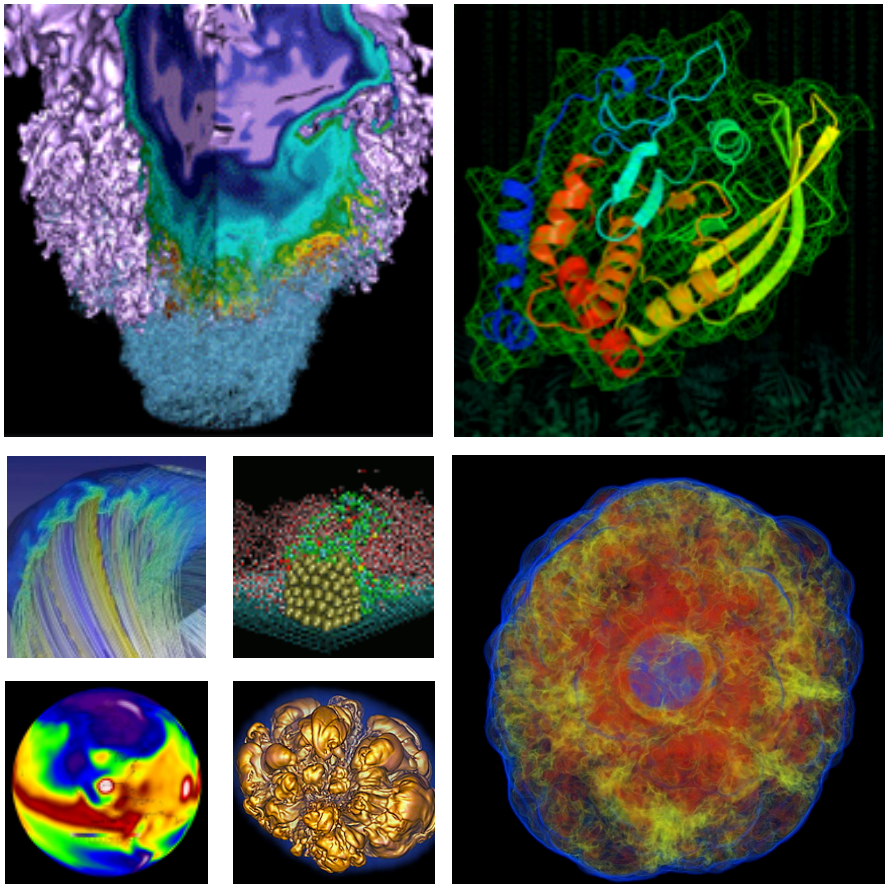
Getting Started at NERSC



Richard Gerber
NERSC User Services

January 17, 2013

U.S. DEPARTMENT OF ENERGY | Office of Science

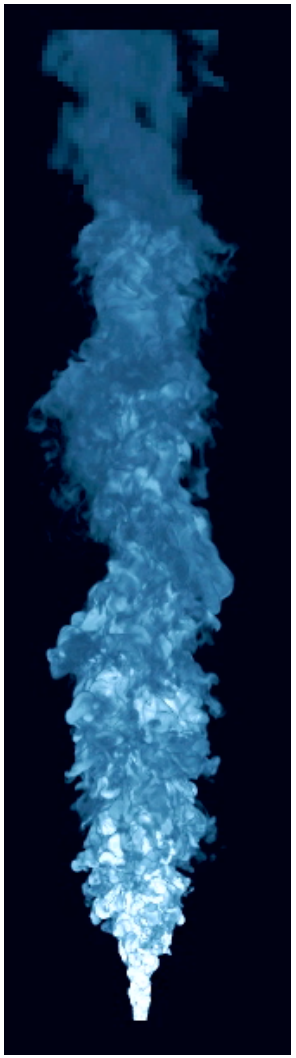
BERKELEY LAB
Lawrence Berkeley National Laboratory

Purpose

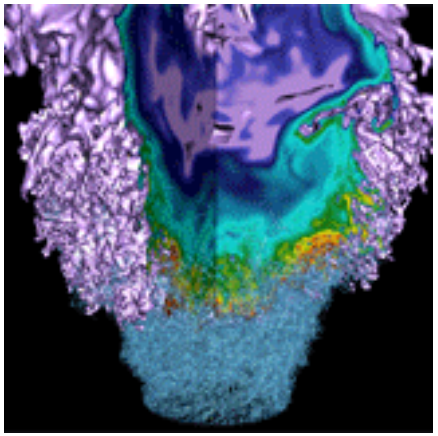
- **This presentation will help you get familiar with NERSC and its facilities**
  - Practical information
  - Introduction to how things work at NERSC
  - Help you get things done efficiently
- **This is not a programming tutorial**
  - But you will learn how to get help and what kind of help is available
  - We can give presentations on programming languages and parallel libraries – just ask
- **We will have some "hands on"**
  - Get your terminal window or SSH-enabled application ready

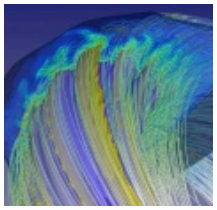
Outline

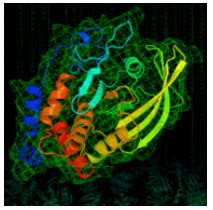
- What is NERSC?
- Computing Resources
- Storage Resources
- How to Get Help
- Accounts and Allocations
- Connecting to NERSC
- Computing Environment
- Compiling Code
- Running Jobs

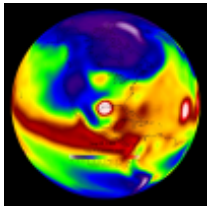


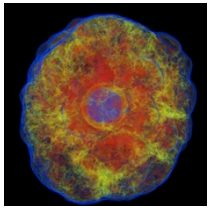
What is NERSC?

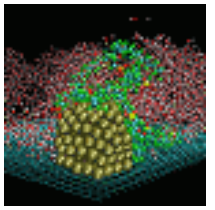












NERSC Leads DOE in Scientific Computing Productivity

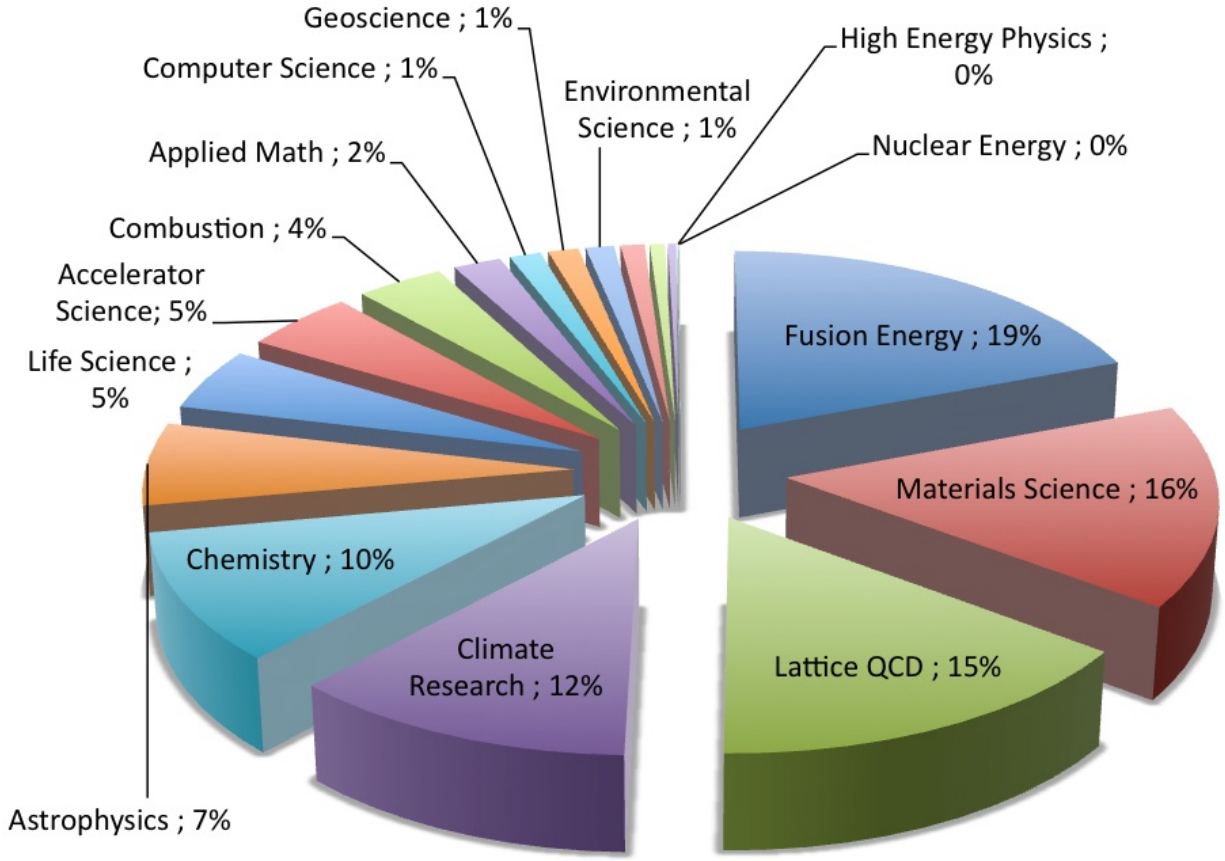
NERSC computing for science

- 5,500 users, 600 projects
- From 48 states; 65% from universities
- Hundreds of users each day
- ***1,500 publications per year***

Systems designed for science

- 1.3PF Petaflop Cray system, Hopper
  - Among Top 20 fastest in world
  - Fastest open Cray XE6 system
  - Additional clusters, data storage

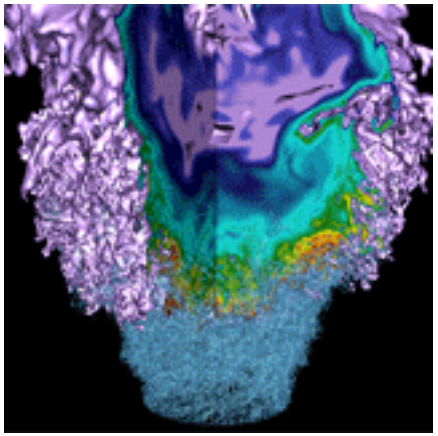
NERSC Workload

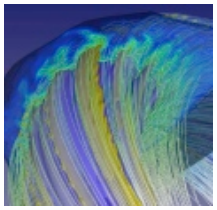


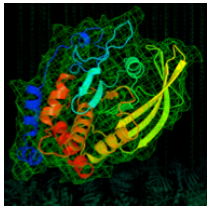
- Fusion Energy
- Materials Science
- Lattice QCD
- Climate Research
- Chemistry
- Astrophysics
- Life Science
- Accelerator Science
- Combustion
- Applied Math
- Computer Science
- Geoscience
- Environmental Science
- Nuclear Physics
- Other
- Engineering
- High Energy Physics
- Nuclear Energy
- Humanities

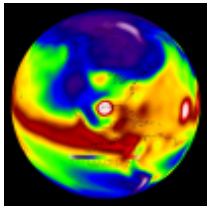
NERSC 2011 Allocations By Science Area

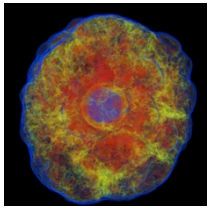
Computing Resources

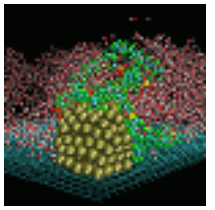












Current NERSC Systems

Large-Scale Computing Systems

Hopper (NERSC-6): Cray XE6

- 6,384 compute nodes, 153,216 cores
- 144 Tflop/s on applications; 1.3 Pflop/s peak

Edison (NERSC-7): Cray XC30 (Cascade)

- To be operational in 2013
- Over 200 Tflop/s on applications, 2 Pflop/s peak

Midrange

140 Tflops total

Carver

- IBM iDataplex cluster
- 9884 cores; 106TF

PDSF (HEP/NP)

- ~1K core cluster

GenePool (JGI)

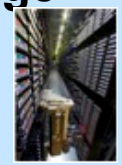
- ~5K core cluster
- 2.1 PB Isilon File System

NERSC Global Filesystem (NGF)

Uses IBM's GPFS

- 8.5 PB capacity
- 15GB/s of bandwidth

HPSS Archival Storage



- 240 PB capacity
- 5 Tape libraries
- 200 TB disk cache

Analytics & Testbeds

Dirac 48 Fermi GPU nodes

Hopper - Cray XE6

- 153,408 cores, 6,392 nodes
- Cray "Gemini" interconnect
- 2 12-core AMD 'MagnyCours' 2.1 GHz processors per node
- 24 processor cores per node
- 32 GB of memory per node (384 "fat" nodes with 64 GB)
- 216 TB of aggregate memory
- 1.2 GB memory / core (2.5 GB / core on "fat" nodes) for applications
- /scratch disk quota of 5 TB
- 2 PB of /scratch disk
- CCM (cluster compatibility) mode available
- PGI, Cray, Pathscale, GNU compilers

Use Hopper for your biggest, most computationally challenging problems.

Edison - Cray XC30 (Phase 1 / 2)

- 10K / 105K compute cores
- Cray ”Aires” interconnect
- 2 8-core Intel ’Sandy Bridge' 2.6 GHz processors per node (Phase 2 TBA)
- 16 / TBA processor cores per node
- 64 GB of memory per node
- 42 / 333 TB of aggregate memory
- Edison Phase 1 access Feb. or March 2013 / Phase 2 in Summer or Fall 2013

- 4 / TBA GB memory / core for applications
- 1.6 / 6.4 PB of /scratch disk
- CCM compatibility mode available
- Intel, Cray, GNU compilers

Use Edison for yours most computationally challenging problems.

Carver - IBM iDataPlex

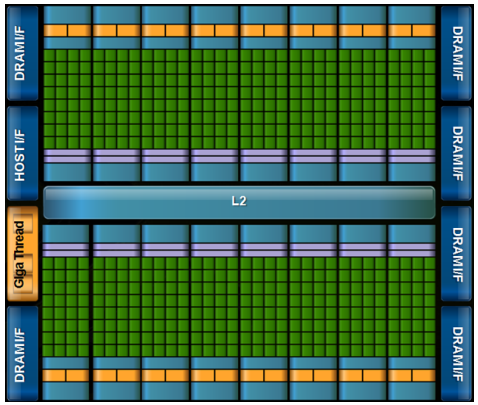
- 9,984 compute cores
- 1,202 compute nodes
- 2 quad-core Intel Nehalem 2.67 GHz processors per node + 80 Westmere nodes
- 8 processor cores per node, 12 on Westmere
- 24 GB of memory per node (48 GB on 160 "fat" Nehalem nodes + 80 Westmere nodes)
- 2.5 GB / core for applications (5.5 GB / core on "fat" nodes)
- InfiniBand 4X QDR

- NERSC global /scratch directory quota of 20 TB
- Full Linux operating system
- PGI, GNU, Intel compilers

Use Carver for jobs that use up to 512 cores, need a fast CPU, need a standard Linux configuration, or need up to 48 GB of memory on a node.

Dirac – GPU Computing Testbed

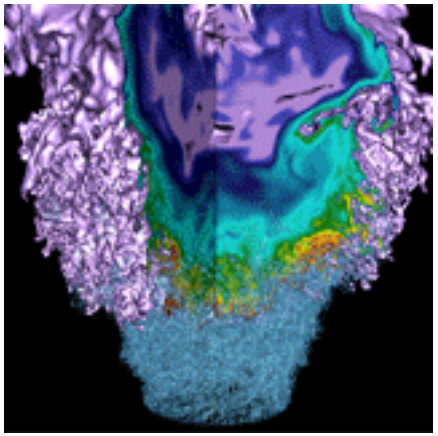
- 50 GPUs
- 50 compute nodes
  - 2 quad-core Intel Nehalem 2.67 GHz processors
  - 24 GB DRAM memory
- 44 nodes: 1 NVIDIA Tesla C2050 (Fermi) GPU with 3GB of memory and 448 cores
- 1 node: 4 NVIDIA Tesla C2050 (Fermi) GPU's, each with 3GB of memory and 448 processor cores.
- InfiniBand 4X QDR

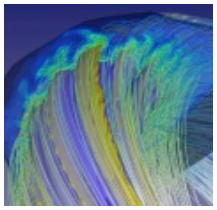


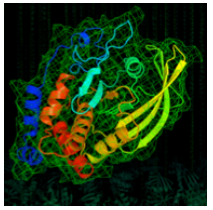
- CUDA 5.0, OpenCL, PGI and HMPP directives
- DDT CUDA-enabled debugger
- PGI, GNU, Intel compilers

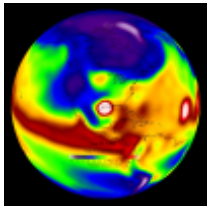
Use Dirac for developing and testing GPU codes.

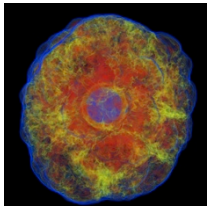
Let's Get Started!

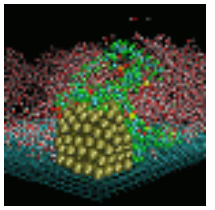












Try It Out

- **SSH to hopper.nersc.gov**
  - UNIX/Mac: ssh –Y –A username@hopper.nersc.gov
  - PC/Other: use your SSH application
- **On Hopper:**

```
% module load training
% cd $SCRATCH
% cp -rp $EXAMPLES/NewUser .
% cd NewUser
% cd flip
```

Try It Out 2

- **On Hopper:**

Fortran Compiler

```
% make
ftn  -c  flip.f90
```

C Compiler

```
cc   -c -o printit.o printit.c
ftn  -o flip  flip.o printit.o
```

Submit Job

```
% qsub flip.pbs
```

Check Status

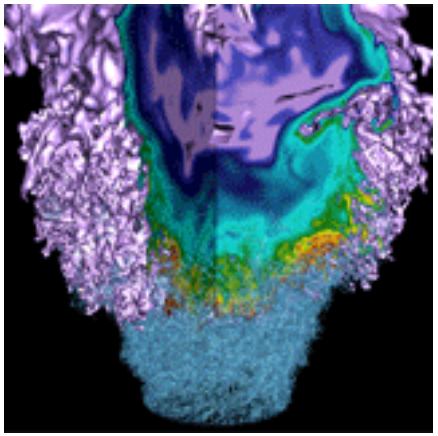
```
% qstat -u <your_user_name>
Job ID     Username Queue  Jobname       SessID NDS  TSK Memory Time  S Time
801493.sdb ragerber debug  flip            --    --   --     --  00:10 Q   --
```

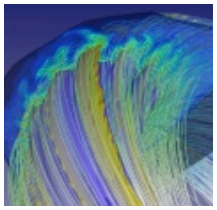
```
% ls
```

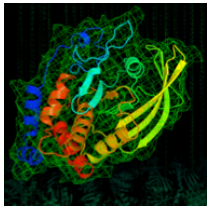
STDOUT → `flip.o.<some number>.sdb`

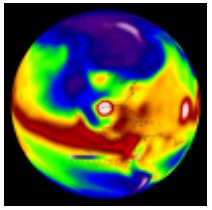
STDERR → `flip.e.<some number>.sdb`

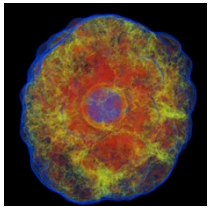
How to Get Help

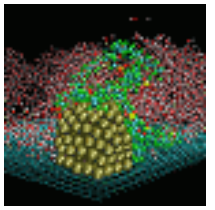












NERSC Services

- **NERSC's mission is to accelerate the pace of scientific discovery**
  - We're extremely focused on scientific results
- **User-oriented systems and services**
  - We think this is what sets NERSC apart from other centers
- **Help Desk / Consulting**
  - Immediate direct access to consulting staff that includes 7 Ph.Ds
- **User group (NUG) has tremendous influence**
  - Monthly teleconferences & yearly meetings
- **Requirement-gathering reviews (meetings in D.C.)**
  - One each for the six DOE Program Offices in the Office of Science
  - http://www.nersc.gov/science/requirements-reviews/
- **Ask, and we'll do whatever we can to fulfill your request**

How to Get Help

http://www.nersc.gov/
***For Users* section**
***My NERSC* page (link at upper right)**

1-800-666-3772 (or 1-510-486-8600)

Computer Operations* = menu option 1 (24/7)

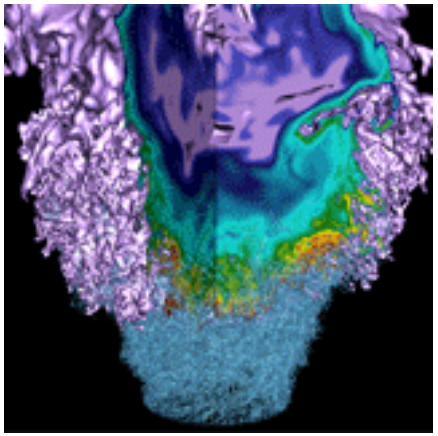
Account Support (passwords) = menu option 2, accounts@nersc.gov

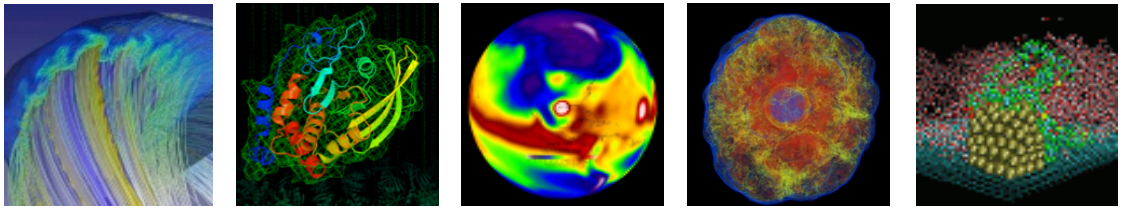
HPC Consulting = menu option 3, or consult@nersc.gov
(8-5, M-F Pacific time)

Online Help Desk = https://help.nersc.gov/

* Passwords during non-business hours

Accounts & Allocations





Accounts

There are two types of "accounts" at NERSC. It is important to differentiate between them.

1. Your personal, private account
   - Associated with your "login" or "user name"
   - Identifies you to our systems and is used when logging into NERSC systems and web services.
   - Your Project's PI or Account Manager requests an account for you.
2. An allocation account, or "repository" (aka "repo")
   - Like a bank account you use to "pay" for computer time.
   - PIs request allocations of time and/or storage
   - An individual user may belong to one or many repositories.

To apply for either type of account, see the NERSC web site at http://www.nersc.gov/.

Allocations

- **You must have an allocation of time to run jobs at NERSC (be a member of a "repo")**
- **Project PIs apply through the "ERCAP" process**
  - At https://nim.nersc.gov
- **Computer time and storage allocations are awarded by DOE**
- **Most allocations are awarded in the fall**
  - Allocation year starts in January
  - 2012: Edison time will be "free"
  - Small startup allocations are awarded throughout the year
  - Additional time available through NISE and ALCC
- **If you or your repo runs out of time, contact your PI or Account Manager. They can give you more existing time or request more for the project from DOE. (See the NERSC web site)**

Accounting Web Interface (NIM)

- Log into the NERSC NIM web site at https://nim.nersc.gov/ to manage your NERSC accounts.
- In NIM you can check your daily allocation balances, change your password, run reports, update your contact information, change your login shell, etc.

Passwords and Login Failures

Passwords

- You may call 800-66-NERSC 24/7 to reset your password
- Change it at https://nim.nersc.gov
- Answer security questions in NIM, then you can reset it yourself at

https://nim.nersc.gov

Login Failures

- 3 or more consecutive login failures on a machine will disable your ability to log in
- Send e-mail to accounts@nersc.gov or call 1-800-66-NERSC to reset your failure count

Hands-On

- **Look at the batch script you submitted**

% `cat flip.pbs`

- **Note: no charging info**
  - You have a "default repo" that is charged
  - If you have more than one repo, and you want to charge against a secondary repo, use
  - Try `getnim -U your_username`

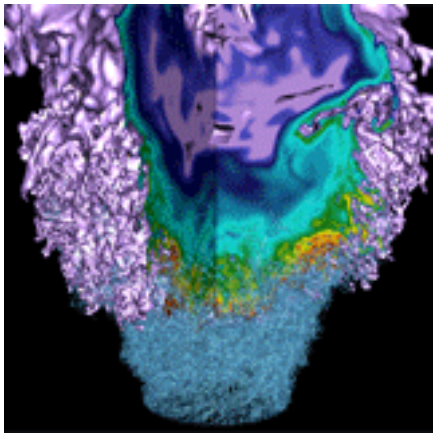
% `qsub -A repo_name flip.pbs`

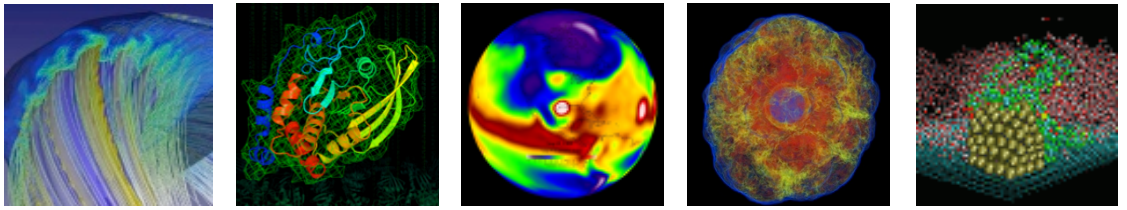
  - or add this to the batch script

```
#PBS -A repo_name
```

- **You can use batch options on qsub command line or put in the batch script**

Connecting to NERSC





- **Yushu Yao**

Hands-On

- **Check your job's status**

```
% qstat -u your_username
(you should see nothing!)
% cat flip.o.your_jobid.sdb
```

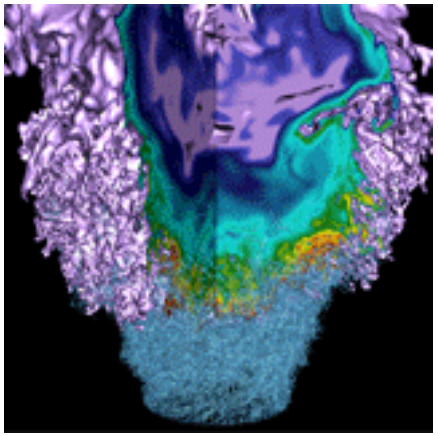
- **This is the Standout Output (STDOUT) file**
  - Text from write *, print *, printf()
  - Info from the batch system
- **Look at the batch queues**

```
% qstat -a
% qs
```

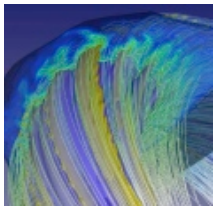
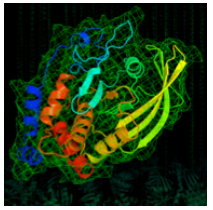
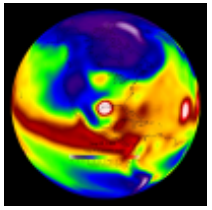
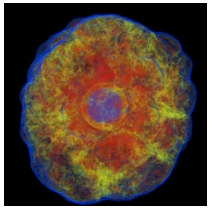
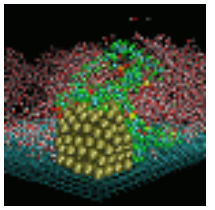
  - "qs" gives you an ordered list of queued jobs
  - Also see

https://www.nersc.gov/users/live-status/global-queue-look/

Data Resources



NERSC

Data Storage Types

- **"Spinning Disk"**
  - Interactive access
  - I/O from compute jobs
  - "Home", "Project", "Scratch", "Projectb", "Global Scratch"
  - No on-node direct-attach disk at NERSC (except PDSF & Genepool)
- **Archival Storage**
  - Permanent, long-term storage
  - Tapes, fronted by disk cache
  - "HPSS" (High Performance Storage System)

Home Directory

- When you log in you are in your "Home" directory.
- Permanent storage
- The full UNIX pathname is stored in the environment variable $HOME

```
hopper04% echo $HOME
/global/homes/r/ragerber
```

- $HOME is a global file system
  - You see all the same directories and files when you log in to any NERSC computer.
- Your quota in $HOME is 40 GB and 1M inodes (files and directories).
  - Use “myquota” command to check your usage and quota

Scratch Directories

- "Scratch" file systems are large, high-performance for "temporary storage".
- Data in $SCRATCH is purged (12 weeks from last access)
- Significant I/O from your compute jobs should be directed to $SCRATCH
- Each user has a personal directory referenced by $SCRATCH (and $SCRATCH2 on Hopper).
- Always save data you want to keep to HPSS (see below)
- $SCRATCH is local on Edison and Hopper, but Carver and other systems use a global scratch file system. ($GSCRATCH points to global scratch on Hopper and Edison)
- Data in $SCRATCH is not backed up.

Project Directories

- All NERSC systems mount the NERSC global "Project" file system. (Projectb for JGI users is similar.)
- "Project directories" are created upon request for projects (groups of researchers) to store and share data.
- The default quota in /project is 4 TB.
- While data can be written and read from a parallel job on all systems, performance ~~will~~ *may* not be as good as on $SCRATCH.
- Data in /project is not purged, but there are no automatic user backups either. Save your important data to HPSS!

File System Access Summary

File System	Hopper	Carver	Euclid	Genepool	Data Transfer Nodes	PDSF
Global homes	Y	Y	Y	Y	Y	
Global scratch	Y	Y	Y	Y	Y	
Global project	Y	Y	Y	Y	Y	Y
Global projectb	Y	Y		Y	Y	
Local scratch	Y			Y		

File System Summary

Summary of File System Policies

File System	Path	Type	Default Quota	Backups	Purge Policy
Global homes	$HOME	GPFS	40GB 1,000,000 Inodes	Yes	Not purged
Global scratch	$GSCRATCH	GPFS	20TB 2,000,000 Inodes	No	Files not accessed for 12 weeks are deleted
Global project	/project/projectdirs/*projectname*	GPFS	4TB 4,000,000 Inodes	Yes	Not purged
Hopper local scratch	$SCRATCH and $SCRATCH2	Lustre	5TB 5,000,000 Inodes (Combined)	No	Files not accessed for 12 weeks are deleted.

I/O Tips

- Use $SCRATCH for good IO performance
- Write large chunks of data (MBs or more) at a time
- Use a parallel IO library (e.g. HDF5)
- Read/write to as few files as practical from your code (try to avoid 1 file per MPI task)
- Use $HOME to compile unless you have too many source files or intermediate (*.o) files
- Do not put more than a few 1,000s of files in a single directory
- Save any and everything important to HPSS

Archival Storage (HPSS)

- For permanent, archival storage
- Permanent storage media is magnetic tape
  - 24PB data in >100M files written to 32k cartridges
  - Cartridges are loaded/unloaded into tape drives by sophisticated library robotics
- 150 TB fast-access disk cache

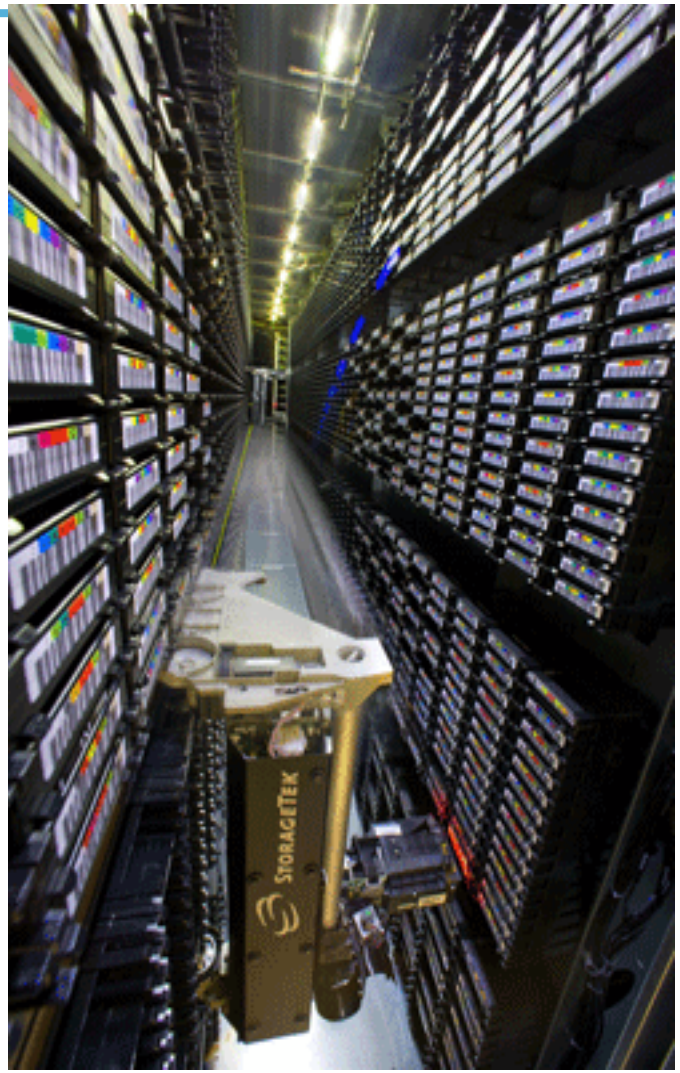
- **Hostname: archive.nersc.gov**
- **Over 24 Petabyes of data stored**
- **Data increasing by 1.7X per year**
- **120 M files stored**
- **150 TB disk cache**
- **8 STK robots**
- **44,000 tape slots**
- **44 PB maximum capacity today**
- **Average data xfer rate: 100 MB/sec**

HPSS Authentication

- **NERSC storage uses a token-based authentication method**
  - User places encrypted authentication token in ~/.netrc file at the top level of the home directory on the compute platform
- **Authentication tokens can be generated in 2 ways:**
  - Automatic – NERSC auth service:
    - Log into any NERSC compute platform; Type "hsi"; Enter NERSC password
  - Manual – https://nim.nersc.gov/ website
    - Under "Actions" dropdown, select "Generate HPSS Token"; Copy/paste content into ~/.netrc; chmod 600 ~/.netrc
- **Tokens are username and IP specific—must use NIM to generate a different token for use offsite**

HPSS Clients

- **Parallel, threaded, high performance:**
  - HSI
    - Unix shell-like interface
  - HTAR
    - Like Unix tar, for aggregation of small files
  - PFTP
    - Parallel FTP
- **Non-parallel:**
  - FTP
    - Ubiquitous, many free scripting utilities
- **GridFTP interface (garchive)**
  - Connect to other grid-enabled storage systems



Archive Technologies, Continued…

- **HPSS clients can emulate file system qualities**
  - FTP-like interfaces can be deceiving: the archive is backed by tape, robotics, and a single SQL database instance for metadata
  - Operations that would be slow on a file system, e.g. lots of random IO, can be impractical on the archive
  - It's important to know how to store and retrieve data efficiently. (See http://www.nersc.gov/users/training/nersc-training-events/data-transfer-and-archiving/)

- **HPSS does not stop you from making mistakes**
  - It is possible to store data in such a way as to make it difficult to retrieve
  - The archive has no batch system. Inefficient use affects others.

Hands-On

- Use htar to save a directory tree to HPSS

```
%cd $SCRATCH  (or where you put the NewUser directory)
%htar -cvf NewUser.tar NewUser
```

- See if the command worked

```
%hsi ls -l NewUser.tar
-rw-r-----   1 ragerber  ccc   6232064 Sep 12 16:46 NewUser.tar
```

- Retreive the files

```
%cd $SCRATCH2
%htar -xvf NewUser.tar
```

- Retreive the tar file

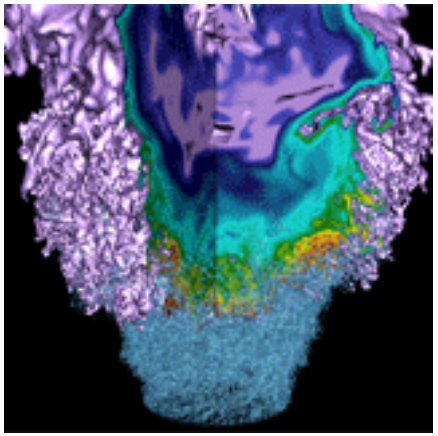
```
%hsi get NewUser.tar
```

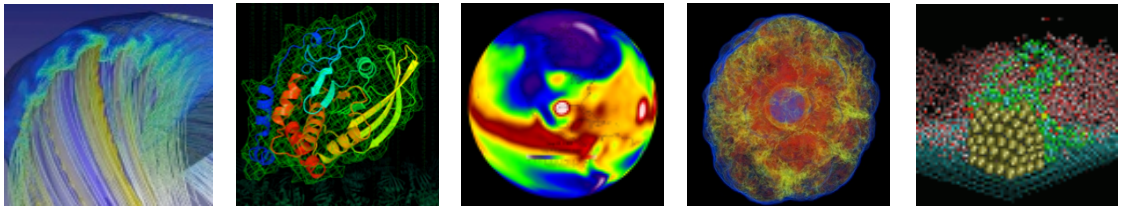
Data Transfer and Archiving

- **A NERSC Training Event**
- **Good information:**

http://www.nersc.gov/users/training/events/data-transfer-and-archiving/

Computing Environment





Shell Initialization Files

- NERSC installs dot-files in your home directory (e.g. .login, .profile)
  - Commands in dot-files are executed when you log in (or start a new shell)
  - Do not modify these or your jobs and compiles will not work correctly.
- Each dot-file sources an additional file with the same name, but with an .ext extension.
  - Put your local modifications in these .ext files (e.g. .login.ext, .profile.ext)

Modules

- Easy access to NERSC's extensive software collection is controlled by the modules utility.
- With modules, you manipulate your computing environment to use applications and programming libraries.
- In many cases, you can ignore modules because NERSC has already loaded a rich set of modules for you when you first log in.
- If you want to change that environment you "load," "unload," and "swap" modules.
- A small set of module commands can do most of what you'll want to do.

module list

- Shows you your currently loaded modules.
- When you first log in, you have a number of modules loaded for you. Here is an example from Hopper.

```
hopper03% module list
Currently Loaded Modulefiles:
  1) modules/3.2.6.6                          11) xpmem/0.1-2.0301.25333.20.2.gem
  2) xtpe-network-gemini                      12) xe-sysroot/3.1.61
  3) pgi/10.9.0                               13) xt-asyncpe/4.9
  4) xt-libsci/10.5.01                        14) atp/1.1.2
  5) xt-mpich2/5.2.1                          15) PrgEnv-pgi/3.1.61
  6) udreg/2.2-1.0301.2966.16.2.gem           16) eswrap/1.0.8
  7) ugni/2.1-1.0301.2967.10.23.gem           17) xtpe-mc12
  8) pmi/2.1.1-1.0000.8296.10.8.gem           18) xt-shmem/5.2.1
  9) dmapp/3.0-1.0301.2968.22.24.gem          19) torque/2.4.8-snap.201004261413
 10) gni-headers/2.1-1.0301.2931.19.1.gem     20) moab/5.3.6-s14846
```

- The most important module is called "PrgEnv-pgi", which lets you know that the environment is set up to use the Portland Group compiler suite.

module avail

- The "module avail" command will list all the available modules. It's a very long list, so I won't list it here
- You can use the module's name stem to do a useful search

```
nid00163% module avail PrgEnv

PrgEnv-cray/4.0.46(default)   PrgEnv-intel/4.0.46(default)
PrgEnv-gnu/4.0.46(default)    PrgEnv-pgi/4.0.46(default)
(abbreviated list)
```

- Here you see that four programming environments are available using the Cray, GNU, intel, and PGI compilers.
- The word "default" is confusing here; it does not refer to the default computing environment, but rather the default version of each specific PrgEnv module. The default on Hopper is PrgEnv-pgi.

module swap

Let's say you want to use the Cray compiler instead of PGI.

```
%module swap PrgEnv-pgi PrgEnv-cray
```

Now you are using the Cray compiler suite. That's all you have to do.

You don't have to change your makefiles, or anything else in your build script unless they contain PGI or Cray-specific options or features.

Hands-On

```
% module list
% ftn -V
pgf90 12.5-0 64-bit target on x86-64 Linux -tp istanbul
% module swap PrgEnv-pgi PrgEnv-cray
% ftn -V
Cray Fortran : Version 8.0.7  Wed Jan 16, 2013  15:11:32
```

- The “ftn” command invokes the Fortran compiler associated with the current PrgEnv

module load

- There is plenty of software that is not loaded by default.
- You can consult the NERSC web pages to see a list, or you can use the "module avail" command to see what modules are available
- For example, if you want to use the NAMD molecular dynamics application. Try "module avail namd".

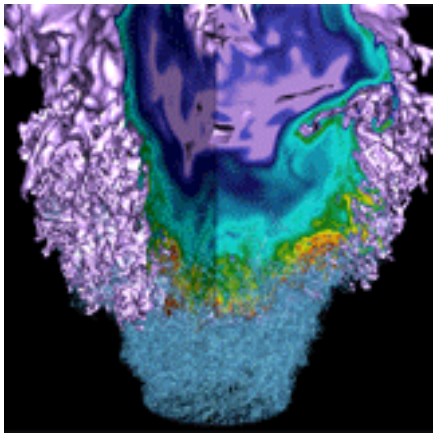
```
nid00163% module avail namd
namd/2.7            namd/2.8b1            namd_ccm/2.8 (default)
namd/2.8            namd/2.9 (default)    namd_ccm/2.9
```

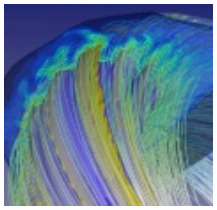
- The default version is 2.9, but say you'd rather use some features available only in version 2.8. In that case, just load that module.

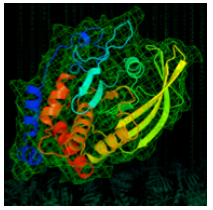
```
nid00163% module load namd/2.8
```

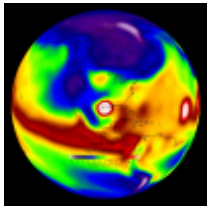
- The “namd2” binary for version 2.8 is now in your UNIX search path.

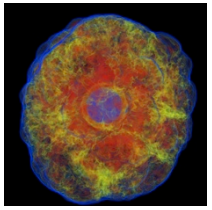
Building Codes

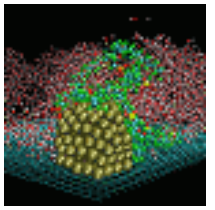












Invoking the Compilers

- **Let's assume that you're compiling**
  - a parallel application
  - using MPI and the code is
  - written in Fortran, C, or C++
- **Then compiling is easy**
  - You will use standard compiler wrapper
  - All the include file and library paths are set
  - Linker options are set

Parallel Compilers

Platform	Fortran	C	C++
Cray	ftn	cc	CC
Others	mpif90	mpicc	mpiCC

```
!Filename hello.f90
program hello

      implicit none
      include "mpif.h"

      integer:: myRank
      integer:: ierror

      call mpi_init(ierror)

      call mpi_comm_rank(MPI_COMM_WORLD,myRank)

      print *, "MPI Rank ",myRank," checking in!"

      call mpi_finalize(ierror)
```

```
% ftn –o hello.x hello.f90
```

That's it!

No `-I/path/to/mpi/include` or `-L/path/to/mpi/lib`

It's all taken care of for you.

"Serial" compilers

- **You can use serial compilers as you would on a typical Linux cluster**
  - gcc, gfortran, pgf90, etc.
  - Binary won't run on compute nodes on Crays (except in CCM mode)
  - You need to supply all the compiler and linker options
  - May have to load a module to access a given compiler (e.g. module load pgi/12.9.0)

Using Programming Libraries (Cray)

All you have to do is load the appropriate module and compile.

Let's compile an example code that uses the HDF5 I/O library.
First let's try it in the default environment.

```
nid00195% cc -o hd_copy.x hd_copy.c
INFO: linux target is being used
Can't find include file hdf5.h (hd_copy.c: 39)
```

The compiler doesn't know where to find the include file.
Now let's load the hdf5 module and try again.

```
nid00195% module load hdf5
nid00195% cc -o hd_copy.x hd_copy.c
```

We're all done and ready to run the program! No need to manually add the path to HDF5; it's all taken care of by the compiler wrapper scripts.

Using Programming Libraries (non-Cray)

Even with the module loaded, the compiler doesn't know where to find the HDF5 files. Consult our web pages for details.

```
% mpicc -o hd_copy.x hd_copy.c
  Can't find file hdf5.h (hd_copy.c: 39)
  PGC/x86-64 10.8-0: compilation aborted
% module load hdf5
% mpicc -o hd_copy.x hd_copy.c
 Can't find file hdf5.h (hd_copy.c: 39)
 PGC/x86-64 10.8-0: compilation aborted
```

Using Programming Libraries (non-Cray)

We have to use environment variables defined in the module (use "module show" to see them).

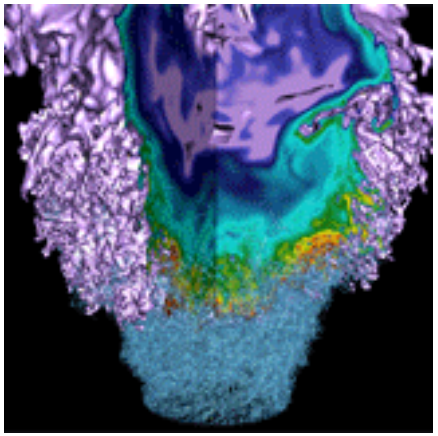
```
% mpicc -o hd_copy.x hd_copy.c $HDF5
% module show hdf5
-------------------------------------------------------------------
/usr/common/usg/Modules/modulefiles/hdf5/1.8.8:

/opt/cray/modulefiles/hdf5/1.8.8:

conflict  hdf5-parallel
conflict  PrgEnv-pathscale
setenv          CRAY_HDF5 74
setenv          INTEL_HDF5 120
setenv          PGI_HDF5 119
prepend-path  PATH /opt/cray/hdf5/1.8.8/bin
setenv          CRAY_HDF5_DIR /opt/cray/hdf5/1.8.8
setenv          CRAY_HDF5_VERSION 1.8.8
prepend-path  PE_PRODUCT_LIST CRAY_HDF5
setenv          HDF5_DIR /opt/cray/hdf5/1.8.8/pgi/119
setenv          HDF5_INCLUDE_OPTS /opt/cray/hdf5/1.8.8/pgi/119/
include
prepend-path  CRAY_LD_LIBRARY_PATH /opt/cray/hdf5/1.8.8/pgi/119/lib
```

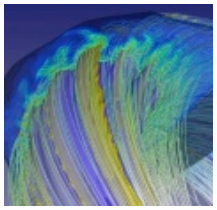
Hands On

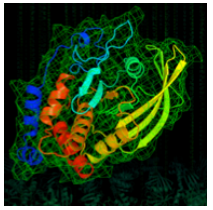
You can try some example compiles and run some jobs by following

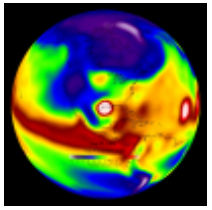
https://www.nersc.gov/users/training/nersc-training-events/getting-started/hands-on/

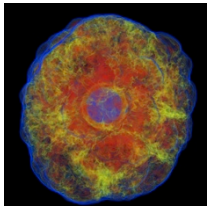
Running Jobs

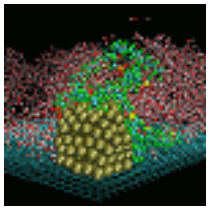












Jobs at NERSC

- **Most jobs are parallel, using 10s to 100,000+ cores**
- **Production runs execute in batch mode**
- **Interactive and debug jobs are supported for up to 30 minutes**
- **Typically run times are a few to 10s of hours.**
  - Each machine has different limits.
  - Limits are necessary because of MTBF and the need to accommodate 5,500 users' jobs
- **Many jobs "package" lower concurrency runs into one job**
  - Can increase throughput
  - Even many "serial jobs"
  - Load balance may be an issue

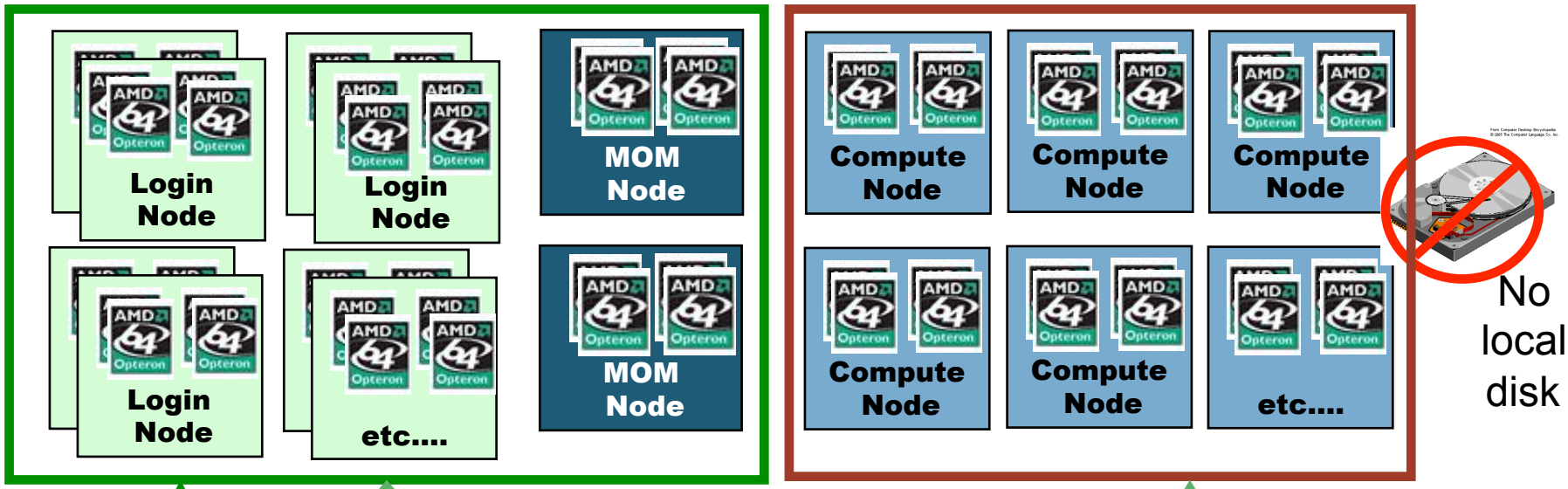
Login Nodes and Compute Nodes

- Each supercomputer has 3 types of nodes that you will use directly
  - Login nodes
  - Compute nodes
  - Job launcher or "MOM" nodes
- Login nodes
  - Edit files, compile codes, run UNIX commands
  - Submit batch jobs
  - Run short, small utilities and applications
- Compute nodes
  - Execute your application; dedicated to your job
  - No direct login access
- Job launcher or "MOM" nodes
  - Execute your batch script commands
  - Carver: "head" compute node; Cray: shared "service" node; not a compute node

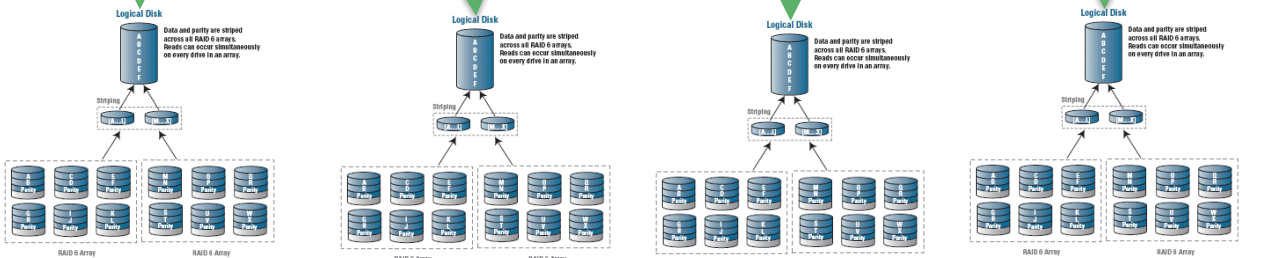
Cray Systems

Full Linux OS – Shared Access

CNL (no logins) – Dedicated



HPSS



home project projectb gscratch scratch

U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB

Carver / Dirac

Full Linux OS – Shared

Login Node

Login Node

Login Node

etc....

Full Linux (no logins) – Dedicated

MOM & Compute

Compute Node

Compute Node

Compute Node

MOM & Compute

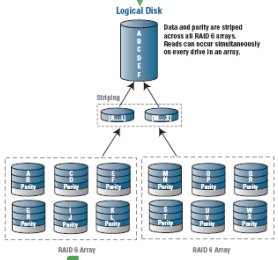
Compute Node

Compute Node

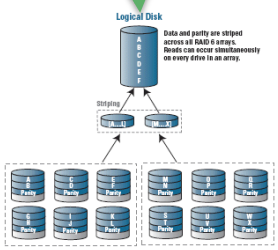
etc....

No local disk

HPSS

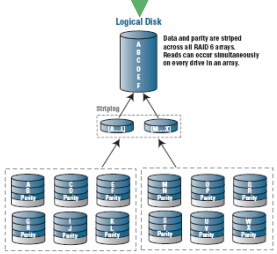


home



project

projectb



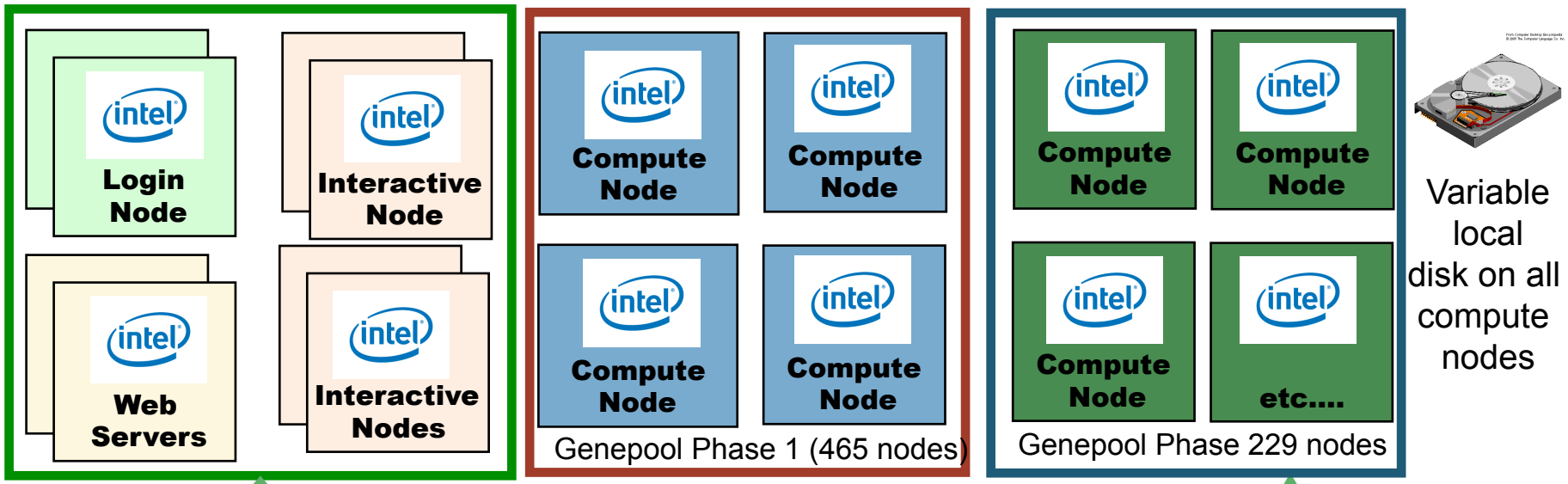
gscratch

U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB

Genepool

Full Linux OS – Shared Accessed using SGE – Full Linux OS



Launching Parallel Jobs

- A "job launcher" executes your code
  - Distributes your executables to all your nodes
  - Starts concurrent execution of N instances of your program
  - Manages execution of your application
  - On Crays: the job launcher is called "aprun"
  - On Carver: "mpirun"
- Only the job launcher can start your job on compute nodes
- You can't run the job launcher from login nodes

Submitting Batch Jobs

- To run a job on the compute nodes you must write a "batch script" that contains
  - Batch directives to allow the system to schedule your job
  - An `aprun` or `mpirun` command that launches your parallel executable
- Submit the job to the queuing system with the `qsub` command
  - `%qsub my_batch_script`

Sample Hopper Batch Script

```
#PBS -q debug
#PBS -l mppwidth=96
#PBS -l walltime=00:10:00
#PBS -N my_job
#PBS -V

cd $PBS_O_WORKDIR
aprun -n 96./my_executable
```

The #PBS directives required for each system are different, so consult the NERSC web site for details.

Monitoring Your Job

- Once your job is submitted, it will start when resources are available
- Monitor it with
  - qstat –a
  - qstat –u username
  - showq
  - qs
  - NERSC web site “Queue Look”

`https://www.nersc.gov/users/live-status/global-queue-look/`

Job Limits

There are per user, per machine job limits. See the NERSC web site for details. Here are the limits on Hopper as of Jan 16, 2013.

Specify these queues (with –q queue_name)

NEVER these!

Submit Queue	Execution Queue[1]	Nodes	Processors	Max Wallclock	Relative Priority	Run Limit[2]	Queued Limit[3]	Queue Charge Factor
interactive	interactive	1-256	1-6,144	30 mins	2	1	1	1
debug	debug	1-512	1-12,288	30 mins	3	1	1	1
regular	reg_1hour	1-256	1-6,144	1 hr	5	8	8	1
	reg_short	1-682	1-16,368	6 hrs	5	16	16	1
	reg_small	1-682	1-16,368	48 hrs	5	16	16	1
	reg_med	683-2,048	16,369-49,152	36 hrs	4	4	4	0.6
	reg_big	2,049-4,096	49,153-98,304	36 hrs	4	2	2	0.6
	reg_xbig[4]	4,097-6,100	98,305-146,400	12 hrs	1	2	2	0.6
--	bigmem[5]	1-384	1-9,216	24 hrs	5	1	1	1
low	low	1-683	1-16,392	24 hrs	7	6	6	0.5
thruput[6]	thruput	1-2	1-48	168 hrs	5	250	500	1.0
scavenger[7]	scavenger	1-683	1-16,392	6 hrs	8	2	2	0
premium	premium	1-2,048	1-49,152	12 hrs	3	1	1	2
ccm_queue[8]	ccm_queue	1-682	1-16,368	96 hrs	4	16	16	1
ccm_int[9]	ccm_int	1-512	1-12,288	30 mins	3	1	1	1
iotask[9]	iotask	1-682	1-16,368	12 hrs	6	1	-	1
xfer[10]	xfer	--	--	12 hrs	--	4	3	0

Interactive Parallel Jobs

- You can run small parallel jobs interactively for up to 30 minutes

```
% qsub –I –V –lmppwidth=32
[wait for job to start]
% cd $PBS_O_WORKDIR
% aprun –n 32 ./mycode.x
```

How Your Jobs Are Charged

- **Your repository account is charged for each core your job was allocated for the entire duration of your job.**
  - The minimum allocatable unit is a node. Hopper has 24 cores/node, so your minimum charge on Hopper is 24*`walltime`.
  - e.g., `mppwidth=96` for 1 hour of run time is charged 96*1 hour = 96 MPP Hours (assuming the default setting of `mppnppn=24`)
  - You are charged for your actual run time, not the value of `walltime` in your batch script.
  - Serial jobs on Carver are charged for just a single core.
- **If you have access to multiple repos, pick which one to charge in your batch script**
  - `#PBS -A repo_name`

Charge Factors & Discounts

- **Each machine has a "machine charge factor" (mcf) that multiplies the "raw hours" used**
  - Hopper has mcf=1.0
  - Carver has mcf=1.5
- **Queues have "priority charge factors" (pcf) and corresponding relative scheduling priorities**
  - Premium pcf=2.0
  - Low pcf=0.5
  - Everything else pcf=1.0
- **On Hopper only:**
  - reg_med, reg_big, reg_xbig jobs get a 40% discount
- **Storage and bandwidth are allocated and charged for HPSS**
  - Exhausting an HPSS allocation is rare
  - See the NERSC web site for details


NERSC


U.S. DEPARTMENT OF
ENERGY
Office of
Science


BERKELEY LAB
Lawrence Berkeley National Laboratory

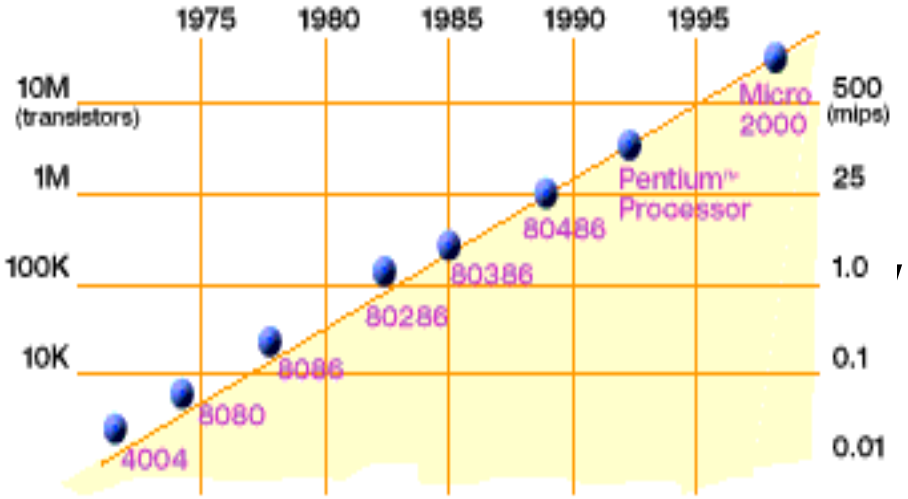
Aside: Why Do You Care About Parallelism?

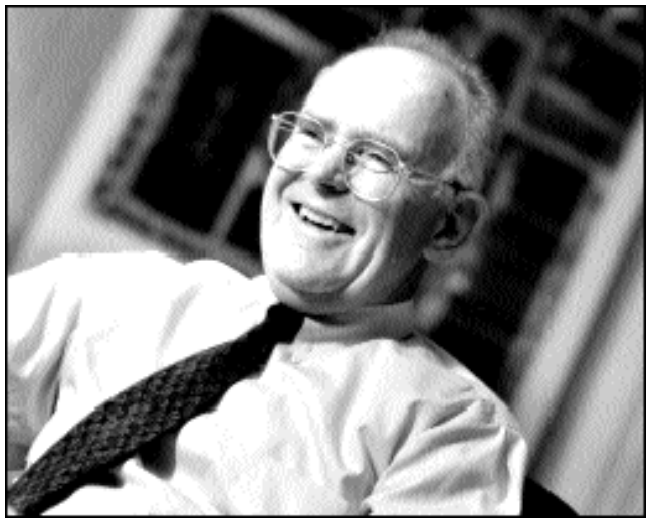
U.S. DEPARTMENT OF ENERGY | Office of Science

NERSC National Energy Research Scientific Computing Center

BERKELEY LAB Lawrence Berkeley National Laboratory

Moore's Law





2X transistors/Chip Every 1.5 years

Called "Moore's Law"

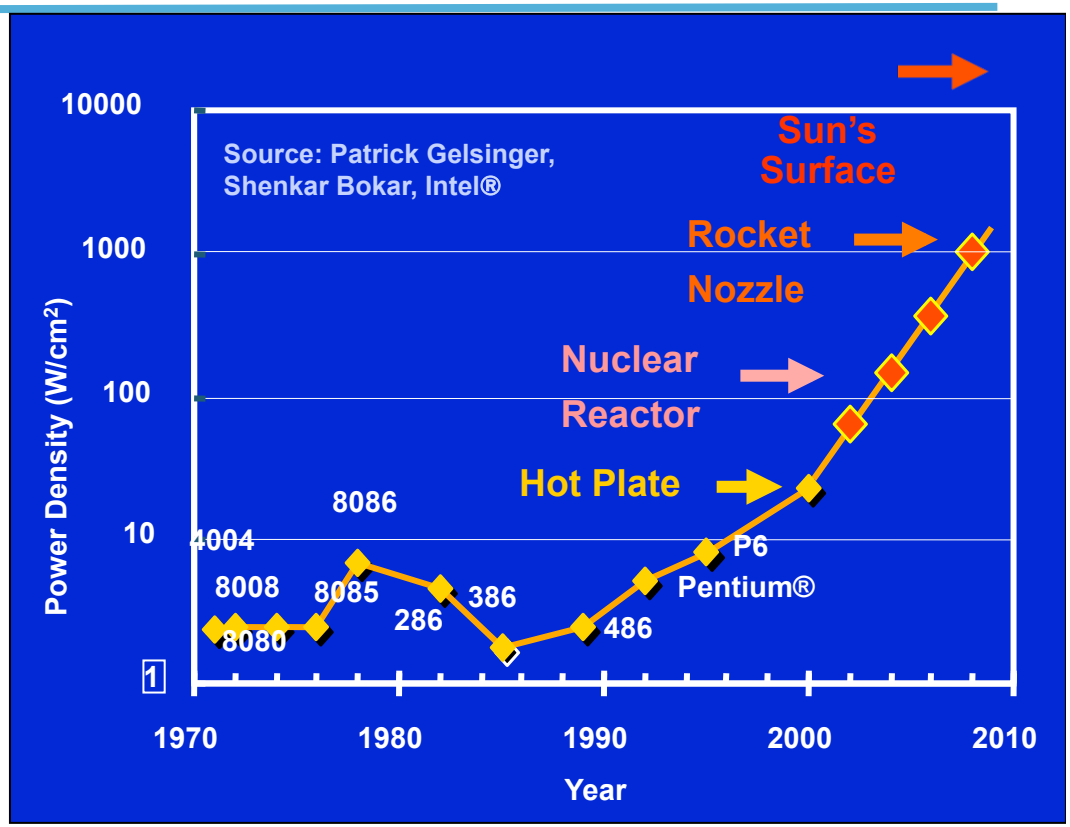
Microprocessors have become smaller, denser, and more powerful.

Gordon Moore (co-founder of Intel) predicted in 1965 that the transistor density of semiconductor chips would double roughly every 18 months.

Slide source: Jack Dongarra

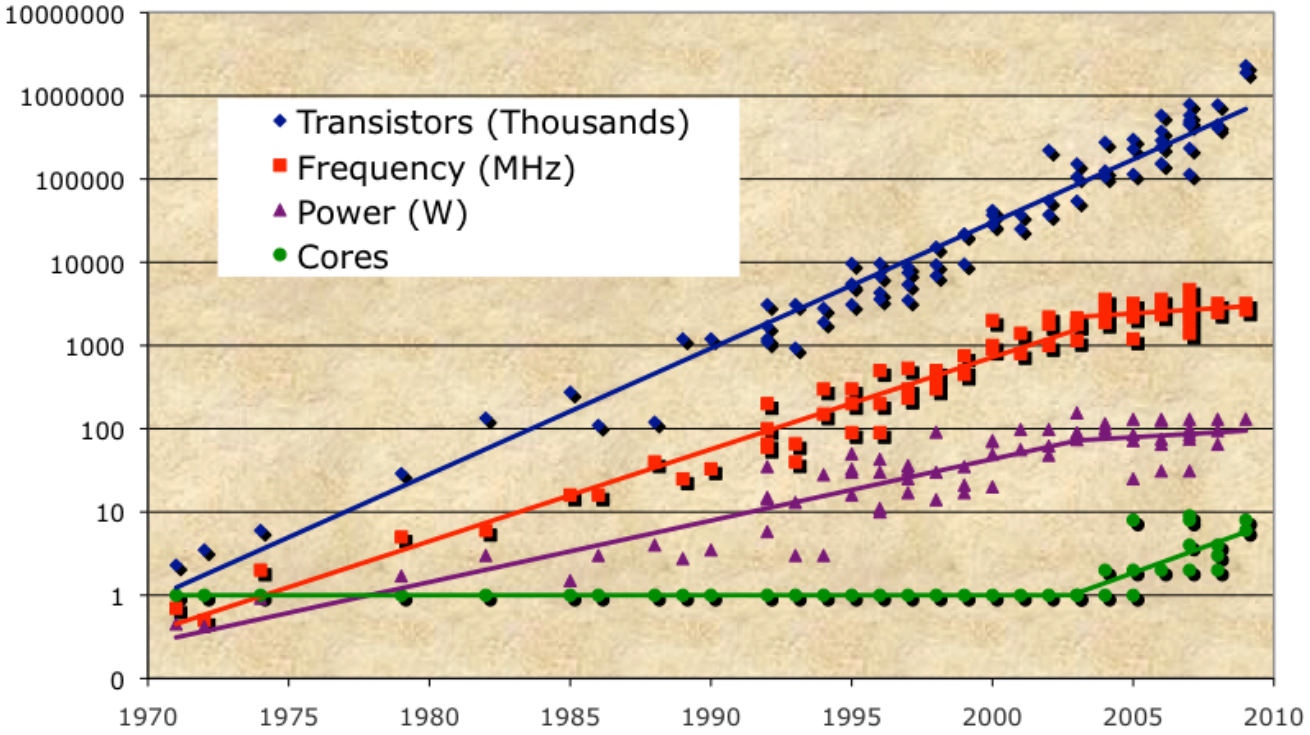
Power Density Limits Serial Performance

- Concurrent systems are more power efficient
  - Dynamic power is proportional to V^2fC
  - Increasing frequency (f) also increases supply voltage (V) → cubic effect
  - Increasing cores increases capacitance (C) but only linearly
  - Save power by lowering clock speed



- High performance serial processors waste power
  - Speculation, dynamic dependence checking, etc. burn power
  - Implicit parallelism discovery
- More transistors, but not faster serial processors

Revolution in Processors



- Chip density is continuing increase ~2x every 2 years
- Clock speed is not
- Number of processor cores may double instead
- Power is under control, no longer growing

Moore's Law reinterpreted

- Number of cores per chip will double every two years
- Clock speed will not increase (possibly decrease)
- Need to deal with systems with millions of concurrent threads
- Need to deal with inter-chip parallelism as well as intra-chip parallelism
- Your take-away:
  - *Future performance increases in computing are going to come from exploiting parallelism in applications*