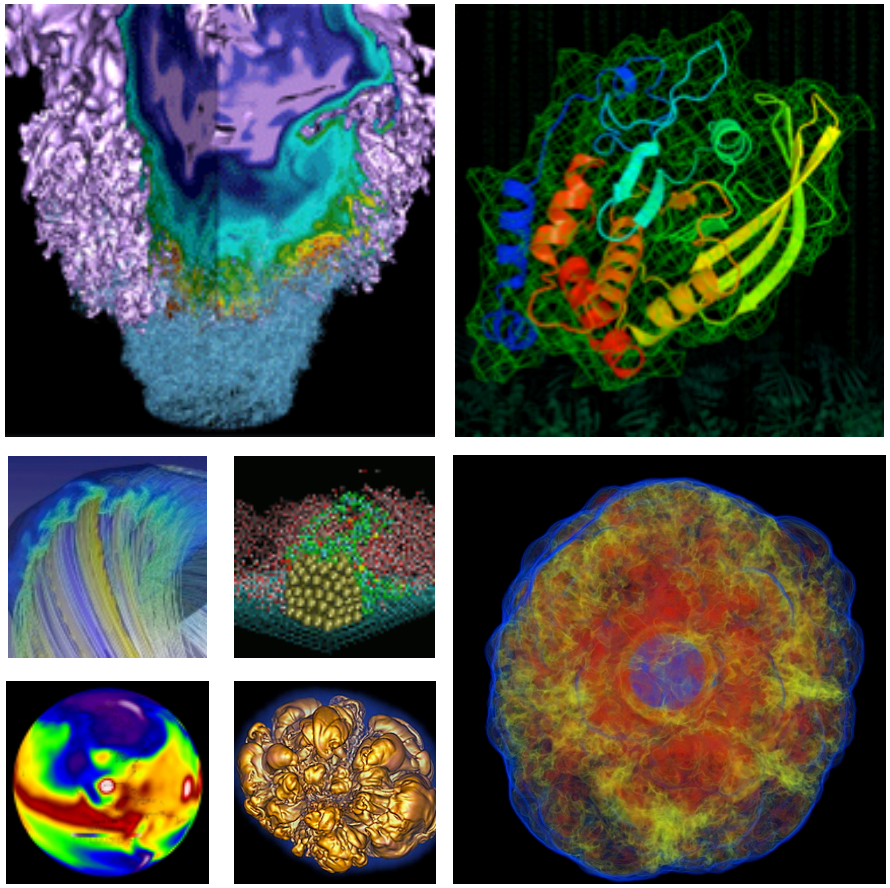
Best Practices for Best Performance on Edison



Zhenjgi Zhao
NERSC User Services Group

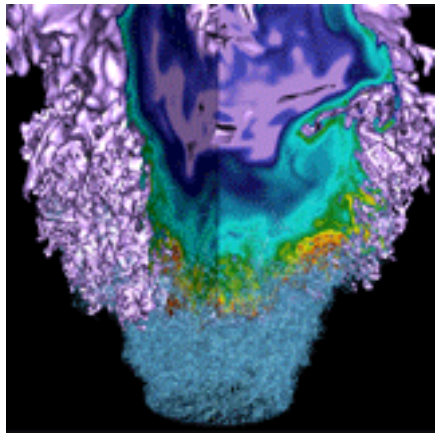
NUG 2014
Feb 6, 2014

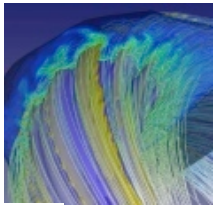
Agenda

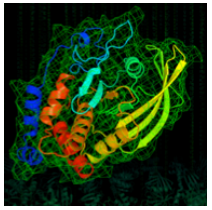
- System Overview
- Compile time optimization
- Run time tuning options
- Node placements on Edison
- A couple of tips for Lustre I/O
- Python applications at scale
- CCM usage

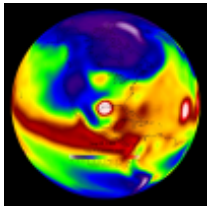
- Will not cover libraries, profiling tools.
- Interconnect will be covered by next talk

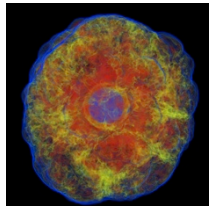
System Overview

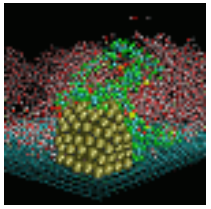












Edison at Glance

- First Cray XC30
- Intel Ivy Bridge 12-core, 2.4GHz processors
- 64GB memory node
- Aries interconnect with Dragonfly topology for great scalability
- Software environment similar to Hopper
- Performs 2.2x sustained performance relative to Hopper
- 3 Lustre scratch file systems configured as 2:2:3 for capacity and bandwidth
- Access to NERSC's GPFS global file system via DVS
- 12 x 512GB login nodes to support visualization and analytics
- Ambient cooled for extreme energy efficiency

Vital Statistics

	Hopper	Edison
Cabinets	68	28
Compute Nodes	6,384	5,192
CPU Cores (*Total / Per-node*)	152,408 / 24	124,608 / 24
CPU Frequency (GHz)	2.1	2.4
Peak Flops (PF)	1.29	2.4
Memory (TB) (*Total / Per-node*)	217 / 32	333 / 64
Memory (Stream) BW* (TB/s)	331	462.8
Memory BW/node* (GB/s)	52	89
File system(s)	2 PB @ 70 GB/s	7.56 PB @ 180 GB/s
Peak Bisection BW (TB/s)	5.1	11
Power (MW Linpack)	2.9	1.9

Baseline performance

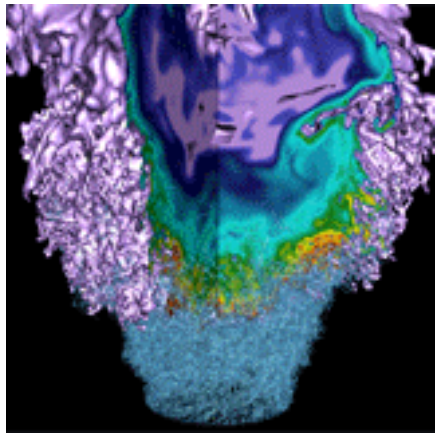
NERSC-6 Application Benchmarks

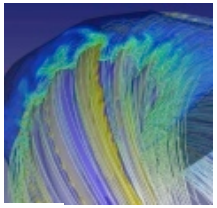
Application	CAM	GAMESS	GTC	IMPACT-T	MAESTRO	MILC	PARATEC
Concurrency	240	1024	2048	1024	2048	8192	1024
Streams/Core	2	2	2	2	1	1	1
Edison Time (s)	273.08	1,125.80	863.88	579.78	935.45	446.36	173.51
Hopper Time(s)	348	1389	1338	618	1901	921	353
Speedup[1)]	**1.3**	**1.2**	**1.5**	**1.1**	**2.0**	**2.1**	**2.0**

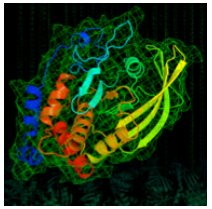
	SSP[2)]
Edison	**258**
Hopper	**144**

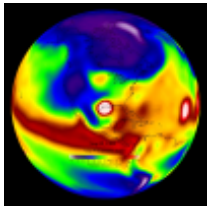
[1)] Speedup=Time(Hopper)/Time(Edison)
[2)] SSP stands for sustained system performance

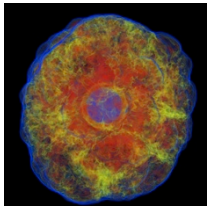
Compile time options

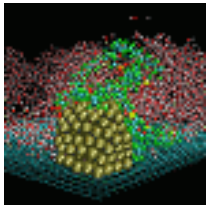












Compilers and NERSC recommended compiler optimization flags

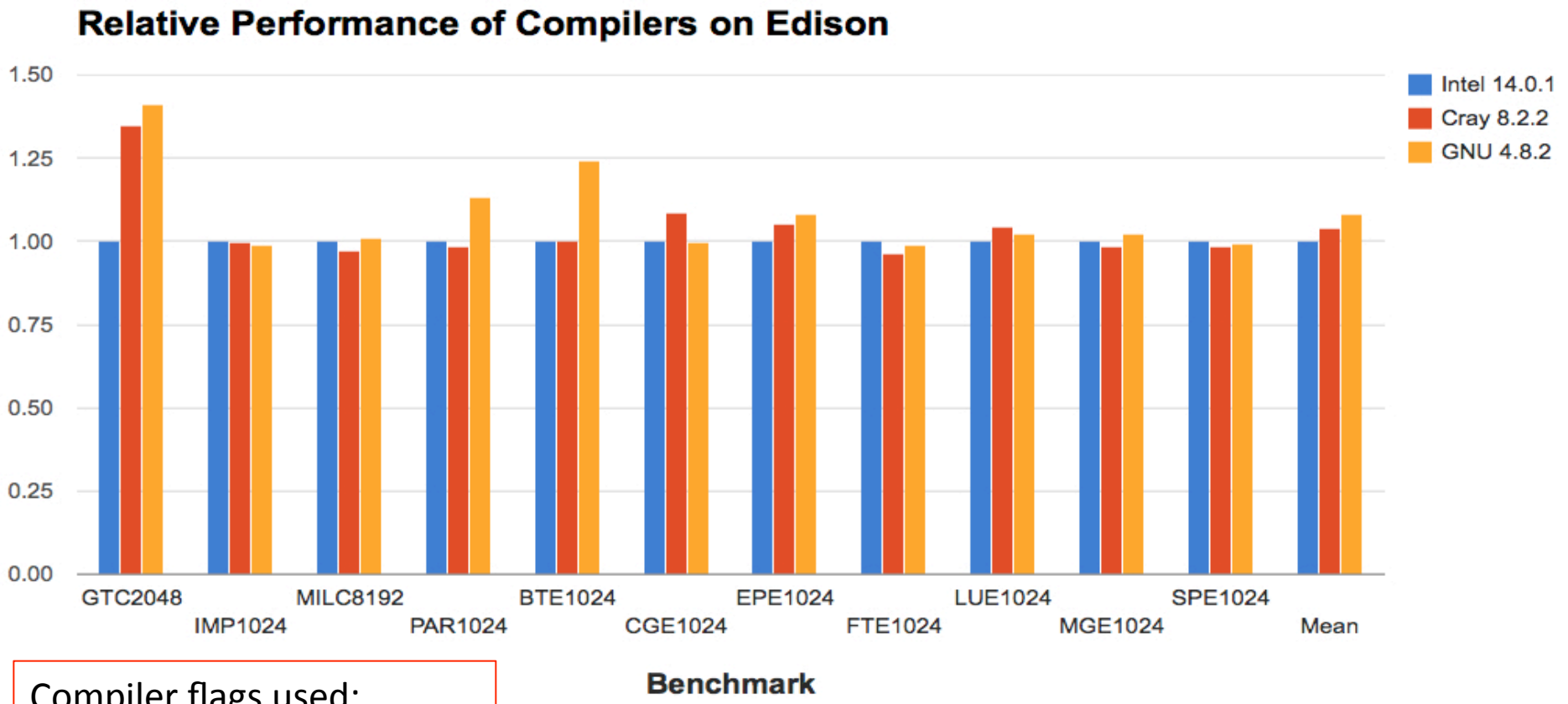
Compiler	Recommended Optimization flags	Default
Intel (default)	-fast –no-ipo	Comparable to the -O2 optimization level; Compiler wrappers add -xAVX
Cray	Default	High optimization; Compiler wrappers add: -hcpu=ivybridge
GNU	-Ofast	No optimization; Compiler wrappers add: -march=core-avx-i

Use the verbose option of the compiler wrappers to see the exact compile/link options

ftn –v hello.f90
cc –v hello.c
CC –v hello.C

Module show craype-ivybridge

Compilers and NERSC recommended compiler optimization flags



Compiler flags used:

Intel: **-fast –no-ipo**

Cray: **default**

GNU: **-Ofast**

Courtesy of Mike Stewart at NERSC

Users are responsible to validate the codes to generate correct results

- **Intel compiler fails to catch the type mismatch in the example below, and generates wrong results.**
- **Cray and GNU compilers do a better job by aborting the compilation**

```
zz217@edison02:~> cat test1.f90
program test1
real y
y=1.0
print *, "y, a(y) = ", y, a(y)
end

real*8 function a(z)
real*8 z
a=z
end
```

```
zz217@edison02:~> ftn test1.f90
zz217@edison02:~> ./a.out
 y, a(y) =    1.000000      0.0000000E+00
```

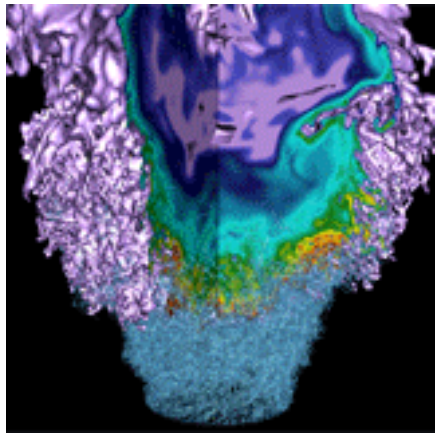
- **Strictly follow language standards in your coding is highly recommended as compilers follow the language standard more strictly now.**

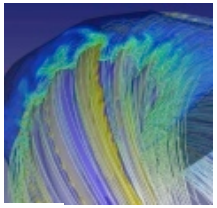
Compiler flags that are helpful to generate useful warnings

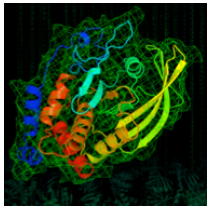
```
zz217@edison02:~> ftn -warn all test1.f90
test1.f90(6): warning #6717: This name has not
been given an explicit type.   [A]
y=a(x)
--^
test1.f90(6): warning #6717: This name has not
been given an explicit type.   [A]
y=a(x)
^
test1.f90(6): error #7977: The type of the function
reference does not match the type of the function
definition.   [A]
y=a(x)
--^
test1.f90(6): error #6633: The type of the actual
argument differs from the type of the dummy
argument.   [X]
y=a(x)
----^
compilation aborted for test1.f90 (code 1)
```

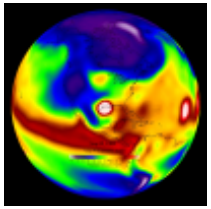
Intel	-warn all
Cray	-m msg_lvl
GNU	-Wall

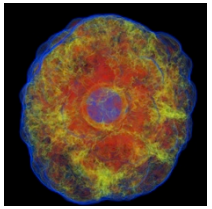
Rum time options

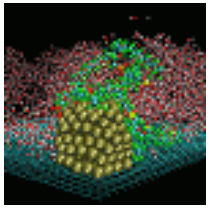




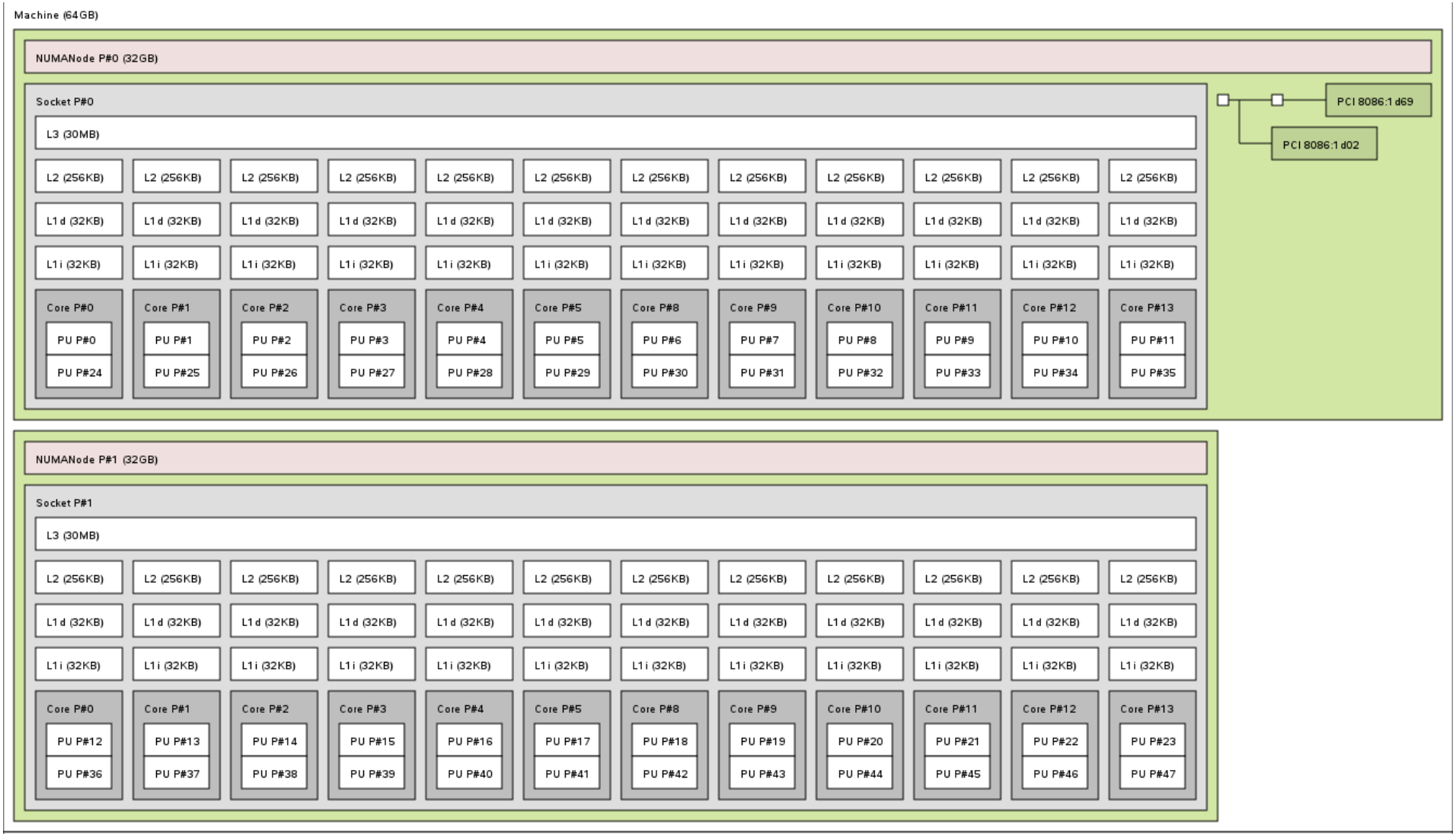








Edison compute node



Host: nid02803
Indexes: physical
Date: Tue Oct 15 09:46:56 2013

Default process/thread affinity and default OMP_NUM_THREADS

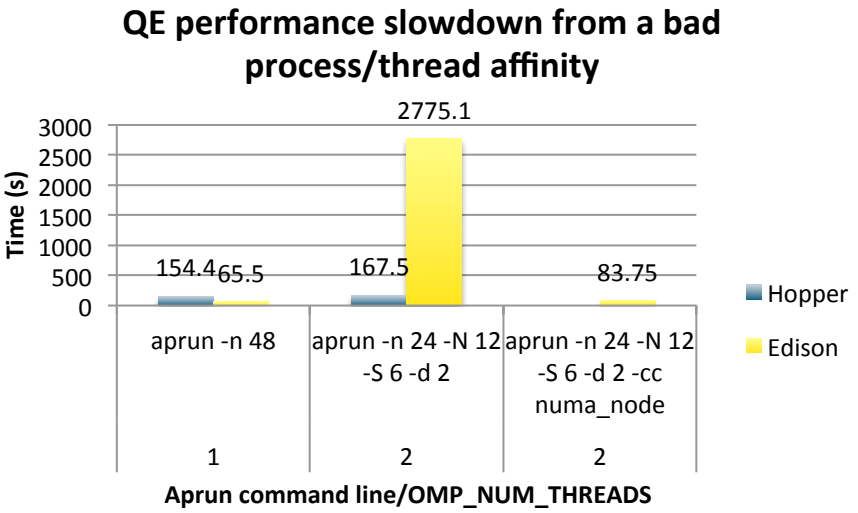
Machine default: one-on-one process/thread to core binding

Compiler	Process/thread affinity	Default OMP_NUM_THREADS
Intel	• Pure MPI codes, the process affinity works fine if running on fully packed nodes • There are issues with thread affinity - all threads from an MPI task are pined to a single core where the MPI task is placed. • An extra thread created by the Intel OpenMP runtime interacts with the CLE thread binding mechanism and causes poor performance	The number of cpu slots available
Cray	Works fine	OMP_NUM_THREADS=1
GNU	Works fine	The number of cpu slots available

Process/thread binding with a binary compiled with an Intel compiler

Export OMP_NUM_THREADS=6
aprun –n4 -N4 -S2 -d6 xthi.intel

Hello from rank 0, thread 0, on nid02877. (core affinity = 0)
Hello from rank 0, thread 1, on nid02877. (core affinity = 0)
Hello from rank 0, thread 2, on nid02877. (core affinity = 0)
Hello from rank 0, thread 3, on nid02877. (core affinity = 0)
Hello from rank 0, thread 4, on nid02877. (core affinity = 0)
Hello from rank 0, thread 5, on nid02877. (core affinity = 0)
Hello from rank 1, thread 0, on nid02877. (core affinity = 6)
Hello from rank 1, thread 1, on nid02877. (core affinity = 6)
Hello from rank 1, thread 2, on nid02877. (core affinity = 6)
Hello from rank 1, thread 3, on nid02877. (core affinity = 6)
Hello from rank 1, thread 4, on nid02877. (core affinity = 6)
Hello from rank 1, thread 5, on nid02877. (core affinity = 6)
Hello from rank 2, thread 0, on nid02877. (core affinity = 12)
Hello from rank 2, thread 1, on nid02877. (core affinity = 12)
Hello from rank 2, thread 2, on nid02877. (core affinity = 12)
Hello from rank 2, thread 3, on nid02877. (core affinity = 12)
Hello from rank 2, thread 4, on nid02877. (core affinity = 12)
Hello from rank 2, thread 5, on nid02877. (core affinity = 12)
Hello from rank 3, thread 0, on nid02877. (core affinity = 18)
Hello from rank 3, thread 1, on nid02877. (core affinity = 18)
Hello from rank 3, thread 2, on nid02877. (core affinity = 18)
Hello from rank 3, thread 3, on nid02877. (core affinity = 18)
Hello from rank 3, thread 4, on nid02877. (core affinity = 18)
Hello from rank 3, thread 5, on nid02877. (core affinity = 18)



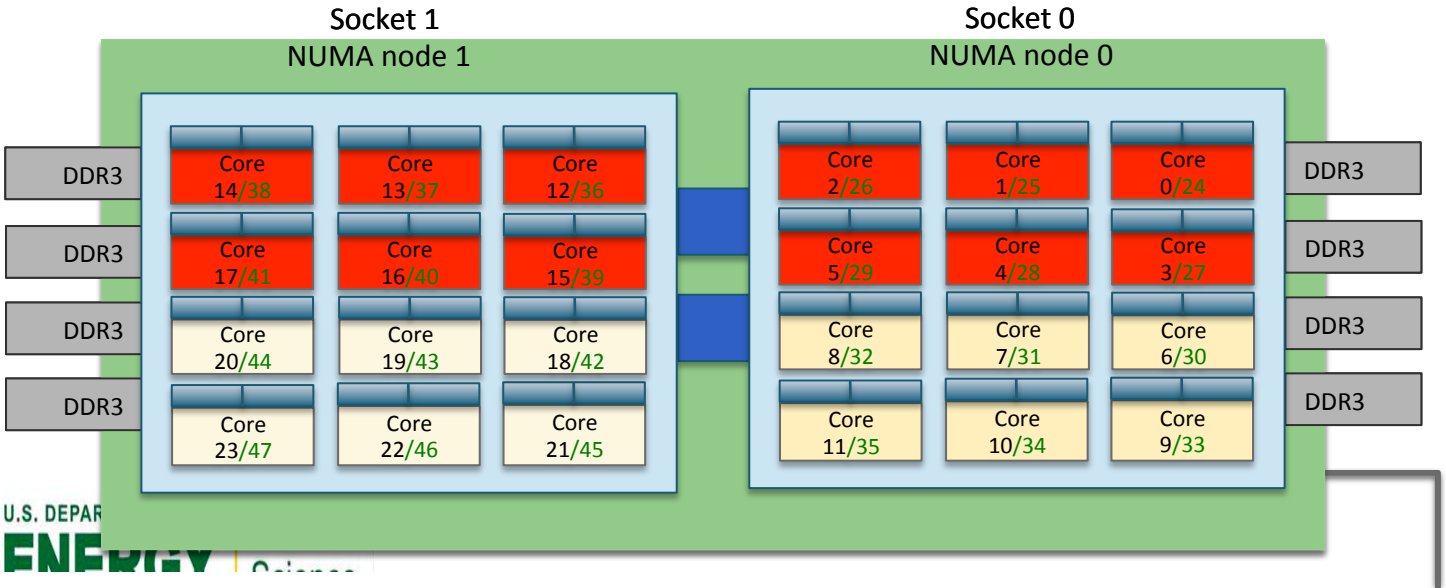
Aprun's –S option need to be used to evenly distribute MPI tasks to the two NUMA nodes

aprun -n 12 -N12 xthi.intel

Hello from rank 0, thread 0, on nid06119. (core affinity = 0)
Hello from rank 1, thread 0, on nid06119. (core affinity = 1)
Hello from rank 2, thread 0, on nid06119. (core affinity = 2)
Hello from rank 3, thread 0, on nid06119. (core affinity = 3)
Hello from rank 4, thread 0, on nid06119. (core affinity = 4)
Hello from rank 5, thread 0, on nid06119. (core affinity = 5)
Hello from rank 6, thread 0, on nid06119. (core affinity = 6)
Hello from rank 7, thread 0, on nid06119. (core affinity = 7)
Hello from rank 8, thread 0, on nid06119. (core affinity = 8)
Hello from rank 9, thread 0, on nid06119. (core affinity = 9)
Hello from rank 10, thread 0, on nid06119. (core affinity = 10)
Hello from rank 11, thread 0, on nid06119. (core affinity = 11)

aprun -n 12 -N12 **-S6** xthi.intel

Hello from rank 0, thread 0, on nid06119. (core affinity = 0)
Hello from rank 1, thread 0, on nid06119. (core affinity = 1)
Hello from rank 2, thread 0, on nid06119. (core affinity = 2)
Hello from rank 3, thread 0, on nid06119. (core affinity = 3)
Hello from rank 4, thread 0, on nid06119. (core affinity = 4)
Hello from rank 5, thread 0, on nid06119. (core affinity = 5)
Hello from rank 6, thread 0, on nid06119. (core affinity = 12)
Hello from rank 7, thread 0, on nid06119. (core affinity = 13)
Hello from rank 8, thread 0, on nid06119. (core affinity = 14)
Hello from rank 9, thread 0, on nid06119. (core affinity = 15)
Hello from rank 10, thread 0, on nid06119. (core affinity = 16)
Hello from rank 11, thread 0, on nid06119. (core affinity = 17)



Manipulate process/thread affinity

- **-S, -sn, -sl, -cc, and -ss options control how your application uses the NUMA nodes.**
  - -n Number of MPI tasks.
  - -N (Optional) Number of MPI tasks per Edison Node. Default is 24.
  - -S (Optional) Number of tasks per NUMA node. Values can be 1-12; default 12
  - -sn (Optional) Number of NUMA nodes to use per Edison node. Values can be 1-2; default 2
  - -ss (Optional) Demands strict memory containment per NUMA node. The default is the opposite - to allow remote NUMA node memory access.
  - -cc (Optional) Controls how tasks are bound to cores and NUMA nodes. The default setting on Edison is -cc cpu which restricts each task to run on a specific core.
- **These options are important on Edison if you use OpenMP or if you don't fully populate the Edison nodes.**

http://portal.nersc.gov/project/training/EdisonPerformance2013/affinity

Recommended aprun options to assure appropriate process/thread affinity

- **Running on unpacked nodes**

```
#PBS –l mppwidth=48    #2 nodes
aprun –n 24 –N 12 –S 6 ./a.out
```

- **Running with OpenMP threads**

```
#for threads per task <= 12
    setenv OMP_NUM_THREADS 12
    #for binaries compiled with Intel compilers
    aprun –n 4 –N 2 -S 1 –d 12 –cc numa_node ./a.out
    # for binaries compiled with GNU or Cray compilers.
    aprun -n 4 -N 2 -S1 -d 12 ./a.out

#for threads per task>12 and  <= 24
    export OMP_NUM_THREADS=24
    #for binaries compiled with Intel compilers
    aprun –n 2 –N 1 –d 24 –cc none ./a.out
    # for binaries compiled with GNU or Cray compilers.
    aprun –n 2 –N 1 -d 24 ./a.out
```

Hyper-Threading (HT) on Edison

- **Cray compute nodes booted with Hyper-Threads always ON**
- **Users can choose to run with one or two tasks/threads per core**
- **Use aprun –j2 option to use Hyper-threading**
  - aprun –j1 –n … Single Stream mode, one rank/thread per core
  - aprun –j2 –n … Dual Stream mode, two ranks/threads per core
  - Default is Single Stream mode
- **Dual Stream is often better if**
  - throughput is more important
  - your code scales extremely well
  - When running at relatively low core counts
- **Single Stream is often better if …**
  - single job performance matters more
  - code does not scale well
- **NERSC-6 SSP applications 4 out of 7 ran with HT**
- **However, HT may hurt code performance, use with caution.**

https://www.nersc.gov/users/computational-systems/edison/performance-and-optimization/hyper-threading/

Core specialization

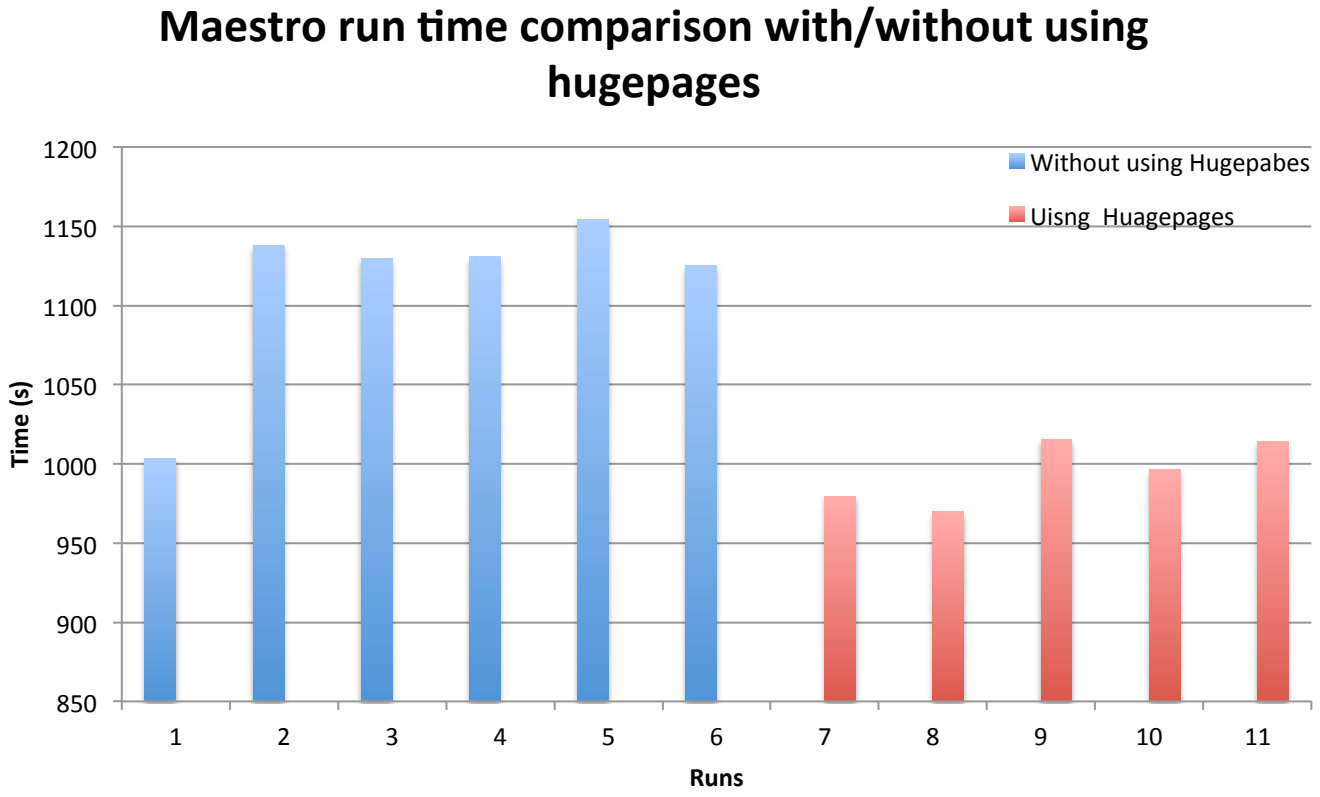
- **System 'noise' on compute nodes may significantly degrade scalability for some applications**
- **Core Specialization can mitigate this problem**
  - M core(s)/cpu(s) per node will be dedicated for system work (service core)
  - As many system interrupts as possible will be forced to execute on the service core
  - The application will not run on the service cpus
- **Use aprun -r to get core specialization**
  - aprun –r[1-8] –n 100 a.out
  - Highest numbered cpus will be used
  - Starts with cpu 48 on Ivy Bridge e nodes
  - Independent of aprun –j setting
- **Apcount provided to compute total number of cores required**
- **Tests with NERSC-6 benchmark codes shows that the impact of core specialization is at best negligible and often negative.**

https://www.nersc.gov/users/computational-systems/edison/performance-and-optimization/core-specialization/

Hugepages may improve your code performance

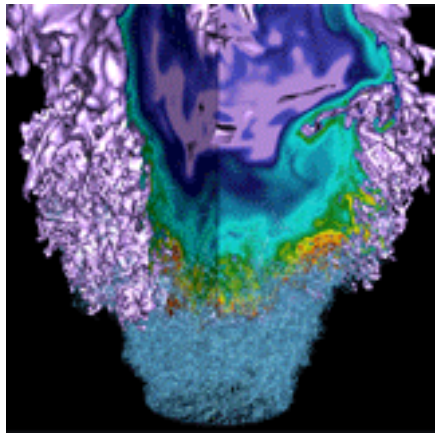
- **Hugepages may improve memory performance for common access patterns on large data sets.**
- **The Aries may perform better with HUGE pages than with 4K pages.**
  - HUGE pages use less Aries resources than 4k pages
  - More important when remotely access large percentage of nodes memory in an irregular manner
- **May get "cannot run errors" if there are not enough Hugepages memory available (memory page fragmentation)**
- **Use modules to change default page sizes (man intro_hugepages)**
  - craype-hugepages2M, craype-hugepages4M, craype-hugepages8M, craype-hugepages16M, craype-hugepages32M, craype-hugepages64M, craype-hugepages128M,craype-hugepages256M,craype-hugepages512M
- **Users are recommended to experiment with hugepages**
- **This feature is implemented at link and run time, to use**
  - Module load craype-hugepages2M
  - cc -o my_app my_app.c
  - Then run with the same hugepages module loaded

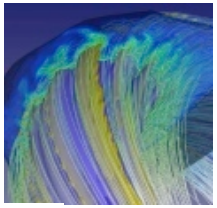
Hugepages may improve your code performance

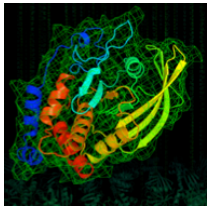


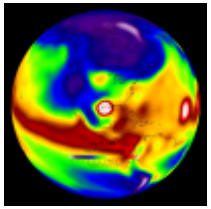
Mastro run time improves by 11% by average when using hugepage memory compared to not using the hugepages.

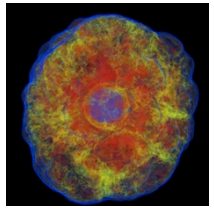
Node placements on Edison

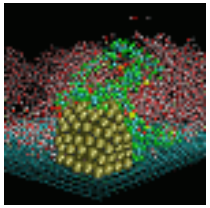










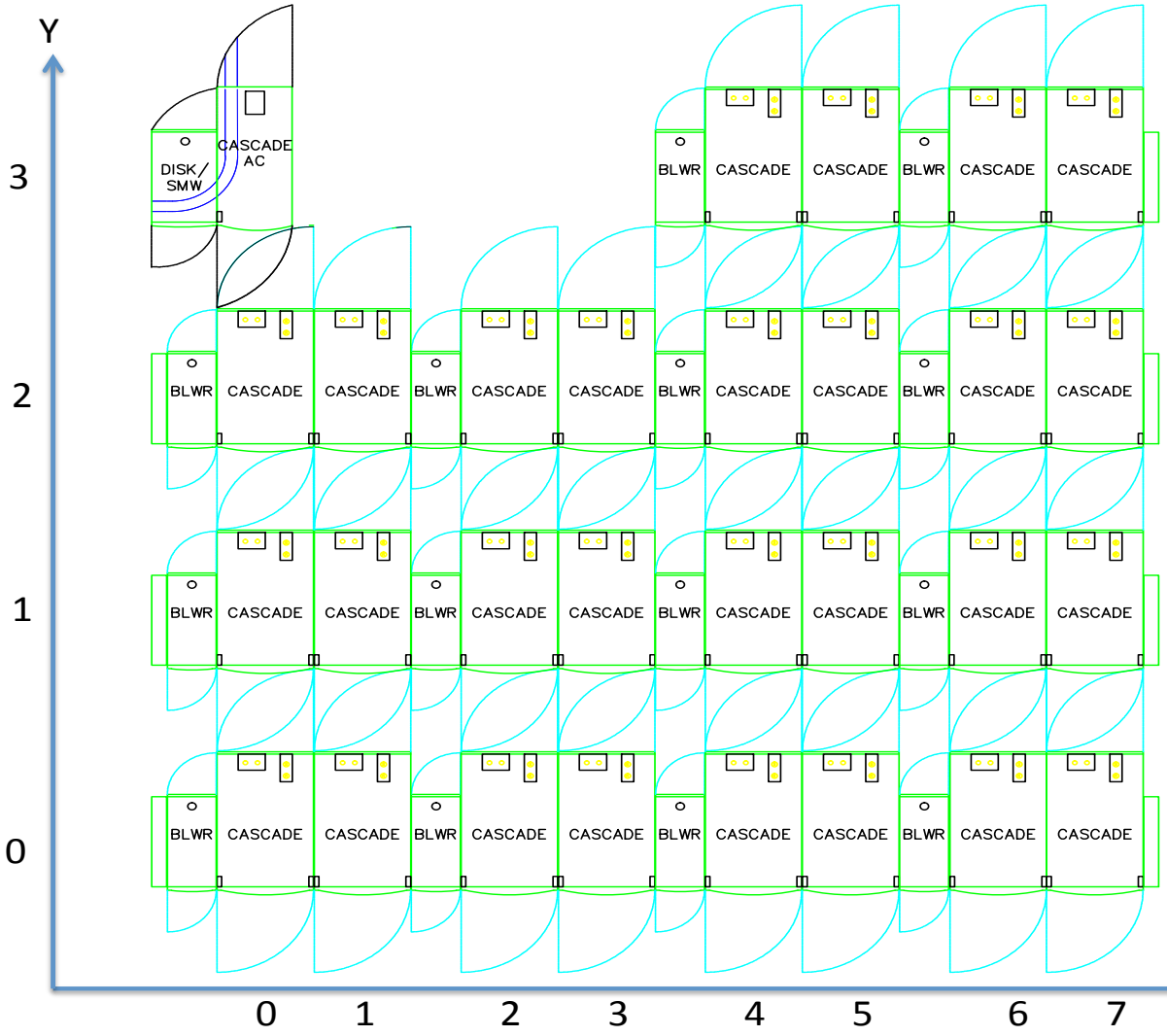


NERSC-6 application benchmark production and dedicated time comparison

Application	CAM	GAMESS	GTC	IMPACT-T	MAESTRO	MILC	PARATEC
Concurrency	240	1024	2048	1024	2048	8192	1024
Streams/Core	2	2	2	2	1	1	1
Dedicated Time (s)	273.08	1,125.80	863.88	579.78	935.45	446.36	173.51
Production Time(s)	277.07	1,218.17	871.06	597.25	996.70	482.87	198.45
Slowdown[1)]	1.5%	8.2%	0.8%	3.0%	6.5%	8.2%	14.4%

[1)] Slowdown=Time(Production)/Time(Dedicated)

Edison Cabinet Floor Layout



Edison cabinet groups:

Group 0: C0-0 C1-0
Group 1: C2-0 C3-0
Group 2: C4-0 C5-0
Group 3: C6-0 C7-0
Group 4: C0-1 C1-1
Group 5: C2-1 C3-1
Group 6: C4-1 C5-1
Group 7: C6-1 C7-1
Group 8: C0-2 C1-2
Group 9: C2-2 C3-2
Group 10: C4-2 C5-2
Group 11: C6-2 C7-2

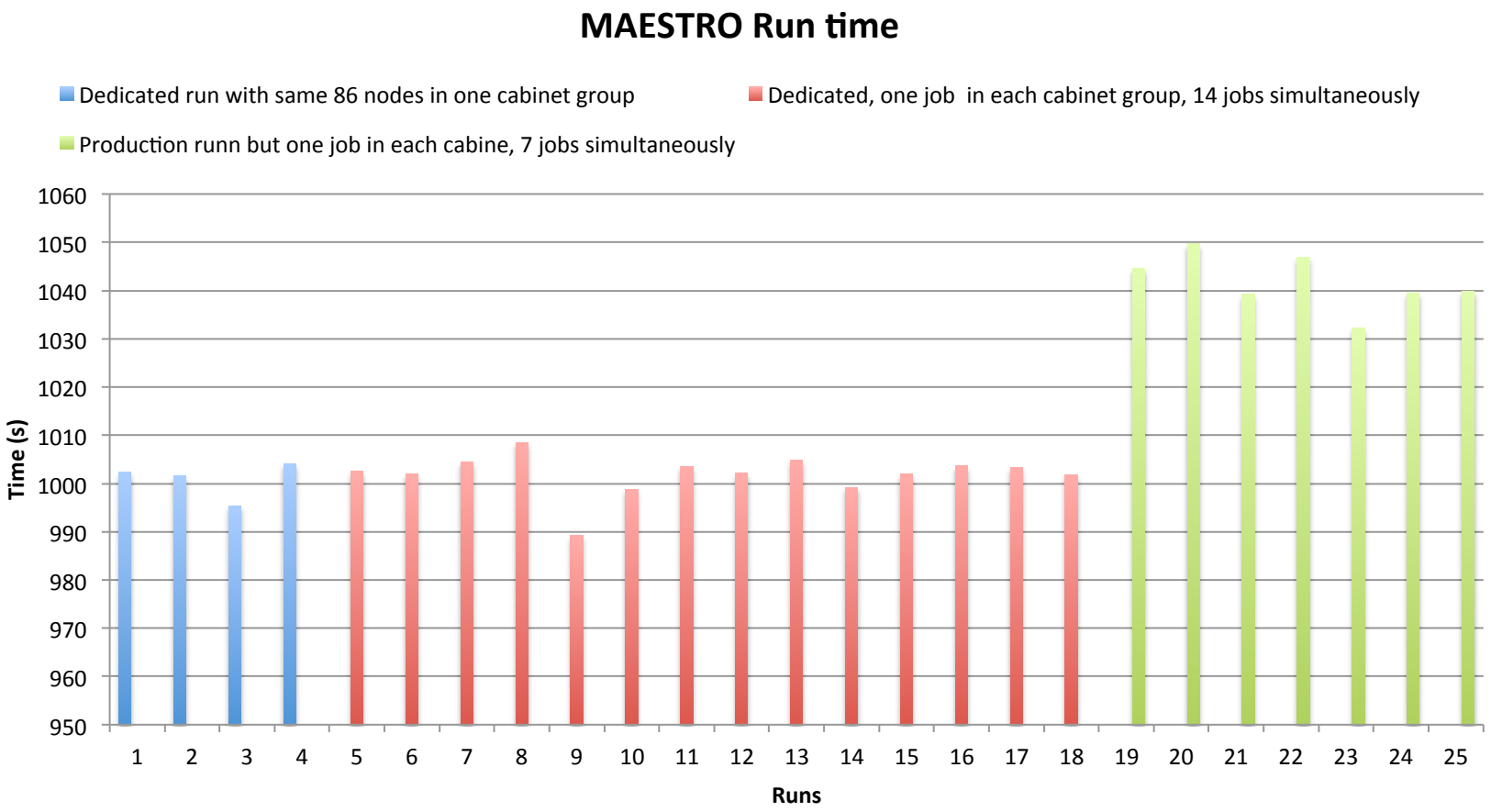
Group 14: C4-3 C5-3
Group 15: C6-3 C7-3

Note:

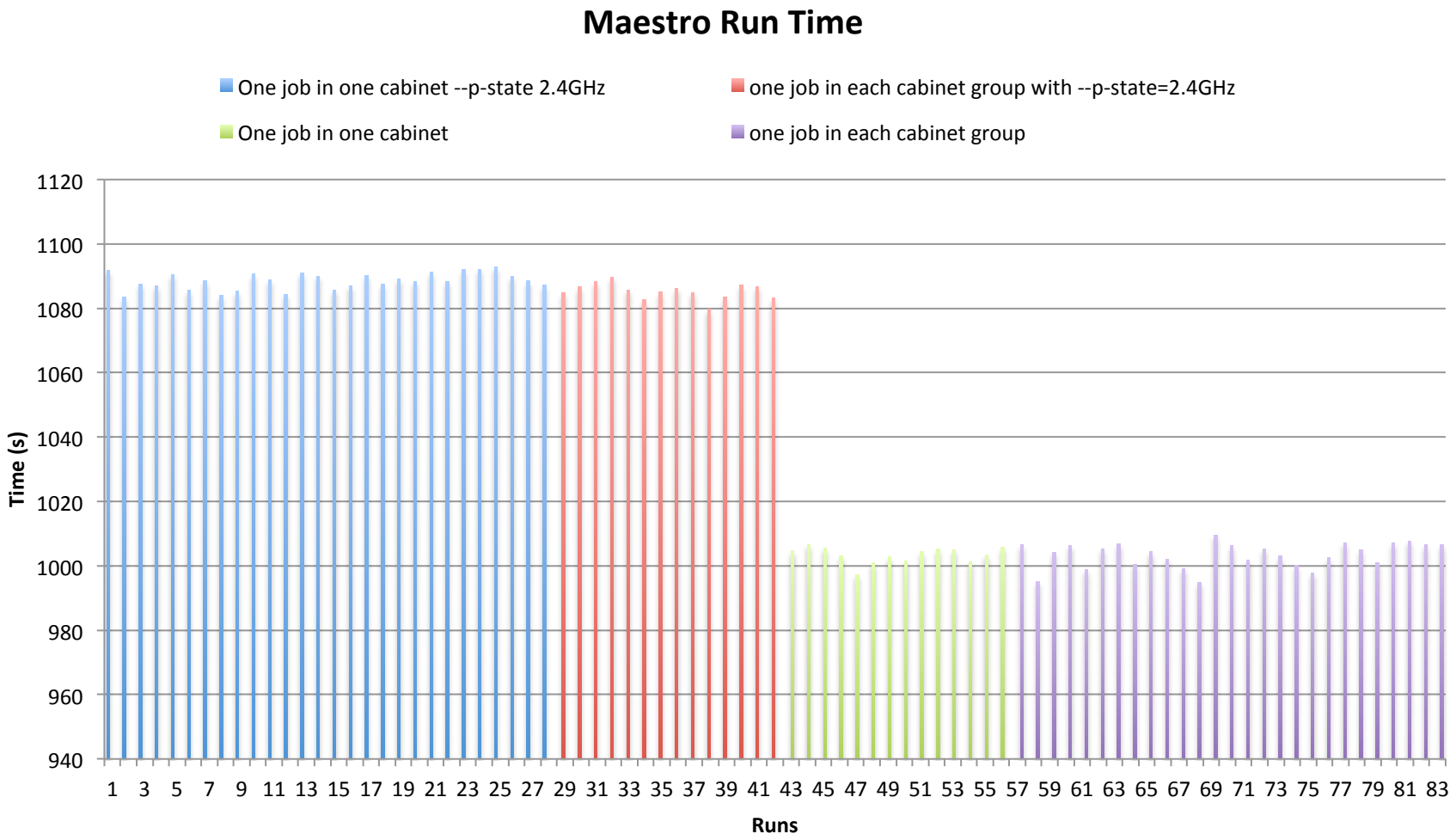
- The groups 12, and 13 are missing in our layout
- Use cnselect x_coord.eq.3 to choose the node list in the cabinet group 3

Edison has 14 cabinet groups, connected with 546 optical cables (Rank 3)

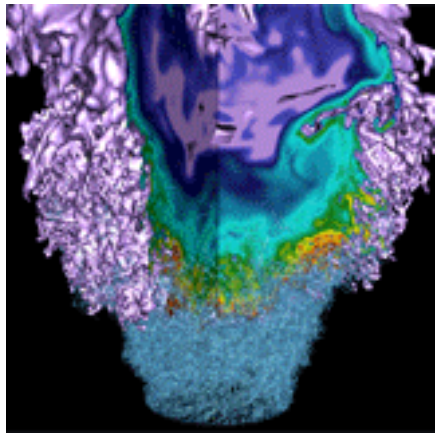
Node placements and run time

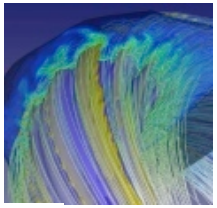


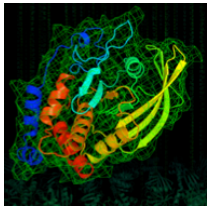
Node placements and run time

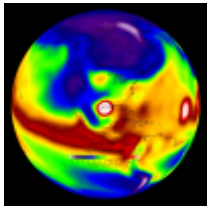


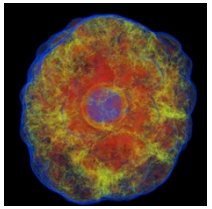
I/O performance

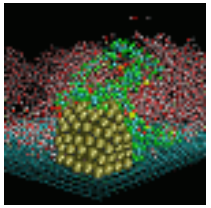












Edison has three Lustre file systems

	Size (PB)	Aggregate Peak Performance (GB/s)	# of Disks	# of OSSs	# of OSTs	Default stripe count
$SCRATCH /scratch1	2.1	48	12	24	96	2
$SCRATCH /scratch2	2.1	48	12	24	96	2
/scratch3	3.2	72	18	24	144	8

https://www.nersc.gov/users/computational-systems/edison/file-storage-and-i-o/edison-scratch3-directory-request-form/

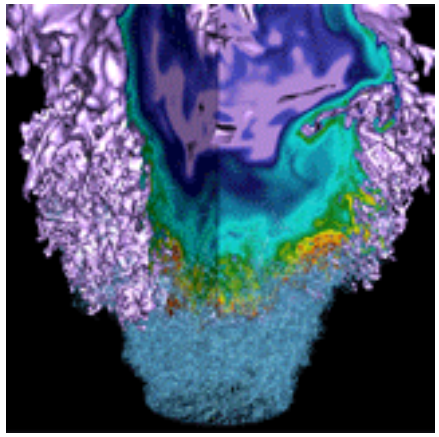
Users are encouraged to experiment with Lustre stripe count, size to obtain a good I/O performance for their workloads, with a general guidance that a larger stripe count may increase bandwidth but subject more contention, and vise versa.

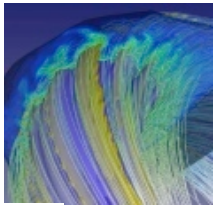
lfs setstripe
lfs getstripe
man lfs

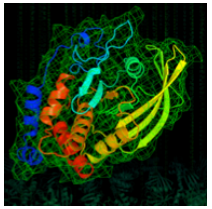
Many factors may affect the I/O performance of your jobs

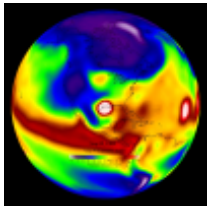
- **Contentions for the resources with other users**
- **Hardware failure or downgraded performance**
- **File system fragmentations**
- **Bad user practices**
  - A user used fixed offset, and stripecount 1 and filled up one of the OSTs a couple of times.
  - Using too large stripe counts for small file I/O inviting contention with other users unnecessarily and get widely varying I/O time

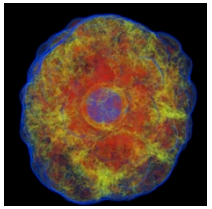
Python applications at scale

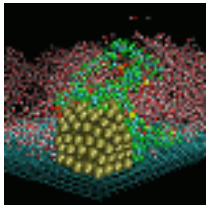




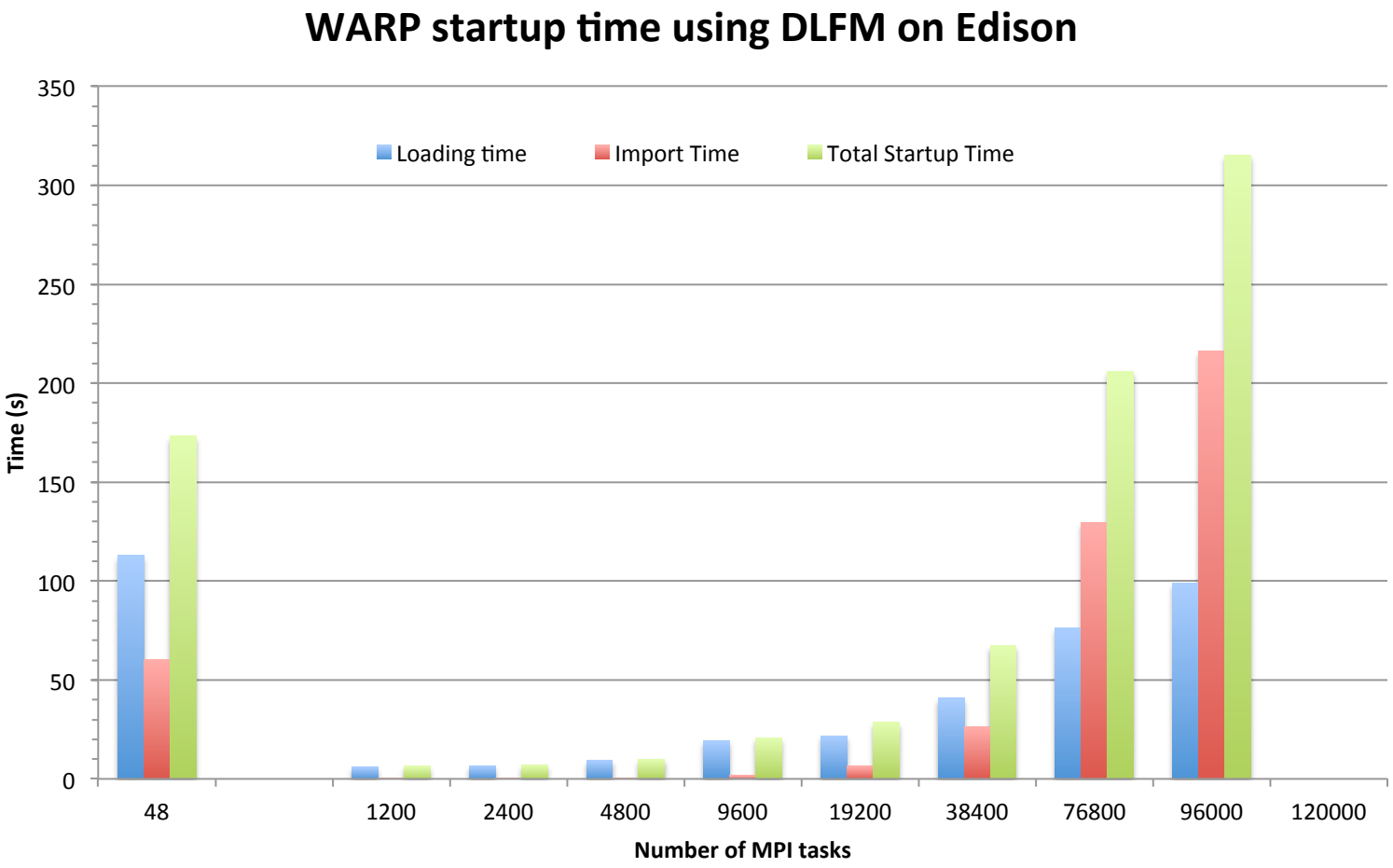








DLFM method effectively reduces python application startup time

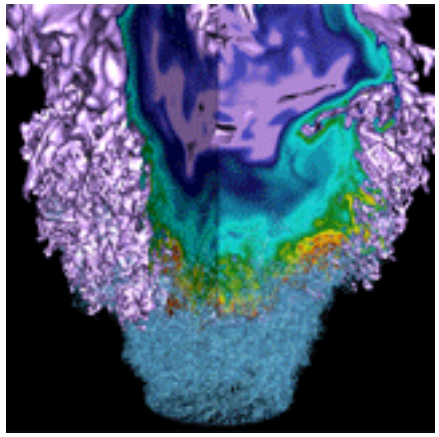


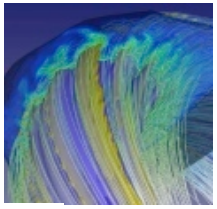
Warp startup time is ~1 minutes at 38.4K cores!

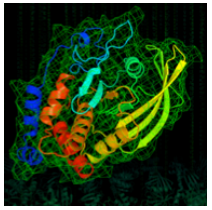
Using DLFM module for large scale python applications

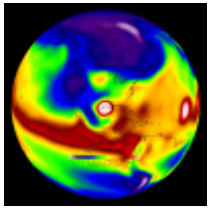
- **DLFM, developed by Mike Davis at Cray, Inc, is a library tool to reduce the python application startup time at large scale.**
- **To access, do module load dlfm**
- **Compile your code using the python available via the dlfm module**
- **Run with two steps**
  - Pilot run with small node count, eg., using 2 nodes collect the needed shared libraries and python modules imported
  - Real run with large number of cores, only one core read in the shared libraries and python imported modules
- **More info is in the DLFM website**

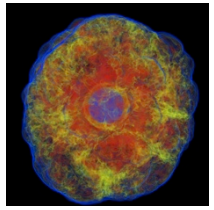
Cluster Compatibility Mode

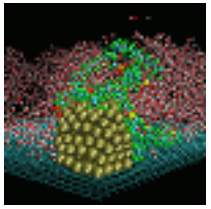












Cluster compatibility mode (CCM)

- CCM is available on Edison to run TCP/IP applications or ISV (Independent Software Vendor) applications.
- G09 and Wien2k run via CCM because they need ssh to compute nodes
- Running g09 over multiple nodes are not recommended due to a performance issue with CCM and also g09's relatively low parallel scalability.

- https://www.nersc.gov/users/computational-systems/edison/cluster-compatibility-mode/

Thank you.