Hopper quiz

- Hopper compute nodes run _____
  A. Light version of Operating System
  B. Standard (full) Linux Operating System

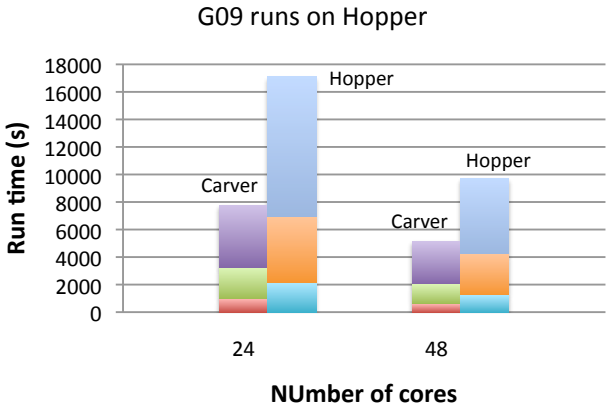
- Hopper compute nodes don't support _____
  A. Dynamic shared libraries
  B. ssh
  C. Both A and B

Hopper quiz

- Hopper compute nodes run ___A___
  A. Light version of Operating System
  B. Standard (full) Linux Operating System

- Hopper compute nodes don't support ___B___
  A. Dynamic shared libraries
  B. **ssh**
  C. Both A and B

This talk is about a way to convert part of Hopper nodes to a generic Linux cluster where you can do ssh between compute nodes



Cluster Compatibility Mode on Hopper

Zhengji Zhao and Helen He

February 2, 2012

NERSC User Group meeting

U.S. DEPARTMENT OF ENERGY | Office of Science

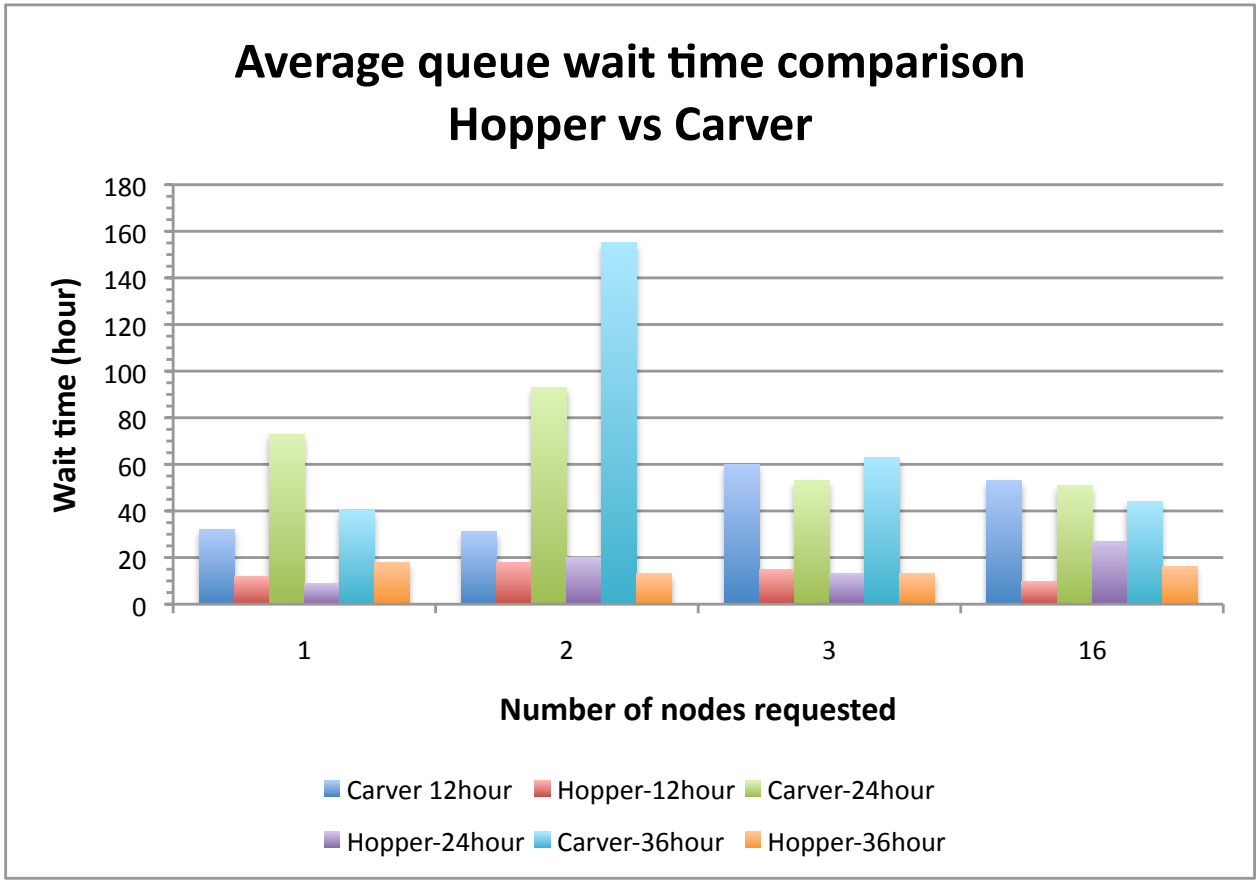
NERSC National Energy Research Scientific Computing Center

BERKELEY LAB Lawrence Berkeley National Laboratory

Why is Cluster Compatibility Mode (CCM) interesting?

- NERSC has a diverse workload, some applications don't run on Hopper due to the lack of TCP/IP support on compute nodes
  - Gaussian code, NAMD replica jobs
  - wien2k
- Interactive IDL, Matlab, etc.
  - With exclusive and larger memory access.
  - Process scratch data locally
- Long queue turnaround on Carver (IBM iDataPlex)

Queue wait time for the last 3 months on Carver and Hopper



2 node jobs requesting 36 hours waited 9 times more on Carver than on Hopper!

Data covers the period of 7/1/2011-10/1/2011 and obtained from https://www.nersc.gov/users/job-information/usage-reports/jobs-summary-statistics/

Outline

- Introduction to CCM
- Applications that CCM enables
- Performance of CCM
- Conclusion
- Hands on (10 minutes)

What is Cluster Compatibility Mode (CCM)?

- CCM is a Cray software solution that provides services needed to run most cluster-based independent software vendor (ISV) applications on the Cray XE6. CCM supports
  - TCP/IP - MPI runs over it
  - Standard services: ssh, rsh, nscd, ldap
  - Complete root file system
- CCM is implemented as a Hopper queue, ccm-queue
  - Dynamically allocates and configures compute nodes when a CCM job starts
  - ccmrun - launch jobs onto the head node
  - ccmlogin – interactively login to compute nodes
  - Nodes are returned to compute nodes pool when a job exits

In CCM applications run in a generic Linux cluster environment

Programming environment of CCM

- Compilers
  - PGI, GNU, INTEL, PATHSCALE, CRAY
  - Default - dynamic linking
- Libraries
  - openmpi/1.4.3 #built outside of the batch system
  - scalapack, fftw/2.1.5, mkl
  - acml, and all serial libraries for Hopper native environment
- Debuggers and profilers
  - ipm-ccm; DDT will be installed soon
- Applications
  - G09 (ssh+OpenMP), NAMD (MPI + socket operations), WIEN2k (MPI+ssh)
  - VASP (MPI), Quantum Espresso (MPI+OpenMP)

How to compile codes for CCM

Compiling codes on mom nodes:

```
qsub –I -l mppwidth=24 –q ccm_queue –l walltime=30:00

module load openmpi

mpicc –mp xhi.c          #dynamic binary by default
Mpif90  test1.f90
```

Alternatively, do direct ssh to get on to one of the 24 mom nodes:

```
ssh hmom05
```

How to run jobs under CCM - submit jobs to the ccm_queue queue

```
zz217@nid00002:~/tests/coreid> qsub -I -l mppwidth=48 -q ccm_queue -l walltime=30:00
qsub: waiting for job 8045.sdb to start
qsub: job 8045.sdb ready

In CCM JOB:  8045.sdb  JID  sdb  USER  zz217  GROUP  zz217
Initializing CCM environment, Please Wait
CCM Start success, 2 of 2 responses

    3.1.61 from xe-image-3.1.61g39.iso

Directory: /global/homes/z/zz217
Thu Dec  8 02:25:00 PST 2011
zz217@nid00029:~> cd $PBS_O_WORKDIR
zz217@nid00029:~/tests/coreid> module load ccm
zz217@nid00029:~/tests/coreid> module load openmpi
zz217@nid00029:~/tests/coreid> export CRAY_ROOTFS=DSL
zz217@nid00029:~/tests/coreid> ccmrun mpirun –np 2 -bynode -hostfile $PBS_NODEFILE
hostname
.....
zz217@nid00029:~/tests/coreid> aprun -n 2 -N1 hostname
nid00019
nid00018
Application 252074 resources: utime ~0s, stime ~0s
```

ccmloign -interactive access to compute nodes

Continued …

```
zz217@nid00029:~/tests/coreid> ccmlogin -V
Last login: Thu Dec  8 02:54:37 2011 from nid00029



    3.1.61 from xe-image-3.1.61g39.iso

zz217@nid00018:~> cat $PBS_NODEFILE|sort -u
nid00018
Nid00019

zz217@nid00018:~> ssh nid00019
Last login: Tue Dec  6 21:21:58 2011 from nid00010

    3.1.61 from xe-image-3.1.61g39.iso
zz217@nid00019:~> cat $PBS_NODEFILE

zz217@nid00019:~>  top
```

-V passes environment variables to head node

Now you can ssh to compute nodes under CCM!

Two special issues with CCM (mpirun)

- mpirun … –hostfile $PBS_NODEFILE …
- Pass environment variables to remote CCM nodes
  - shell startup files
  - --prefix and –x
  - env >~/.ssh/environment
  - Build static binary #static build doesn't work yet due to missing some lower level static libraries

Sample job script for MPI jobs

```
#!/bin/bash -l
#PBS -N test_ccm
#PBS -q ccm_queue
#PBS -l mppwidth=48,walltime=30:00
#PBS -j oe

cd $PBS_O_WORKDIR
module load ccm
export CRAY_ROOTFS=DSL
mpicc xthi.c
ccmrun mpirun -np 48 -hostfile $PBS_NODEFILE a.out
```

In .bashrc.ext add

```
module load openmpi
```

How to run OpenMP+MPI jobs

```
.bashrc.ext
module load openmpi/1.4.3
export OMP_NUM_THREADS=6
```

```
#!/bin/bash -l
#PBS -N test_ccm
#PBS -q ccm_queue
#PBS -l mppwidth=48,walltime=30:00
#PBS -j oe

cd $PBS_O_WORKDIR
module load ccm
export CRAY_ROOTFS=DSL
mpicc -mp xthi.c
export OMP_NUM_THREADS=6

ccmrun mpirun -np 8 -cpus-per-proc 6 -bind-to-core -hostfile $PBS_NODEFILE a.out
```

Process affinity:

-cpus-per-proc 6 -bind-to-core

Alternatively,

ccmrun mpirun –x OMP_NUM_THREADS –prefix /usr/common/usg/openmpi/1.4.3/pgi -np 8 -cpus-per-proc 6 -bind-to-core -hostfile $PBS_NODEFILE a.out

Gaussian 09 job script

```
#PBS -S /bin/tcsh
#PBS -N ccm_g09
#PBS -q ccm_queue
#PBS -l mppwidth=48,walltime=24:00:00
#PBS -j oe

module load ccm
setenv CRAY_ROOTFS DSL

set input=input
set output=output.L2S24.$PBS_JOBID
module load g09

mkdir -p $SCRATCH/g09/$PBS_JOBID
cd $SCRATCH/g09/$PBS_JOBID
ccmrun g09l < $PBS_O_WORKDIR/$input >
$PBS_O_WORKDIR/$output
```

```
.tcshrc.ext:
module load g09
```

Where g09l is a csh script defines a few envs and use PBS_NODEFILE to pass the node list for the job.

Sample job script to run multiple serial Jobs

- Currently not supported on Hopper
  - Under CCM:

```
#!/bin/bash -l
#PBS -q ccm_queue
#PBS -l mppwidth=24
#PBS –l walltime=1:00:00

cd $PBS_O_WORKDIR
module load ccm
export CRAY_ROOTFS=DSL
ccmrun ./a1.out &
ccmrun ./a2.out &
…
ccmrun ./a8.out&
wait
```

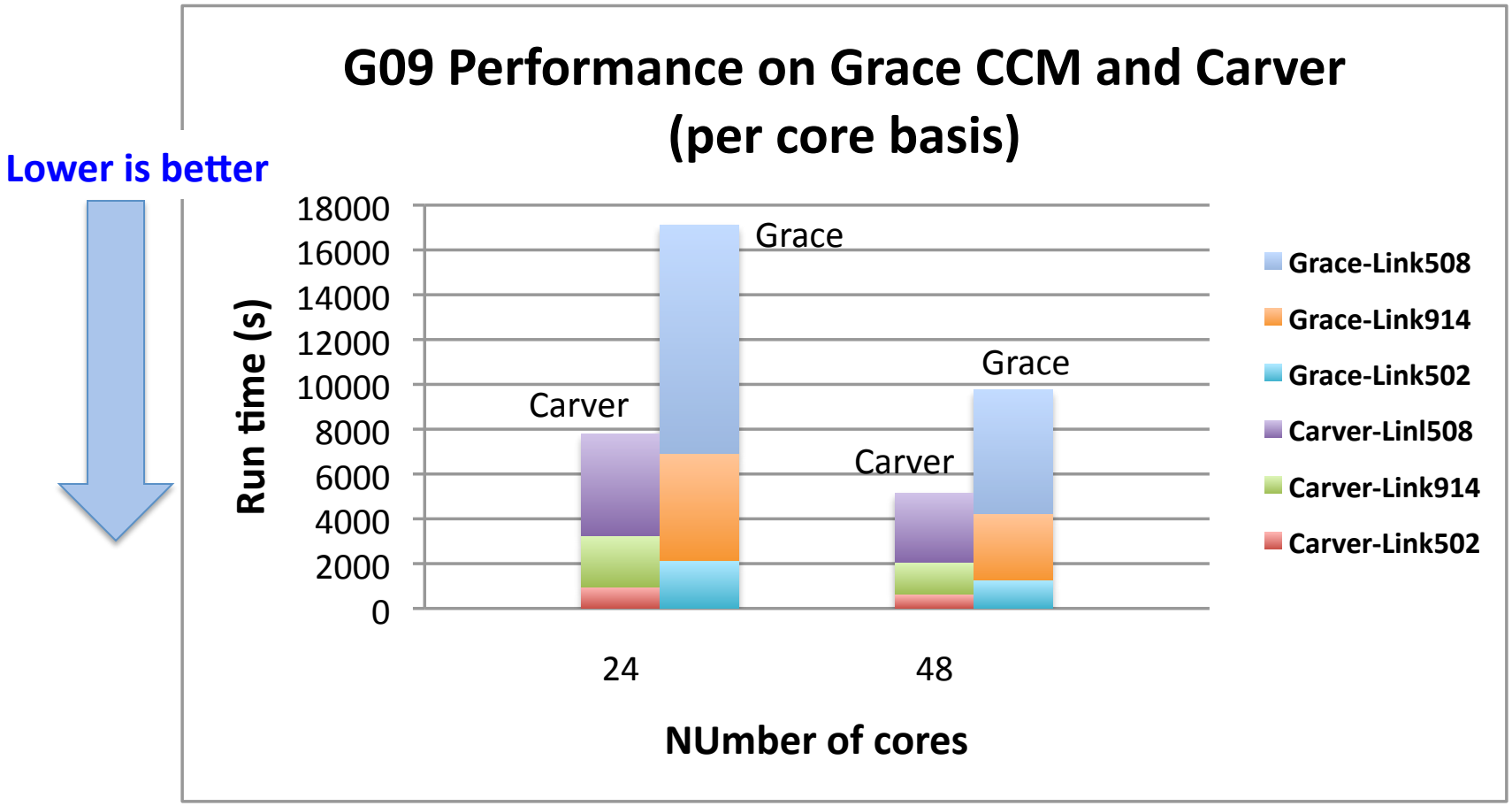
```
OR:
% qsub –I –V -q ccm_queue -l
mppwidth=24 –l walltime=1:00:00
..
% module load ccm
export CRAY_ROOTFS=DSL
% ccmlogin

% ./a1.out &
  ./a2.out &
  …
  ./a8.out &
```

CCM enables G09 jobs on Hopper

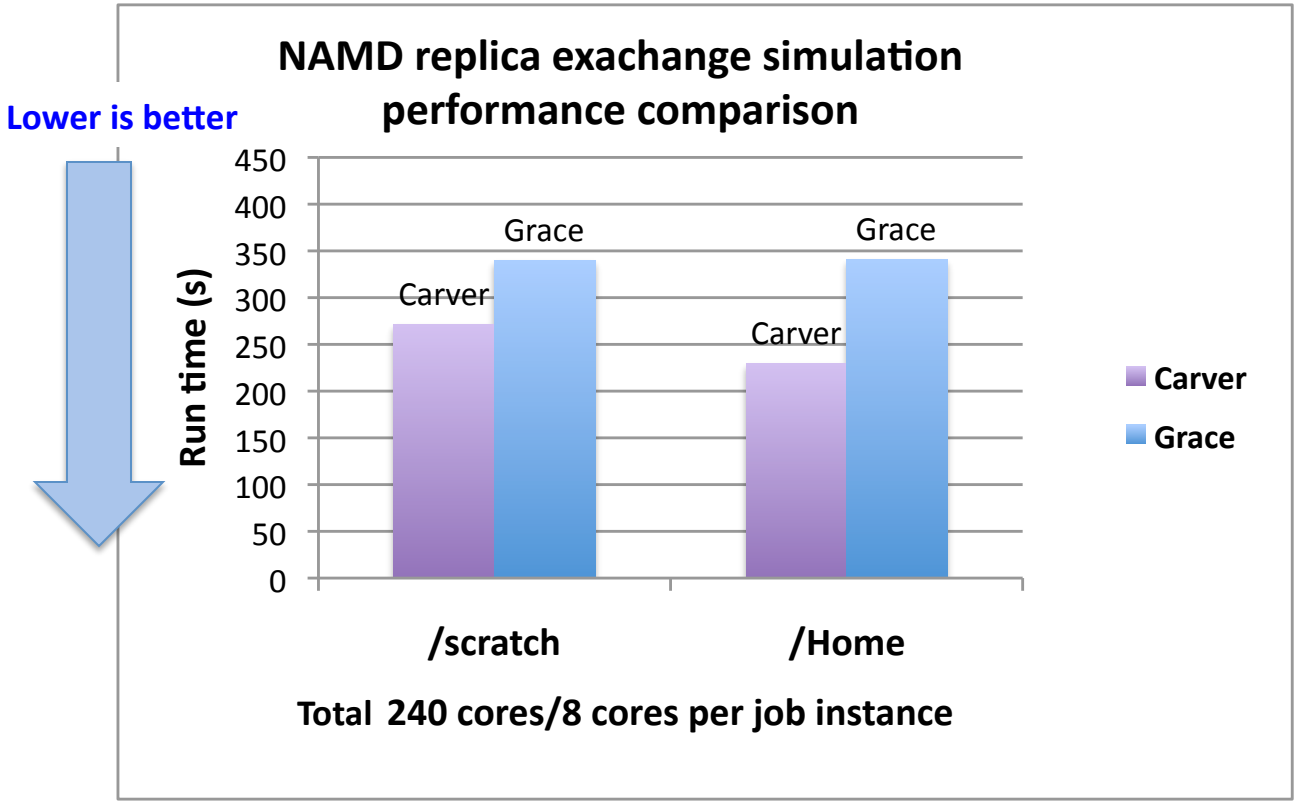
- G09 parallel implementation:
  - Master-slave mode
  - Intra node: OpenMP
  - Inter node: **ssh**
- A g09 job consists of Links - component executables
- Test case (user case):
  - UHF calculation for a system with 61 atoms, NBasis=919
  - Figure (next slide) shows the 3 most time consuming components of the job, Link 502, Link 914 and Link 508.
- CCM jobs were run on Hopper test machine, Grace

CCM enables G09 jobs on Hopper



1. G09 runs around 2 times (runtime doubled) slower than that on Carver when running on the same number of cores.
2. Most Carver g09 jobs will fit to run on 1-2 nodes on Hopper.

CCM enables NAMD replica jobs on Hopper



NAMD replica job:

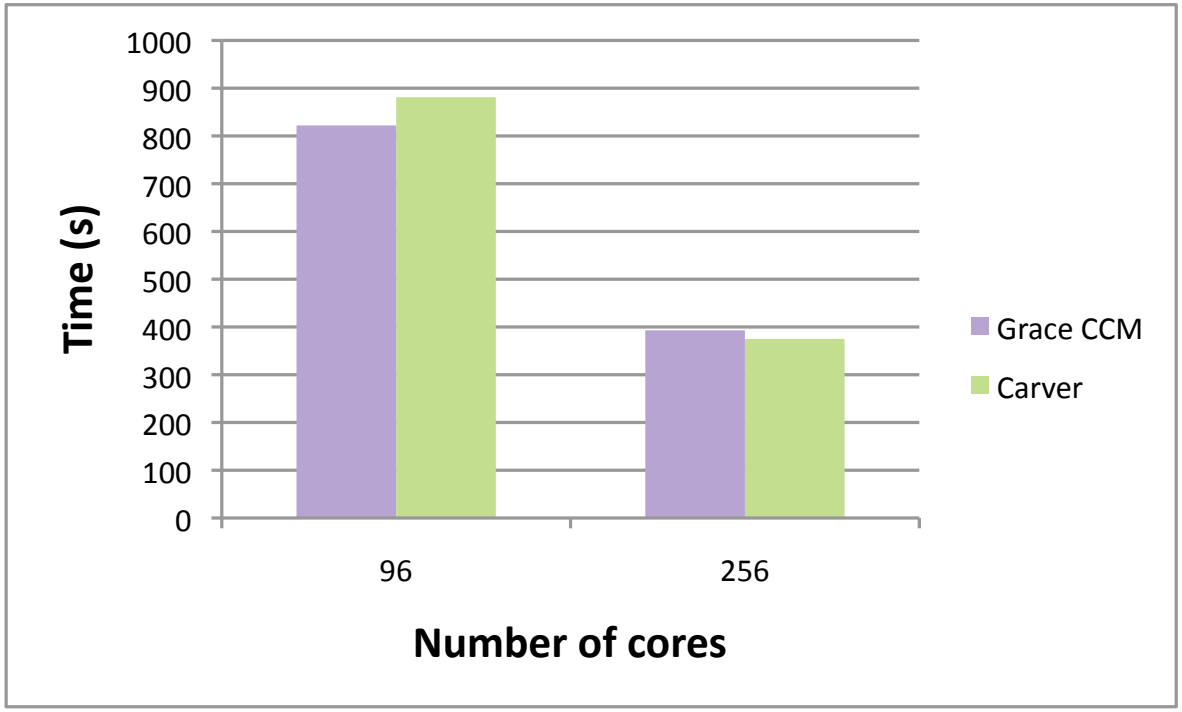
1. Runs multiple job instances simultaneously
2. Message passing within a job instance uses mpi (OpenMPI);
3. Communication between job instances uses **socket** operations

Test case (user case):

1. 30 job instances for a system with 95K atoms
2. Figure shows the run time for the first 10,000 MD steps.

1. NAMD replica job on Grace /home runs around 50% slower than that on Carver /home;
2. On Grace /scratch it runs around 25% slower than that on Carver /gscratch.

Wien2k runs on Hopper under CCM



Wien2k:

Two levels of parallelization:

- MPI for message passing within a k-point
- **ssh** between k-points
- Shell scripts used to combine a few binaries

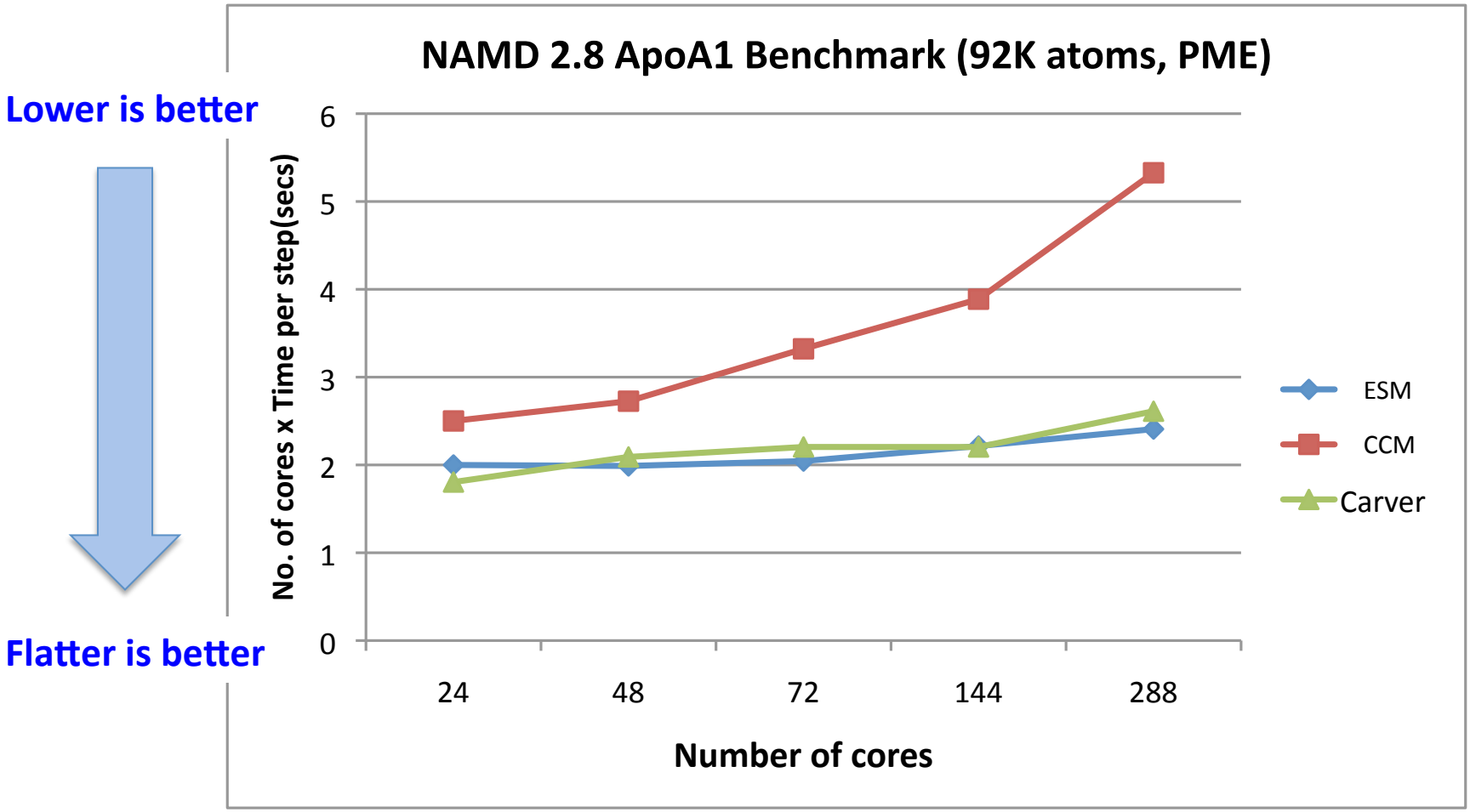
CCM allows multiple processes share a single node

Test case (standard benchmark, TiC):

- 1000 kpoints
- Each k-point used 1 core

Wien2k jobs run around the same speed on Grace under CCM when compared to Carver.

Pure MPI NAMD parallel scaling



Pure MPI NAMD with CCM (OpenMPI + fftw/2.1.5) is slower and doesn't scale when compared to Hopper. ESM is for hopper native environment

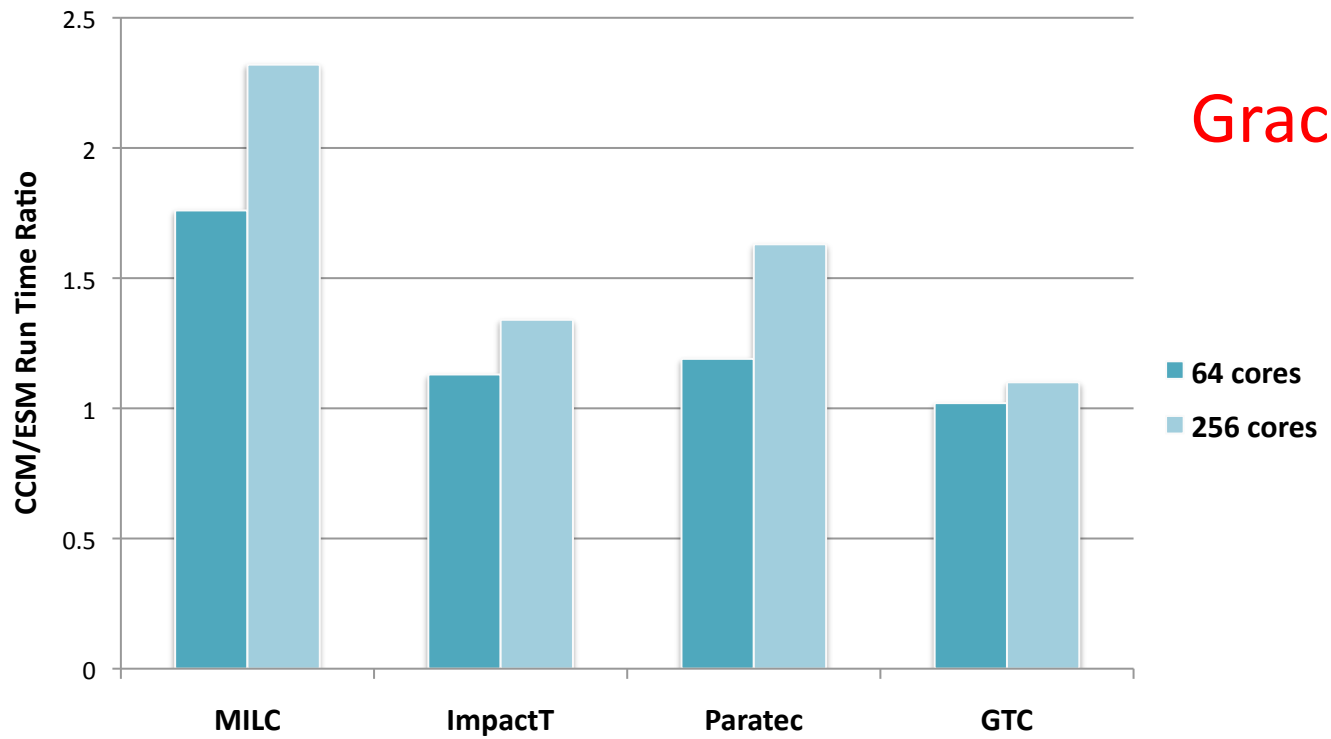
N6 Benchmark Applications

Benchmark	Science Area	Algorithm	Compiler Used	Concurrency Tested	Libraries
MILC	Lattice Gauge Physics (NP)	Conjugate Gradient, sparse matrix, FFT	GNU	64, 256	
ImpactT	Accelerator Physics (HEP)	PIC, FFT	PGI	64, 256	
Paratec	Material Science (BES)	DFT, FFT, BLAS3	PGI	64, 256	Scalapack, FFTW
GTC*	Fusion (FES)	PIC, Finite Difference	PGI	64, 256	

- Cray MPICH2 over Gemini network is used for ESM.
- OpenMPI runs over TCP/IP and utilizes OFED interconnect protocol over Gemini High Speed Network (HSN).

N6 Benchmark Performance Comparison between CCM and ESM

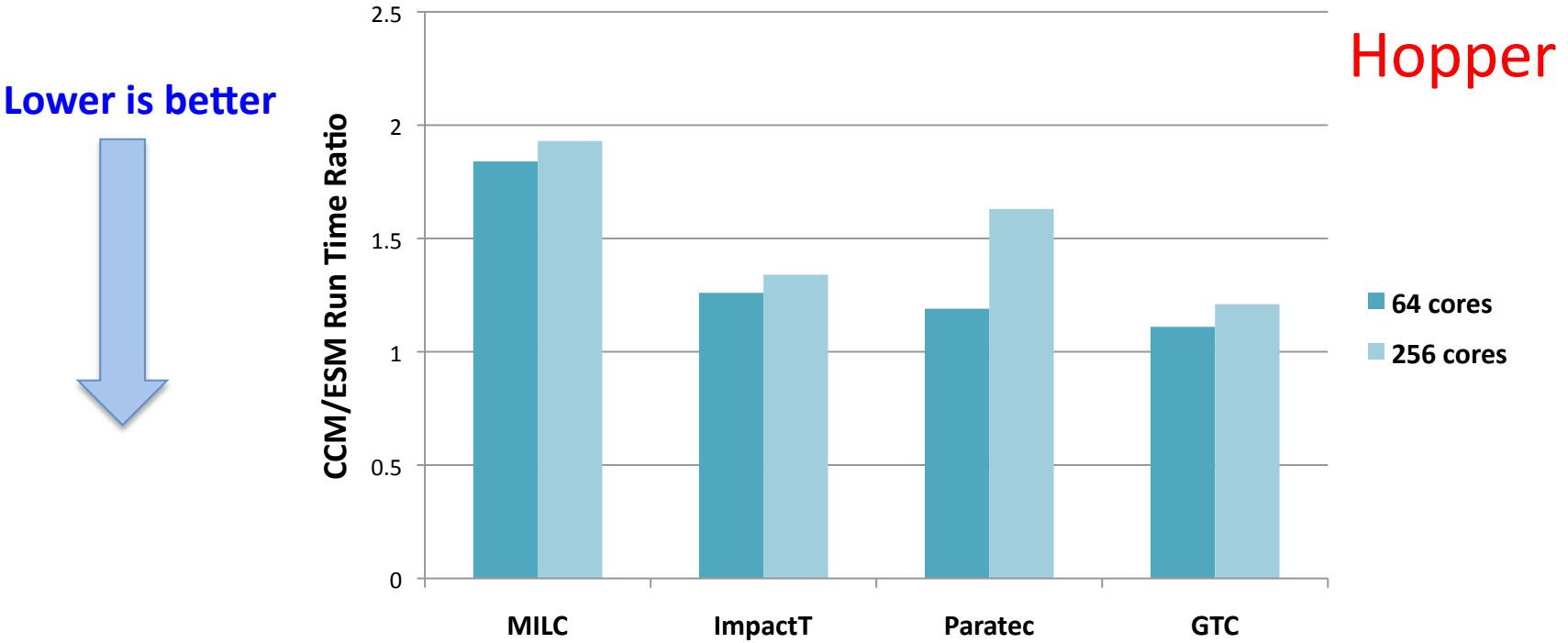
Lower is better



Grace

1. CCM slows down more with 256 cores than with 64 cores.
2. CCM run time is between 1.1 times with GTC 64 cores to 2.4 times with MILC 256 cores as of ESM.

N6 Benchmark Performance Comparison between CCM and ESM



1. CCM slows down more with 256 cores than with 64 cores.
2. CCM run time is between 1.1 times with GTC 64 cores to 1.9 times with MILC 256 cores as of ESM.

*ImpactT and Paratec failed to run at 256 cores (seg fault), so these two data were Grace results.

*Hopper CCM utilizes the interconnect protocol over Gemini interconnect

Current issues with CCM

- Performance issues
  - ISV Application Acceleration (IAA) utilizes OFED interconnect protocol over Gemini High Speed Network (HSN), work in progress
  - Process binding issue impacts performance.
  - Cray shared root file system
- Cray provides execution environment rather than development environment
  - Optimized ESM features don't work under CCM
  - Users need to deal with the complication due to the multiple layers of batch system.
  - CCM environment is not as friendly and complete as ESM environment
- No serial queue. Multiple users can not share a node.

Conclusion

- CCM extends the capability of Hopper, enables applications that couldn't run otherwise due to the lack of TCP/IP support in Hopper native environment.
- Hopper N6 benchmark suite run 1.1 - 1.9 times slower in CCM than in ESM at 256 cores.
- CCM helps the scientific productivity of our g09 users and other users who need TCP/IP network in their workflow by utilizing a shorter queue turnaround on Hopper.

Conclusion --continued

- CCM dynamic queue configuration fits in hopper existing workload seamlessly
- Additional work needed to make CCM a more pleasant user environment with better performance.
  - Cray's goal is to scale up to 2048 PE's per job
  - Provide modules for CCM environment, eg., PrgEnv-pgi-ccm
- CCM is in experimental stage now
  - Charge factor, max walltime, etc., are subject to a further adjustment

Users are encouraged to try out CCM on Hopper

10 minutes hands on

- Get the materials
  - cp –pr /project/projectdirs/training/NUG2012/ccm $SCRATCH
  - cd $SCRATCH/ccm
- Problems and feedback?
  - consult@nersc.gov
- Learn more at
  - http://www.nersc.gov/users/computational-systems/hopper/cluster-compatibility-mode/

Acknowledgement

- Gary Lowell for his technical support and help.
- Katie Antypas