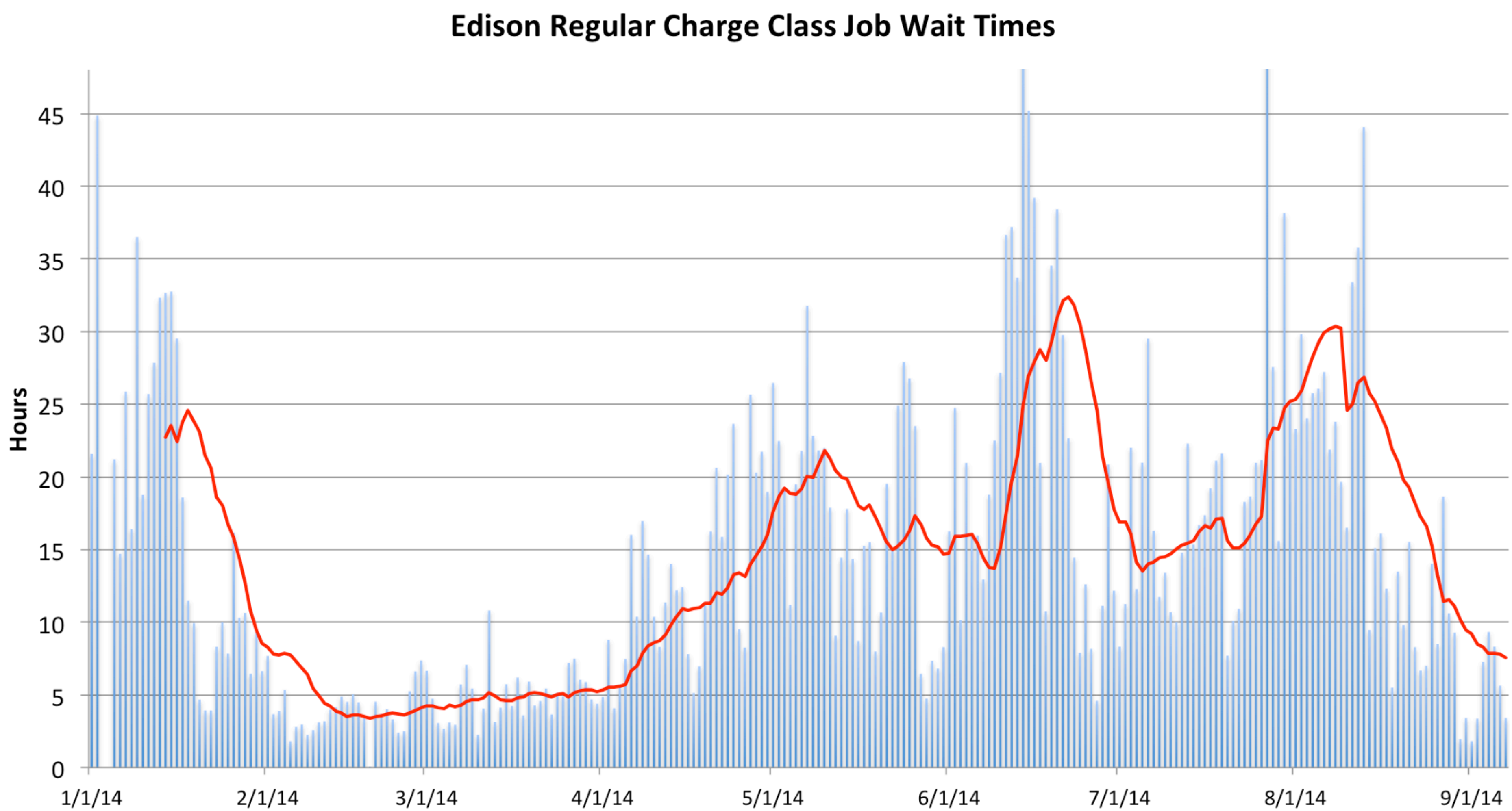
Long Waits on Edison & Low Utilization on Hopper

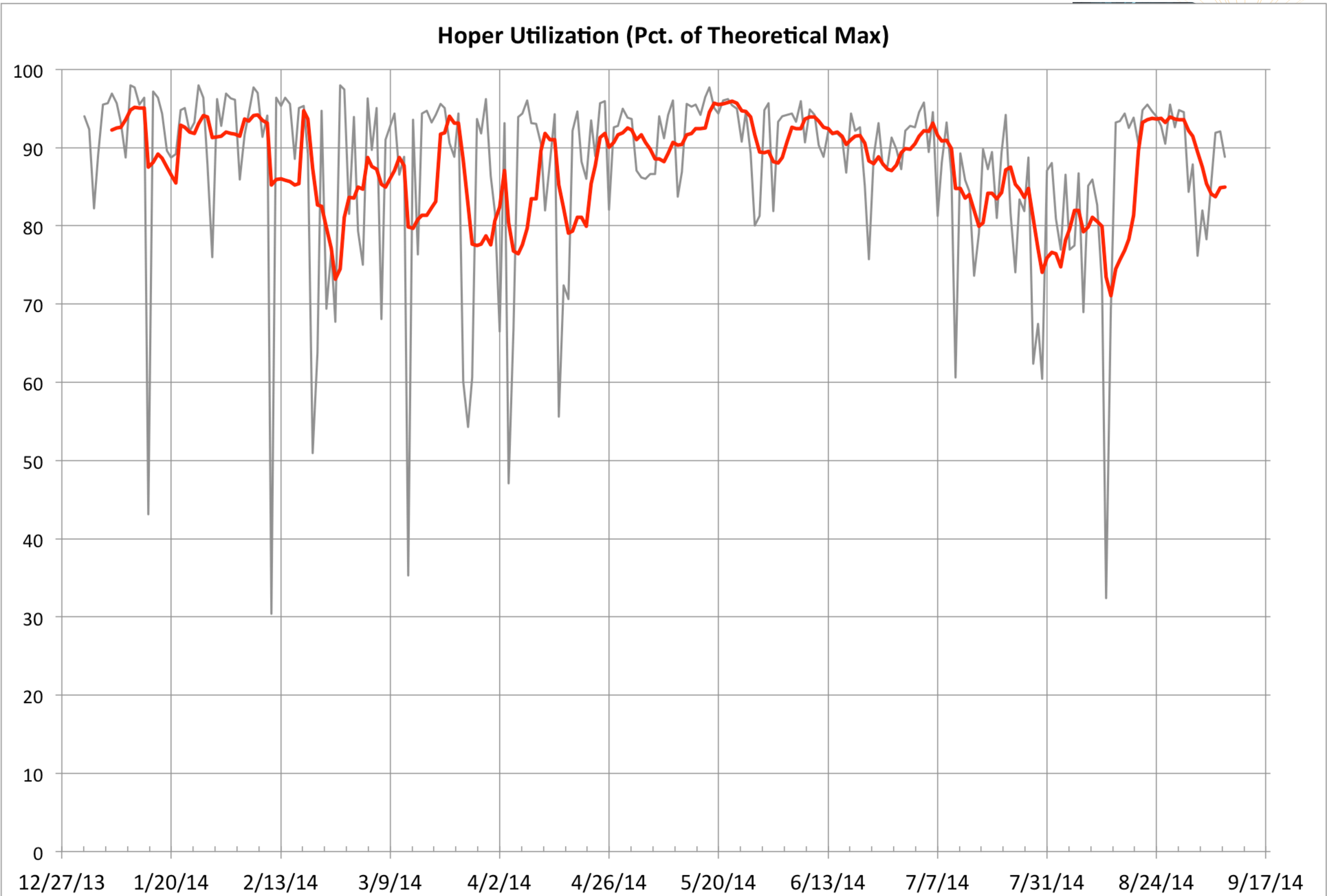
- **By late June Edison wait times had increased dramatically**
- **At the same time Hopper utilization was "low" (still close to 80-90%!)**
- **NERSC took action on August 19**
  - Queue and run limits were relaxed on Hopper
  - Hopper regular charge jobs were discounted 20%.
  - Run limit on Hopper low queue increased to 48 hours

Edison Backlog



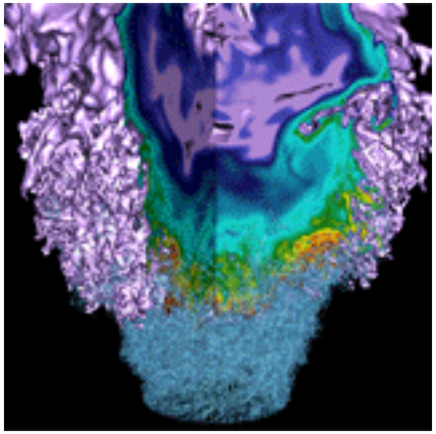
Edison Wait Times

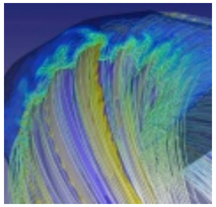


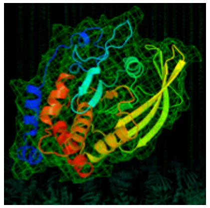

Hoper Utilization (Pct. of Theoretical Max)
100
90
80
70
60
50
40
30
20
10
0
12/27/13
1/20/14
2/13/14
3/9/14
4/2/14
4/26/14
5/20/14
6/13/14
7/7/14
7/31/14
8/24/14
9/17/14

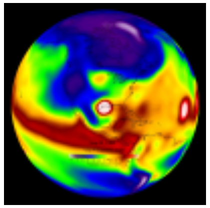
Edison Memory Replacement and Outage

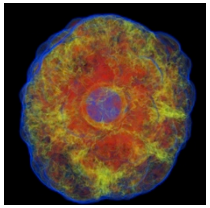
Zhengji Zhao

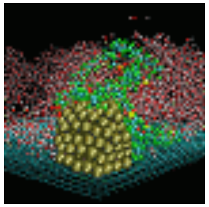












NERSC 40 YEARS at the FOREFRONT 1974-2014

U.S. DEPARTMENT OF ENERGY | Office of Science

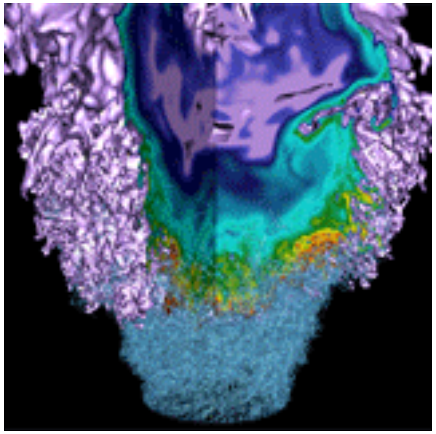
BERKELEY LAB

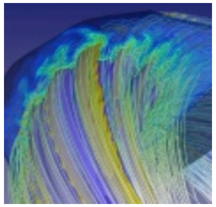
Edison Memory Replacement

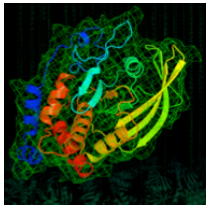
- **REMINDER: Edison outage 9/25/14 to 9/29/14**
- **We're upgrading memory to support 1866 MHz memory clock speed (currently running at 1600 MHz)**
- **16.6% increase in memory bandwidth (streams)**
- **Will require another partial outage in early 2015, at which point the memory speed will be increased to 1866 MHz**

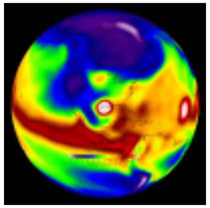
SL6 and chos on Carver

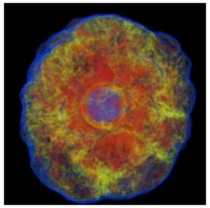
Lisa Gerhardt

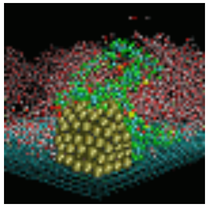












Carver's Current Status

- **On Monday, August 18th Carver's base OS was upgraded from Scientific Linux 5.5 to Scientific Linux 6.4**
- **Expanded to offer two user environments**
  - Users can choose which OS they want
  - Scientific Linux 5.5 (same as before)
  - Scientific Linux 6.3
  - Done using CHOS
- **Carvergrid is still on original OS, will be upgraded to SL6 and CHOS soon**

What is CHOS?

- **Software stack that allows support of many different OS's simultaneously**
- **Can be thought of as essentially a chroot to an alternate OS (CHroot OS)**
  - File systems, batch integration
  - Seamless to the user
- **Successfully used on PDSF since 2004**

Why go to CHOS?

- **Allows us to offer newer software while still supporting older software**
- **Newest versions of some of our more popular software were not installable under SL5**
  – Matlab, IDL
- **Greatly simplifies underlying architecture for system administration**
  – Can install software updates without perturbing user systems
  – System software has a smaller memory footprint on the compute nodes
  – For Carver, were able to update underlying OS to Scientific Linux 6.4

Interacting with CHOS

- **Users are in CHOS from the beginning of a session**
  – ssh starts chos as part of logging in
- **Your CHOS is determined by a ".chos-carver" file in your home directory**
  – Current default is SL5, "sl5carver" in .chos-carver
  – SL 6, "sl6carver"
  – No .chos-carver file, get the default CHOS
- **Use "chosenv" to see what CHOS you're in**

This is a lower case L.

Changing CHOS

- **Users can change CHOS at will**
- **bash:**

  export CHOS=sl6carver

  chos

  bash –l
- **csh, tcsh**

  setenv CHOS sl6carver

  chos
- **For long term running, it's recommend to put chosen CHOS in .chos-carver and get a fresh login**

Submitting Jobs with CHOS

- **Your batch jobs will run in whatever CHOS you're in when you submit**
- **Possible to run in another CHOS**
  - qsub -v CHOS=sl6carver <your_job.script>
  - Add "#PBS -v CHOS=sl6carver" to top of job script

Cron Jobs with CHOS

- **If CHOS is not declared your cron jobs will run in minimalist base CHOS**
  - No modules, very limited software stack

0 */6 * * * CHOS=sl6carver chos <your_cron>

Carver CHOS Documentation

http://www.nersc.gov/users/computational-systems/carver/user-environment/

USER ENVIRONMENT

There are two primary ways that users can control their environment: CHOS and modules.

CHOS

Carver runs Scientific Linux 6.4 as its native operating system. The native operating system is not intended for general use. Instead, the *chos* utility is used to create a Scientific Linux environment on both the login nodes and in batch jobs. Currently Scientific Linux 5.5 (sl5carver) is the default. After September 22, 2014, the default will be Scientific Linux 6.3 (sl6carver).

To automatically select a system version you need to create a file in your home directory named **.chos-carver** (with the dot at the beginning). In this file you should have one and only one line:

In your .chos file:	The operating system you get:
sl5carver	64-bit Scientific Linux 5.5
sl6carver	64-bit Scientific Linux 6.3

When you log in you should have a full working environment with the OS of your choice.

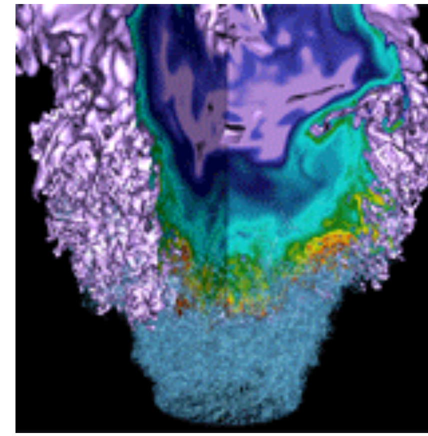
For most day-to-day work, the above approach is sufficient. However, there are times when it may be necessary to move between OS versions, for example, when migrating applications

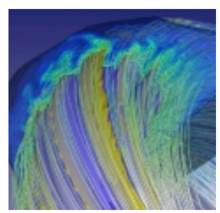
Future Plans

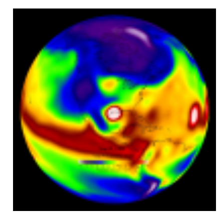
- **Current default CHOS is sl5carver (same as before upgrade)**
  - Users who do nothing end up in this CHOS
- **All new software installations will be in sl6carver**
- **Tentative plan is to change the default to sl6carver on 9/22**
  - PRO: New users will automatically start in newer software, Encourages existing users to upgrade to new software (SL 5 is becoming less widely supported)
  - CON: Users will have to take action, either recompile their code or adding a .chos-carver file to stay in SL 5.5
- **We would like NUG's recommendation about whether to change the default to sl6carver (SL 6.3)**

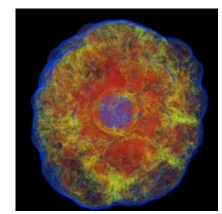
Hopper scheduler issues

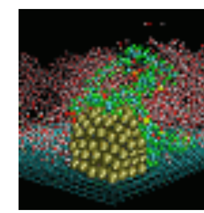
Helen He











NERSC 40 YEARS at the FOREFRONT 1974-2014

U.S. DEPARTMENT OF ENERGY | Office of Science

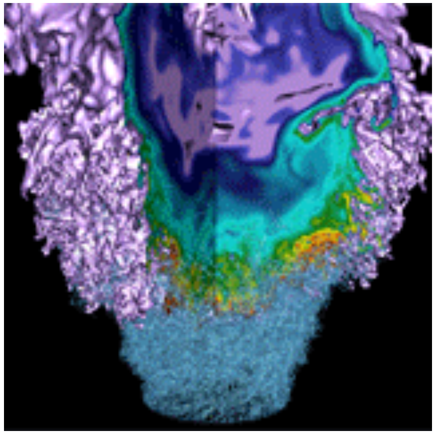
BERKELEY LAB

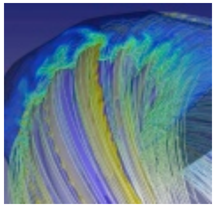
“apsched” errors on Hopper

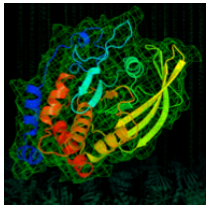
- **Users getting the following error message intermittently**
  - “apsched: request exceeds max nodes, alloc”
  - started in April, later in mid July, and recently again from late August
  - Problem identified as Torque/Moab batch scheduler becomes out of sync with the ALPS (the Cray Application Level Placement Scheduler) reservation status. A bug has been filed with the vendor.
  - This bug affects both Hopper and Edison. However, users get fewer errors on Edison:
    - A) Edison has 7-digit reservation ids for ALPS. Hopper will have this feature after an OS upgrade (early next year)
    - B) There is a system cron job updates ALPS internal table of the batch status on Edison. We added this on Hopper on Sept 8.
  - We are still seeing this error, but fewer.
- **However, new error message seen from Sept 10:**
  - “apsched: no resource confirmation entry for resId xxxx was found”
  - Cause unknown, do not know if related to the workaround B above. Under investigation.

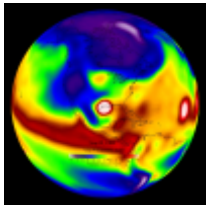
NESAP & Application Readiness

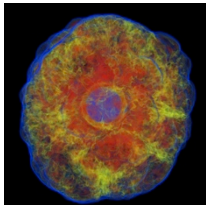
Harvey Wasserman

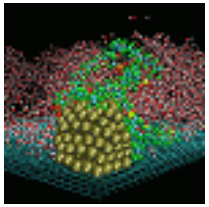












NESAP Has Begun

- Purpose: Get codes (more) ready for manycore systems
- Accelerate application performance
- Produce science results on Cori
- Collaboration between code groups, NERSC, and vendors
- Over 50 application teams applied.
- Twenty teams accepted for collaboration, early access, deep-dive consultation, early access to hardware
- About 25 more accepted for early access to hardware
- DOE program manager input and interest in results
  - Many highly qualified teams not accepted at this level
- Accepted projects span science areas, representation in workload (NERSC/DOE/elsewhere), current readiness for manycore architecture
- See NERSC.gov -> News -> NERSC Center -> "NERSC Selects 20 NESAP Code Teams"

20 NESAP Collaboration Codes

ASCR (2)

Almgren (LBNL) – **BoxLib AMR Framework** used in combustion, astrophysics

Trebotich (LBNL) – **Chombo-crunch** for subsurface flow

BES (5)

Kent (ORNL) – **Quantum Espresso**

Deslippe (NERSC) – **BerkeleyGW**

Chelikowsky (UT) – **PARSEC** for excited state materials

Bylaska (PNNL) – **NWChem**

Newman (LBNL) – **EMGeo** for geophysical modeling of Earth

BER (5)

Smith (ORNL) – **Gromacs** Molecular Dynamics

Yelick (LBNL) – **Meraculous** genomics

Ringler (LANL) – **MPAS-O** global ocean modeling

Johansen (LBNL) – **ACME** global climate

Dennis (NCAR) – **CESM**

HEP (3)

Vay (LBNL) – **WARP & IMPACT-** accelerator modeling

Toussaint (U Arizona) – **MILC** Lattice QCD

Habib (ANL) – **HACC** for cosmology

NP (3)

Maris (U. Iowa) – **MFDn** *ab initio* nuclear structure

Joo (JLAB) – **Chroma** Lattice QCD

Christ/Karsch (Columbia/ BNL) – **DWF/HISQ** Lattice QCD

FES (2)

Jardin (PPPL) – **M3D** continuum plasma physics

Chang (PPPL) – **XGC1** PIC plasma

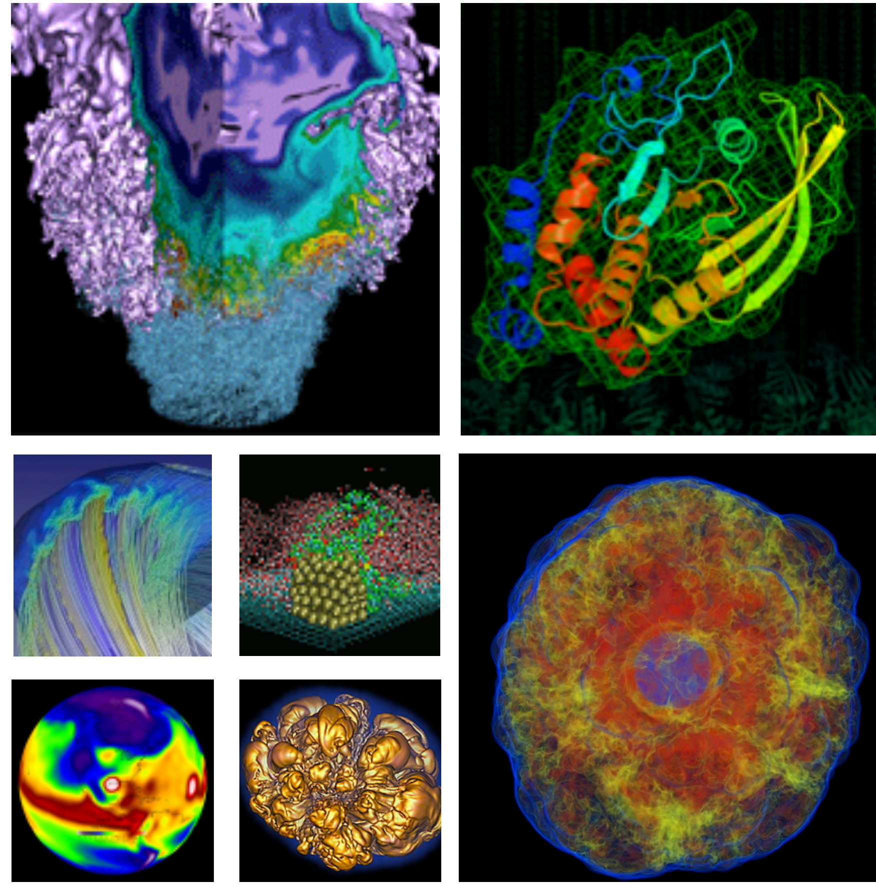
Carver and Dirac Retirement Reminders

- **Carver will be retired on August 31, 2015**
  - Transition your code and workflows to Edison
  - Tell us if you can't run on Edison or Hopper
  - Plans and advice:
    http://www.nersc.gov/users/computational-systems/carver/retirement-plans/
- **Dirac will be retired Friday, Dec. 12, 2014**
  - Queues will stay open to almost the end to allow shorter jobs to be run to the end.
  - 2014-12-12: Dirac power off
    - 10:00 Queues disabled
    - 17:00 System power off

NUGEX Elections

- **Eight seats on NUGEX are up for election in December 2014**
  - Fusion – 2: Ethier, Vay
  - High Energy Physics – 3: Borrill, Gottlieb, Tsung
  - Nuclear Physics – 2: Kasen, Savage
  - At large – 1: Newman
- **Contact Frank Tsung (tsung@physics.ucla.edu) if you are interested in running for one of these spots.**

Evolution of parallel programming models in a legacy scientific application



NERSC 40 YEARS at the FOREFRONT 1974-2014

Scott French

NERSC User Services Group

NUG Monthly Teleconference
September 11, 2014

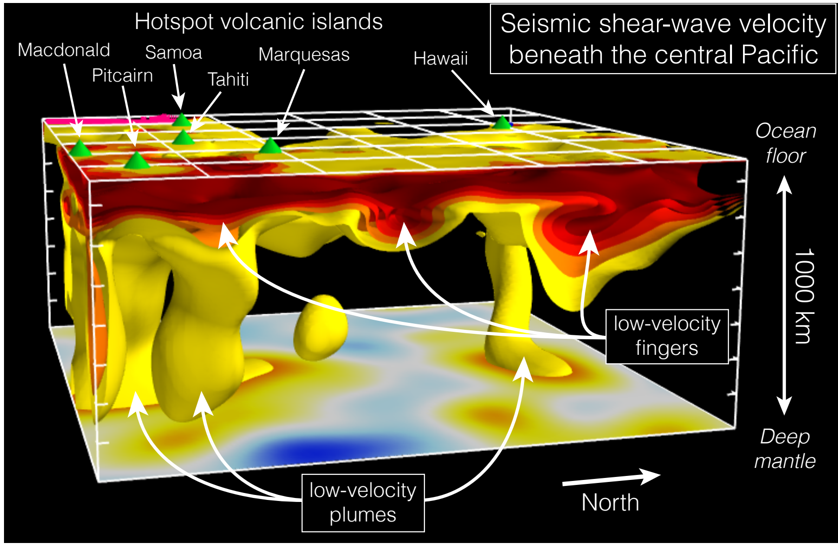
U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB
Lawrence Berkeley National Laboratory

Application: Global seismic tomography

- **Scientific goal:** To better understand the evolution and interior dynamics of our planet by imaging its deep structure
- **Technique:** *Waveform tomography*
  - **Objective**: Model of material properties
  - **Observations**: Seismograms of natural earthquakes (hundreds)
  - **Predictions**: Numerical simulations of seismic wave propagation

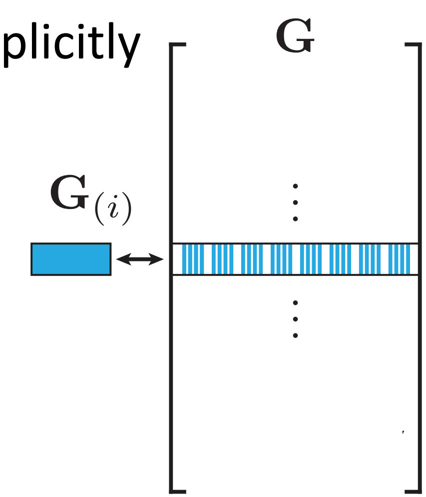
French et al., 2013, Science



- **Non-linear inverse problem**
  - Prediction (spectral finite element) is expensive: 500K – 1M hours
- **Iterative optimization method should converge quickly**
  - Typically want ≤ 10 iterations (two phases each: prediction, assimilation)
  - Traditionally use a *Gauss-Newton* scheme in assimilation phase

Optimization via Gauss-Newton

- **Typical problem size:** N_m = 1e4 – 2e5 earth-model parameters
  - Factorization of Gauss-Newton Hessian (N_m x N_m) feasible in this regime, avoids matrix-free (too many maps over data) or quasi-Newton (too many iterations)
- **How to assemble the Gauss-Newton Hessian $\mathbf{G}^{\mathrm{T}}\mathbf{G}$?**
  - $\mathbf{G}$: matrix of partial derivatives relating predictions to the earth model
    - **Size**: dimension of data (1e7) x number of parameters (N_m)
  - Each datum (a seismogram) supplies one column-strided panel of $\mathbf{G}$
  - Unfortunately, $\mathbf{G}$ is *non-sparse* and *too large* to form explicitly
- **Solution: Form $\mathbf{G}^{\mathrm{T}}\mathbf{G}$ directly**
  - Reduces storage requirements significantly over forming $\mathbf{G}$
  - Repeated indexed augmented assignment (+=) into $\mathbf{G}^{\mathrm{T}}\mathbf{G}$



Index i is one **datum**

Evolution of programming models

more data

higher resolution

Late 1990's	Sequential solution
Mid 2000's	Parallelized, replicated Hessian estimate • *MPI for work coordination and Hessian reduction*
2010	One MPI process / Hessian per NUMA domain • *OpenMP threads compute per-waveform updates* • *Still MPI for work coordination and Hessian reduction*
Late 2013	Hessian **no longer fits** on a single compute node …

- **Requires a distributed solution**: Must support assembly from concurrent updates with *data-dependent* indexed access patterns
- **A number of *simplifying assumptions* can be made**
  - Updates are **independent** (data parallel), **commutative,** and **associative**
  - **No loads / gets** of distributed matrix elements during assembly
    - State only needs to converge once all updates are "committed"
    - Thereafter, dependent computations can start (e.g. ScaLAPACK)

Implementation: Goals and requirements

- **Many implementation strategies, a scalable solution should:**
  - Exploit **simplifying assumptions**
  - **Overlap** computation and communication
  - **Minimize synchronization**
    - Load balance is difficult to achieve – no bulk synchronous exchange
    - No coordination aside from dynamic work distribution
- **Requirements for a distributed matrix abstraction**
  - Support for block-cyclic etc. distributions (ScaLAPACK, MPI-IO)
  - Should fit seamlessly into the production application
    - OpenMP and MPI interoperability
    - > 95% of application is in C, would prefer to stay in this language family
  - Ensure isolation of concurrent += updates, parameterized by indexed strided-slicing operations: e.g. `GtG[ix,ix] += GtG_i[:,:];`

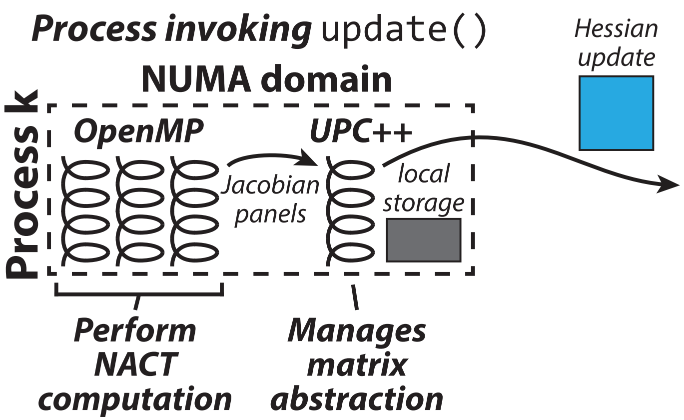
Implementation: Design and interface

- **Solution adopts the <u>Partitioned Global Address Space</u> model**
  - Motivated by fast **non-blocking remote memory access**
  - Chose **UPC++**, a set of PGAS extensions to C++ (Zheng, et al. IPDPS'14)
  - Modeled largely on UPC (and others, e.g. X10), but adds:
    - *Dynamic remote memory management (allocate / free on remote target)*
    - *Asynchronous remote task invocation (schedule code to run on remote target)*
  - Interoperable with MPI and OpenMP (usual caveats on mixing RTs)
- **Distributed matrix abstraction:** `ConvergentMatrix`
  - In a nutshell, two-phase one-sided updates:
    - **Phase I**: Buffer allocated on owner (target); += r.h.s. data copied to target
    - **Phase II**: Async task applies update on target in isolation (frees buffer)
  - Simple interface: <u>`update`</u> *initiates update*, <u>`commit`</u> *ensures completion of prior updates (collective)*, and <u>`get_local_data`</u> *returns ptr to local matrix data*

Implementation: Design and interface

An illustrative example

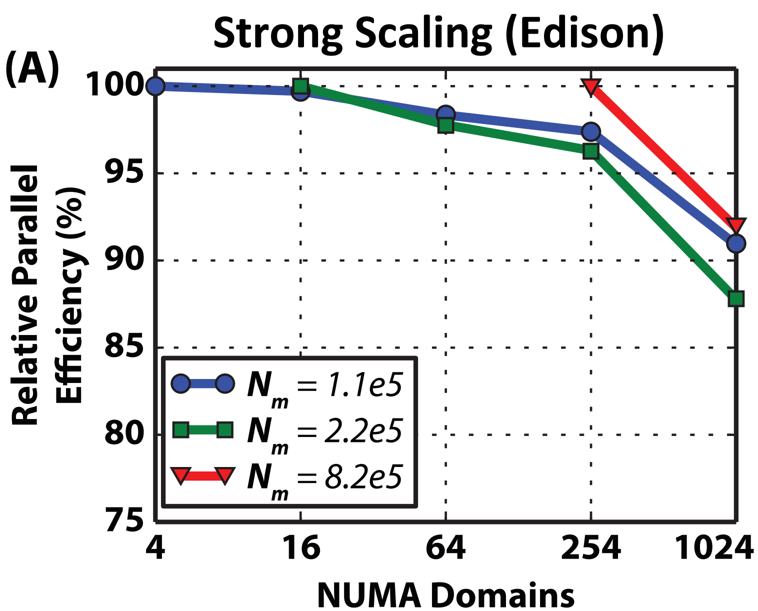
- Example follows the path of a single matrix update
- Configuration: One process per NUMA domain, but now UPC++




```
ConvergentMatrix<float,...> GtG( M, M );
...
// for each locally computed update
GtG.update( GtG_i, slice_idx_i );
```

Evaluation: Strong scaling

- **Approach: Abstract away application**
  - Test framework generates synthetic updates: Realistic Hessian sizes (up to next-gen ≥2.5TB), access-patterns, update rates, concurrency levels



- GNU Compilers 4.8.2 (-O3)
- GASNet-1.22 / UPC++ master

In terms of relative parallel efficiency:

$$E_R(P) = \frac{T(P_0)}{P/P_0 \cdot T(P)}$$

$N_{\mathrm{m}} = 1.1 \times 10^5$

P	**Cores**	N_{up}	$T(P)$ s	$E_R(P)$	N_{up}	$T(P)$ s	$E_R(P)$
4	48	4096	5070.59	100.0%	32768	39948.20	100.0%
16	192	4096	1271.40	99.7%	32768	10016.61	99.7%
64	768	4096	322.24	98.3%	32768	2538.74	98.3%
256	3072	-	-	-	32768	640.96	97.4%
1024	12288	-	-	-	32768	171.68	90.9%

$N_{\mathrm{m}} = 2.2 \times 10^5$

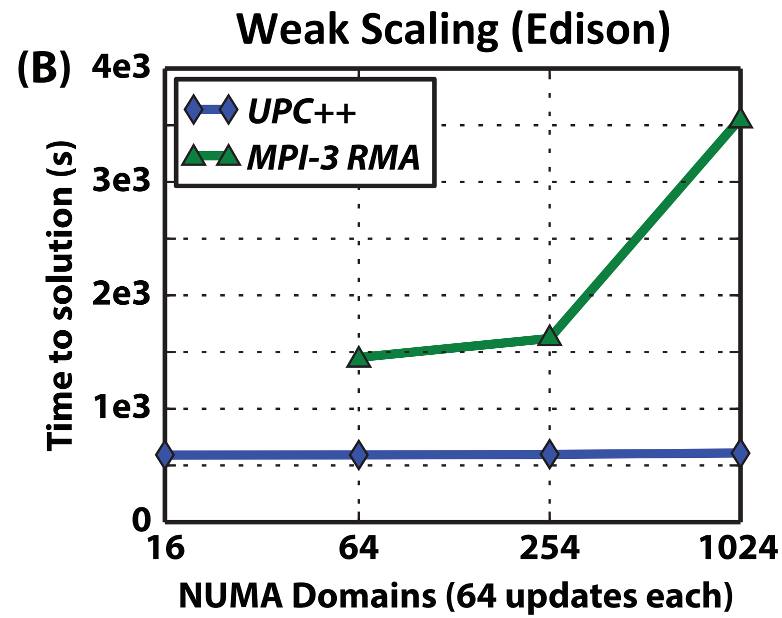
P	**Cores**	N_{up}	$T(P)$ s	$E_R(P)$	N_{up}	$T(P)$ s	$E_R(P)$
16	192	4096	2318.57	100.0%	32768	18079.84	100.0%
64	768	4096	592.80	97.8%	32768	4622.56	97.8%
256	3072	-	-	-	32768	1173.27	96.3%
1024	12288	-	-	-	32768	321.92	87.7%

$N_{\mathrm{m}} = 8.2 \times 10^5$

P	**Cores**	N_{up}	$T(P)$ s	$E_R(P)$	N_{up}	$T(P)$ s	$E_R(P)$
256	3072	32768	2399.96	100.0% 16	65536	4703.16	100.0%
1024	12288	32768	703.72	85.3% 32	65536	1279.66	91.9%

Alternative implementation: MPI-3 RMA

- **Functional requirements met with MPI-3 RMA (similar # SLOC)**
  - `MPI_Accumulate` + `MPI_SUM` and passive `MPI_Win_lock` / `unlock`
    - **Pro:** UPC++ / GASNet RTs not needed
    - **Pro:** Elemental atomicity: *MPI RT has more freedom in scheduling updates?*
    - **Con:** Elemental atomicity: *Element-wise concurrency control?*
    - **Con:** Black box: *Design tradeoffs sub-optimal for our use case? (e.g. locality implications of true passive target)*
- **Right: weak scaling (dataset size)**
  - 64 updates / NUMA domain
  - Matrix size held fixed: N_m = 2.2e5
    - GNU Compilers 4.8.2 (-O3)
    - Cray MPI 6.2.0 (MPI-3)



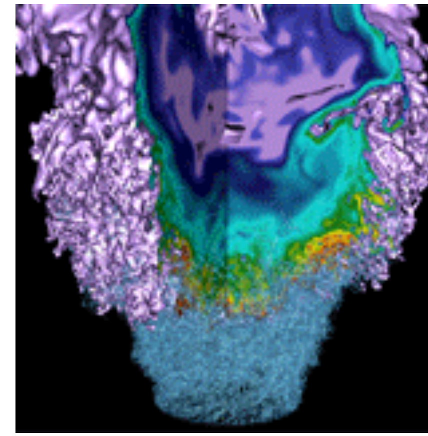
			UPC++	MPI
P	Cores	N_{up}	$T(P)$ s	$T(P)$ s
16	192	1024	591.18	fail
64	768	4096	592.50	1452.24
256	3072	16384	597.24	1620.22
1024	12288	65536	609.96	3560.28

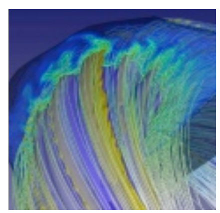
** Overflow in `MPI_Type_Indexed`

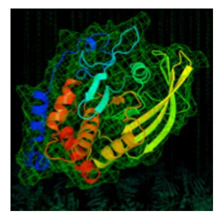
Discussion and Conclusions

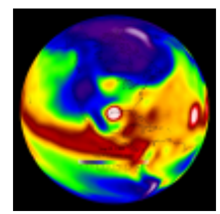
- **PGAS-based solution enables us to solve problems we could not have attempted otherwise**
  - Yielded the first-ever seismic model Earth's mantle obtained using SEM-based waveform tomography (*French and Romanowicz, 2014, GJI accepted*)
  - Ready to scale to the next generation of problem size
- **Broader implications for HPC**
  - Illustrative example: ***Progressive adoption of mixed-model parallelism*** to confront / exploit architectural changes and adapt to changing scientific goals
    - {MPI} → {MPI + OpenMP} → {MPI + OpenMP + PGAS}
  - Application fits into an ***increasingly common motif***: Data-driven concurrent computations that update shared global state with complex access patterns
  - UPC++ feature-set ***enables novel solutions to such problems*** and an provides an easy ***onramp to adoption of the PGAS model***
    - Familiar / popular language (C++), interoperability with MPI and OpenMP, etc.

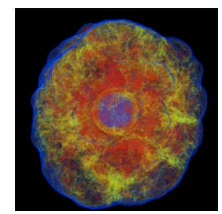
Extra Slides

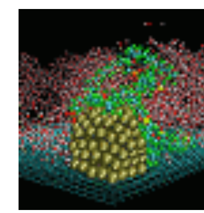












Challenges: Ensuring progress

- **Progress in the asynchronous task queue**
  - When are asynchronous tasks actually executed?
    - Implications for memory management: When will the receive buffers be freed?
  - Solutions for finer control over task queue:
    - `peek()` / `drain()` for querying / flushing the queue
    - Progress thread: runs in the background, executing remotely enqueued tasks
- **Progress in GASNet**
  - GASNet Active Messages handlers required for: (a) tasks to enter queue on target and (b) remote memory allocation on target (not for copy)
    - AM polling within UPC++ (and implicitly within GASNet ops)
  - Progress thread assists GASNet progress (`peek()` induces polling)
- **Potential for deadlock**
  - Communications operations separate across runtimes
    - Separate *in time* or concurrent but handled by *different threads*
  - Low probability of classic deadlock problem when mixing parallel RTs

More on MPI implementation

- **Why not `MPI_Win_flush`?**
  - Still need to lock to start passive epoch; either
    - Redundant lock / unlock with `MPI_LOCK_EXCLUSIVE`
    - Global (whole run) lock / unlock with `MPI_LOCK_SHARED` (slow!)
- **Why not `MPI_Raccumulate` for "async" update?**
  - Still need to check on it; again either:
    - Redundant lock / unlock with `MPI_LOCK_EXCLUSIVE`
    - Slow global epoch lock / unlock with `MPI_LOCK_SHARED`
- **How about faster memory?**
  - Already use `MPI_Alloc_mem`
  - Maybe `MPI_Win_allocate`?
    - Good question! Trying that
- **How about window optimizations?**
  - Say, using `accumulate_ops = same_op`?
    - Trying that too!

Thank you.