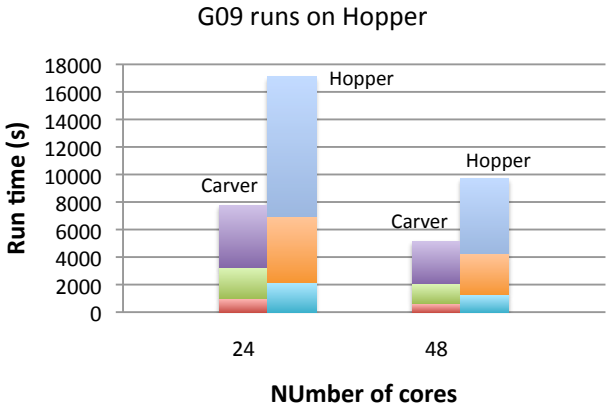


Cluster Compatibility Mode on Hopper

Zhengji Zhao, Helen He and Katie Antypas

NERSC

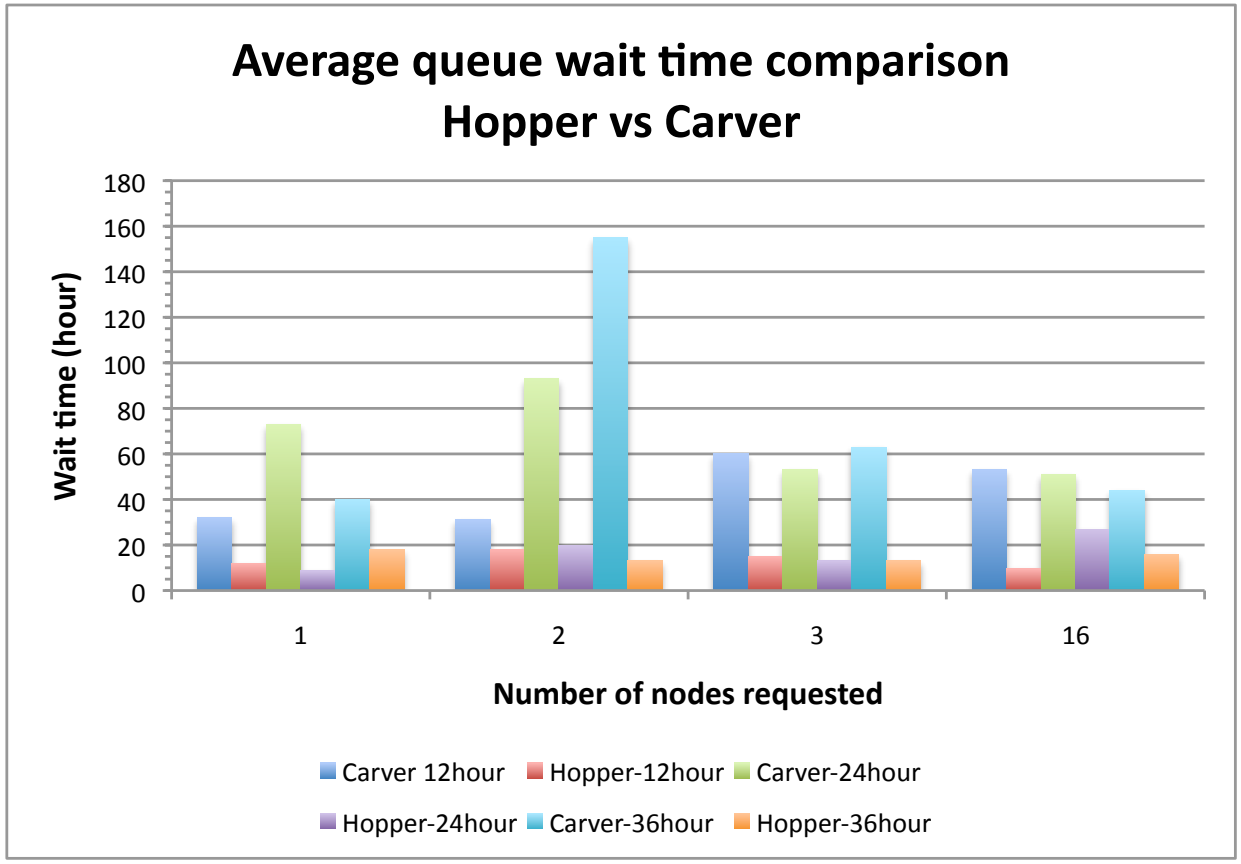
CUG meeting, May 1, 2012

U.S. DEPARTMENT OF ENERGY | Office of Science

NERSC National Energy Research Scientific Computing Center

BERKELEY LAB Lawrence Berkeley National Laboratory

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



Hopper - Cray XE6,
Two 12-core AMD 'MagnyCours' 2.1 GHz processors
Peak flops: 1.28 Petaflops
6384 nodes, **153,216 cores**
24 cores/node
32 GB memory per node
Gemini Interconnect

Carver – IBM iDataPlex
Intel Nehalem 2.67 GHz processors, 8 cores/node
Peak flops: 106.5 Tflops
1202 nodes, **9984 cores**
24GB memory per node
4X QDR InfiniBand

2 node jobs requesting 36 hours waited 4 times longer 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/

Why these users prefer to stay on Carver?

- Some applications don't run on Hopper due to the lack of TCP/IP support on compute nodes
  - Gaussian code, NAMD replica simulations
  - Wien2k
- Friendly environment for serial workload
- Faster processor
- Larger memory per core
- More queue and memory options
  - Huge memory node (1TB)
  - 3 weeks long queue
  - Serial queue
- Data analysis tools not available on Hopper
  - Matlab

Outline

- Introduction to CCM
- Applications that CCM enables
- Performance of CCM
- CCM utilization at NERSC
- Benefits and issues
- Conclusion

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 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 released to compute nodes pool at job exit

In CCM, applications run in a generic Linux cluster environment

Programming environment of CCM on Hopper

- Compilers
  - PGI, GNU, INTEL, PATHSCALE, CRAY
  - Default - dynamic linking
- Libraries
  - OpenMPI #built outside of the batch system
  - ScaLAPACK
  - ACML, FFTW and all serial/thread libraries for Hopper Extreme Scalability Mode (ESM)
  - MKL from Carver
- Debuggers and profilers
  - IPM; DDT
- Applications
  - G09 (ssh+OpenMP), NAMD (MPI + socket operations), WIEN2k (MPI+ssh)
  - VASP (MPI), Quantum Espresso (MPI+OpenMP)
  - Matlab, IDL

How to compile codes for CCM

- Compile somewhere else, any x86_64 platform
- Compile on Hopper login nodes or mom nodes
  - PGI, GNU, Intel, pathscale, Cray compilers are available
  - Use native compiler calls, eg., pgif90, pgcc, and pgCC
  - Or Use parallel compiler wrappers from OpenMPI, mpif90, mpicc and mpiCC

```
module load openmpi_ccm
mpicc xthi.c
```

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

```
nid00002:~/ccm> 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
nid00029:~> cd $PBS_O_WORKDIR
nid00029:~/ccm> module load ccm
nid00029:~/ccm> module load openmpi_ccm
nid00029:~/ccm> export CRAY_ROOTFS=DSL
nid00029:~/ccm> ccmrun mpirun –np 2 -bynode -hostfile $PBS_NODEFILE –prefix /usr/
common/usg/openmpi/default/pgi hostname
…..
nid00029:~/ccm> aprun -n 2 -N1 hostname
nid00019
nid00018
Application 252074 resources: utime ~0s, stime ~0s
```

ccmlogin -interactive access to compute nodes

Continued …

```
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

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

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

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

nid00019:~>  top
```

-V passes environment variables to head node

Now you can ssh to compute nodes under CCM!

CCM run time environment management

- The mpirun from OpenMPI does not pass the environment variables to the remote CCM nodes. To pass environment to remote CCM nodes
  - shell startup files, ~/.bashrc.ext, ~/.cshrc.ext
  - Use mpirun with the --prefix and –x options
  - env >~/.ssh/environment
- The file $PBS_NODEFILE contains the compute node list allocated for a CCM job
  - mpirun … –hostfile $PBS_NODEFILE …

Sample job script for MPI jobs

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

cd $PBS_O_WORKDIR
module load ccm
export CRAY_ROOTFS=DSL

#compile
Module load openmpi_ccm
mpicc xthi.c

#run
ccmrun mpirun -np 48 -hostfile $PBS_NODEFILE –prefix /usr/common/
usg/openmpi/default/pgi a.out
```

How to run OpenMP+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 -mp xthi.c
export OMP_NUM_THREADS=6

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

Process affinity:

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

Gaussian 09 job script

```
#PBS -S /bin/csh
#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

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

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

In ~/.cshrc.ext file, add
module load g09

```
% cat g09l
#!/bin/csh
setenv GAUSS_EXEDIR /usr/common/usg/g09/c1/g09/linda-exe:$GAUSS_EXEDIR
set nodelist="""`cat $PBS_NODEFILE | sort -u`"""”
setenv GAUSS_LFLAGS "-vv +getload +kaon -delay 500 -wait 1200 -workerwait 1800 –mp 24 -nodelist $nodelist"
setenv GAUSS_SCRDIR `pwd`
g09 $argv
```

Sample job script to run multiple serial Jobs on a single node

```
#!/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 multiple_serial_jobs.sh
```

```
% cat multiple_serial_jobs.sh
./a1.out &
./a2.out &
…
./a24.out &
wait
```

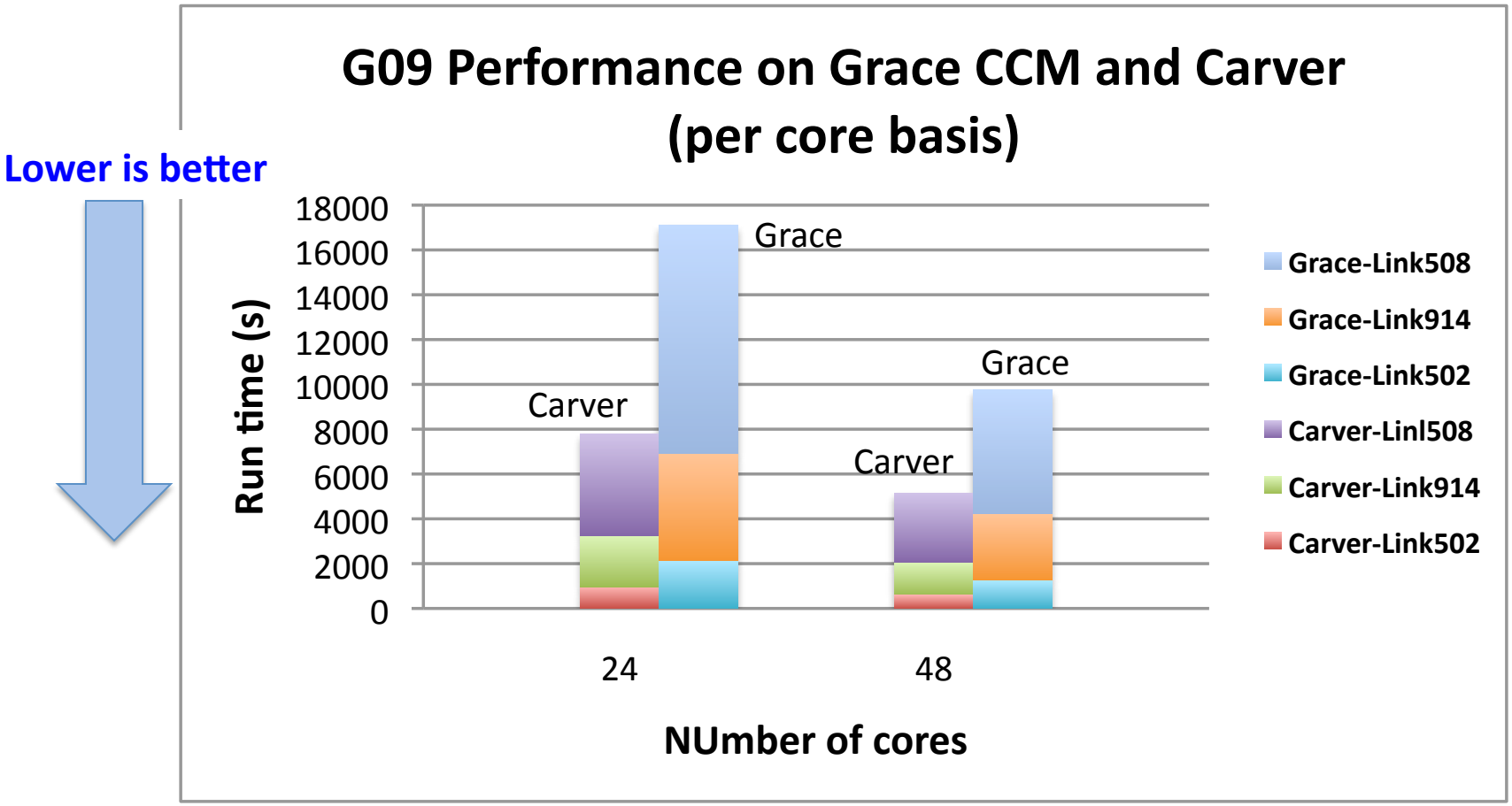
```
OR: Interactively run
% qsub –I –V -q ccm_int -l
mppwidth=24 –l walltime=00:30:00
…
% cd $PBS_O_WORKDIR

% module load ccm
% export CRAY_ROOTFS=DSL
% ccmlogin –V
…
% cd $PBS_O_WORKDIR
% ./a1.out &
%  ./a2.out &
   …
% ./a24.out &
```

NOTE: ccmrun wrappers the aprun command, so no multiple ccmrun commands can be launched on a single node.

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.
- G09 were run under CCM1 on Hopper test machine, Grace

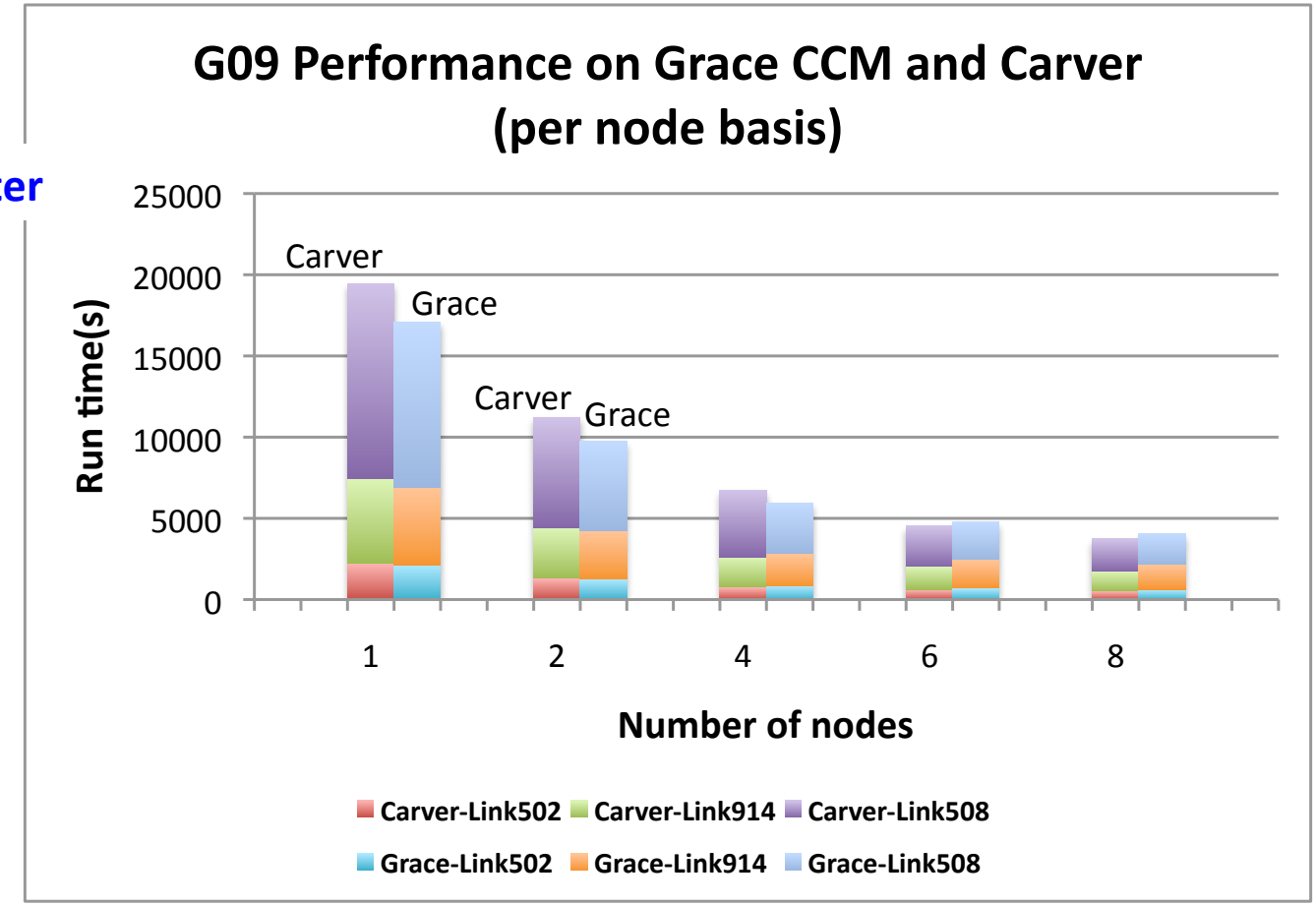
CCM enables G09 jobs on Hopper



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.

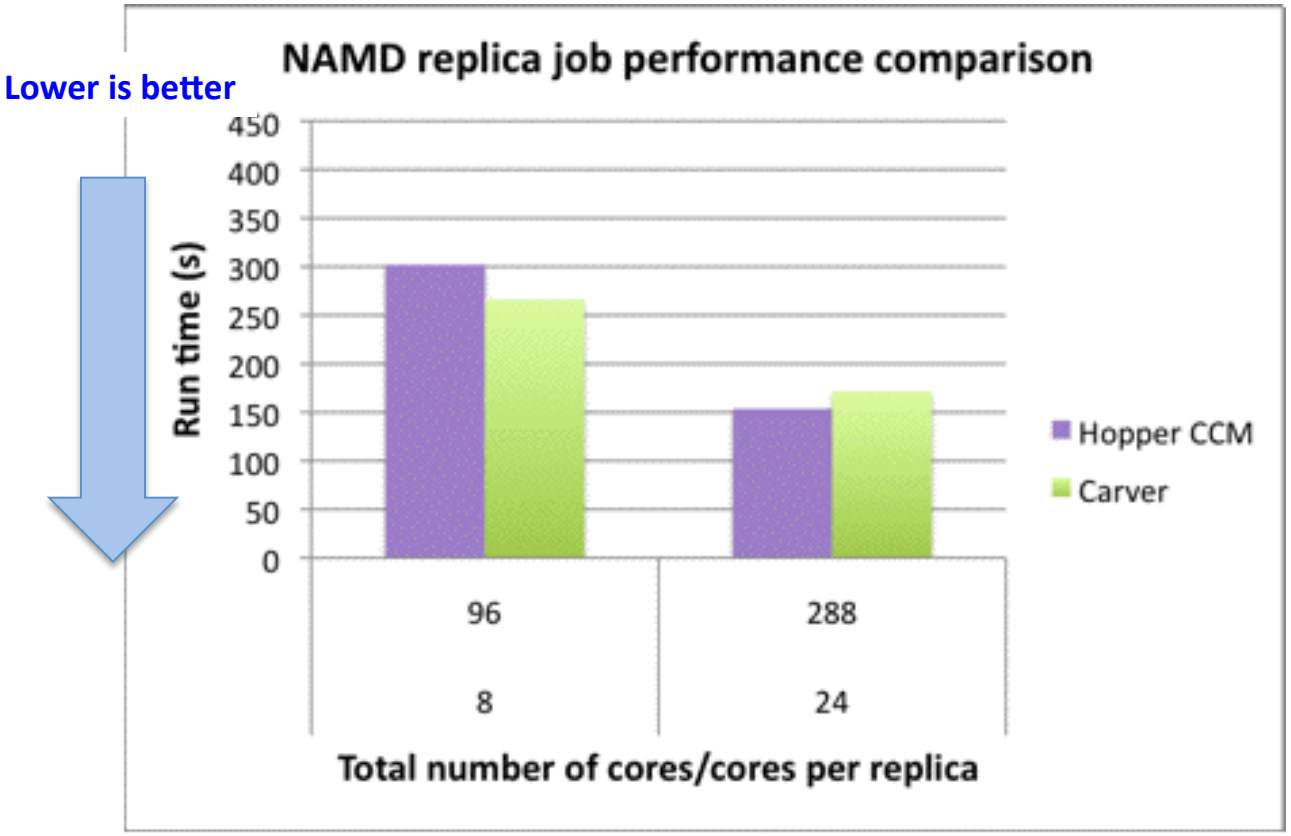
On a node basis G09 performance is more comparable, though memory affinity issues lower performance on Grace

Lower is better



1. All g09 jobs ran with 24 threads with a single task on the node.
2. There is no good way to control the process/memory affinity through ssh, so the OpenMP threads don't run at the optimal task/thread ratio and placement on the Grace nodes.

CCM enables NAMD replica jobs on Hopper



NAMD replica job:

1. Runs multiple job instances (replicas) simultaneously
2. Message passing within a replica uses MPI;
3. Communication between replicas uses **socket** operations

Test case (user case):

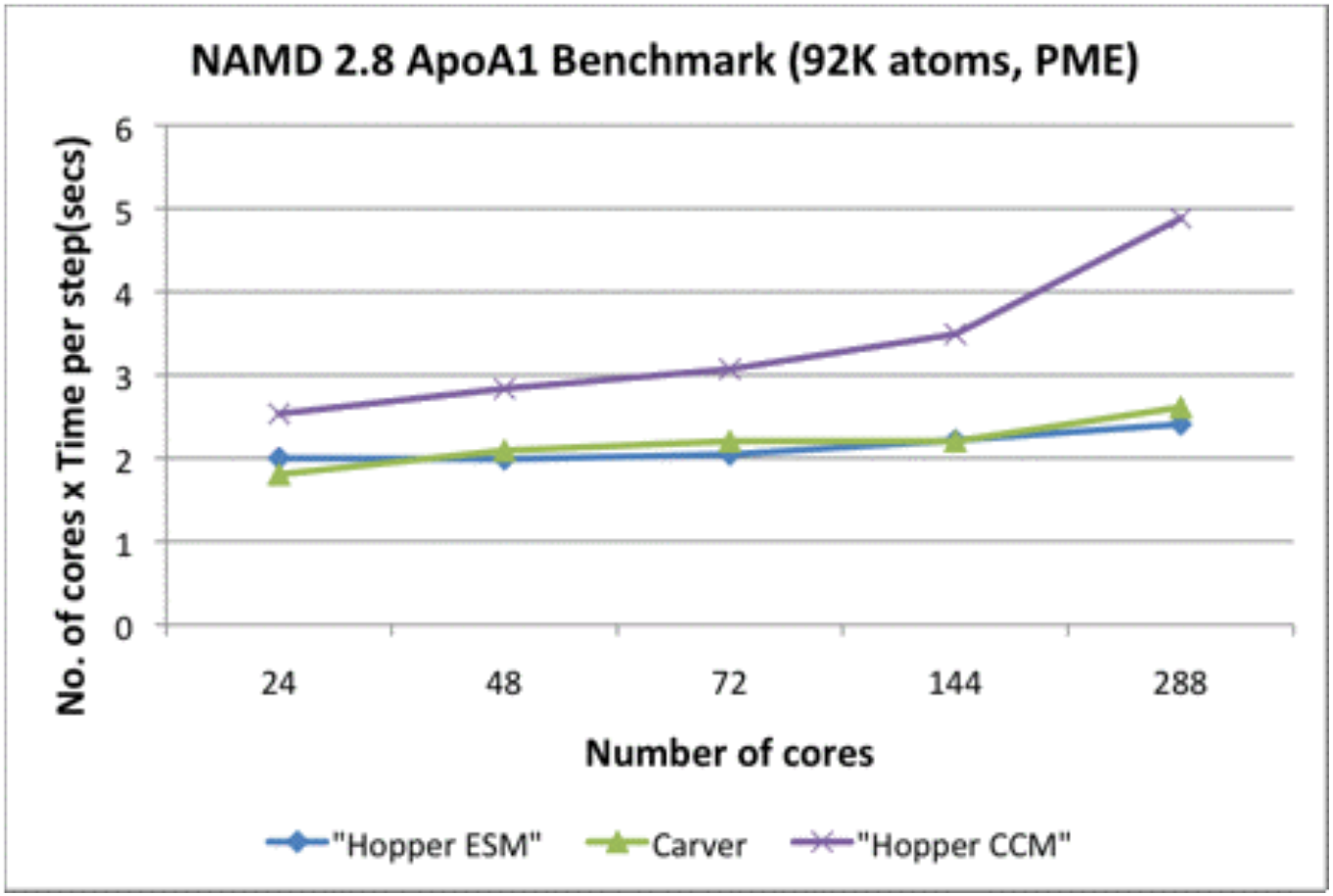
1. 12 replicas simulated at the same time for a system with 95K atoms
2. The runtime is for the first 1,000 MD steps.

1. NAMD replica job under Hopper CCM runs around 14% slower than on Carver when 8 cores used per replica; but 10% faster than Carver when 24 cores used per replica
2. Slower file system on Carver may account for its slowdown at 288 cores

Pure MPI NAMD parallel scaling

Lower is better

Flatter is better



Pure MPI NAMD (OpenMPI + fftw/2.1.5) scales well up to 144 cores (for this 92K system) under Hopper CCM, but scaling drops significantly at 288 cores.

Wien2k runs on Hopper under CCM

Wien2k:

Parallel implementation:

- 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 (user case):

- 21 k-points
- For each k-point used 8 cores at 84 core run; 12 cores at 252 core run

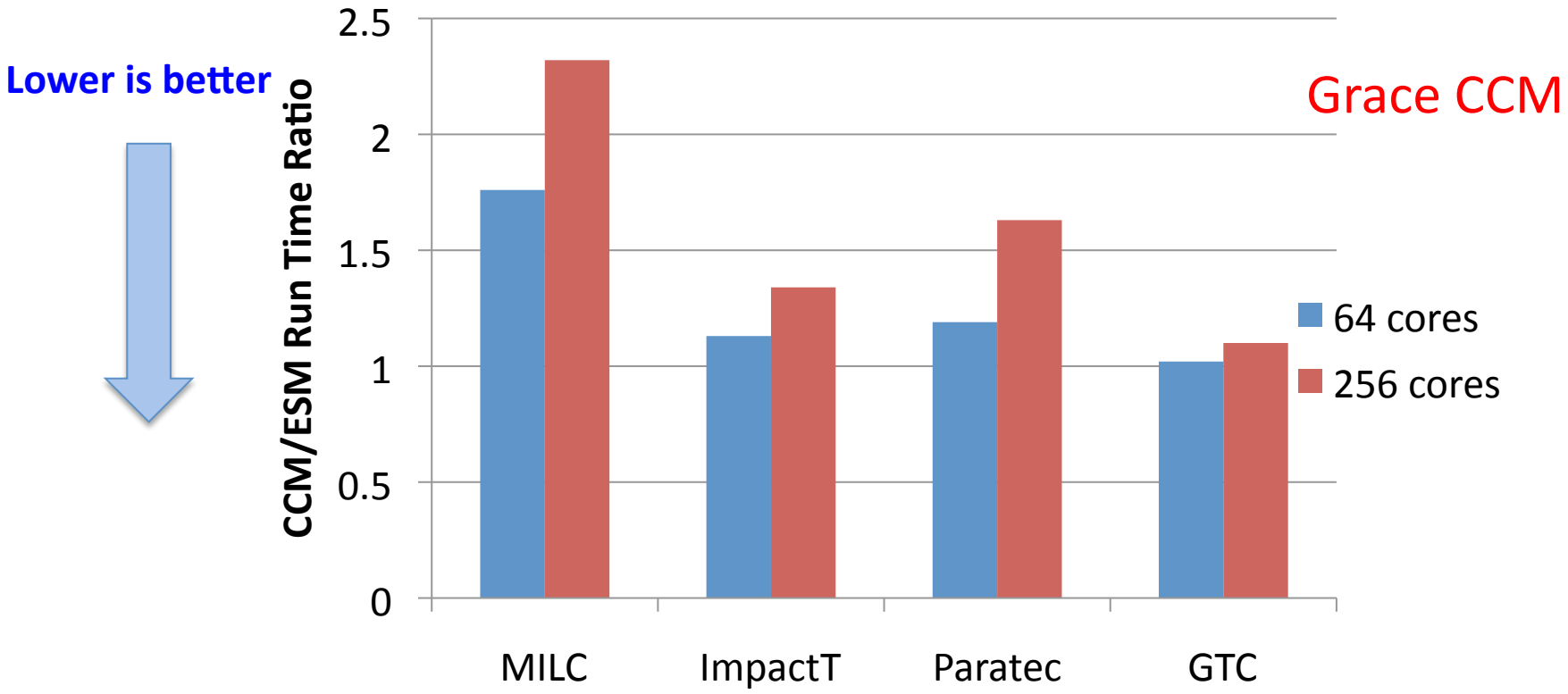
WIEN2k runs slower under Hopper CCM than on Carver by ~30% at 252 core run; At 84 core run, it runs slower under CCM than on Carver by around 90%.

N6 Application Benchmarks

Benchmark	Science Area	Algorithm	Compiler Used	Concurrency Tested	Libraries
MILC	Lattice Gauge Physics	Conjugate Gradient, sparse matrix, FFT	GNU	64, 256, 512, 1024	
ImpactT	Accelerator Physics	PIC, FFT	PGI	64, 256	FFTW
Paratec	Material Science	DFT, FFT, BLAS3	PGI/Intel	64, 256	Scalapack FFTW
GTC	Fusion	PIC, Finite Difference	PGI	64, 256, 512, 1024	

- 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).

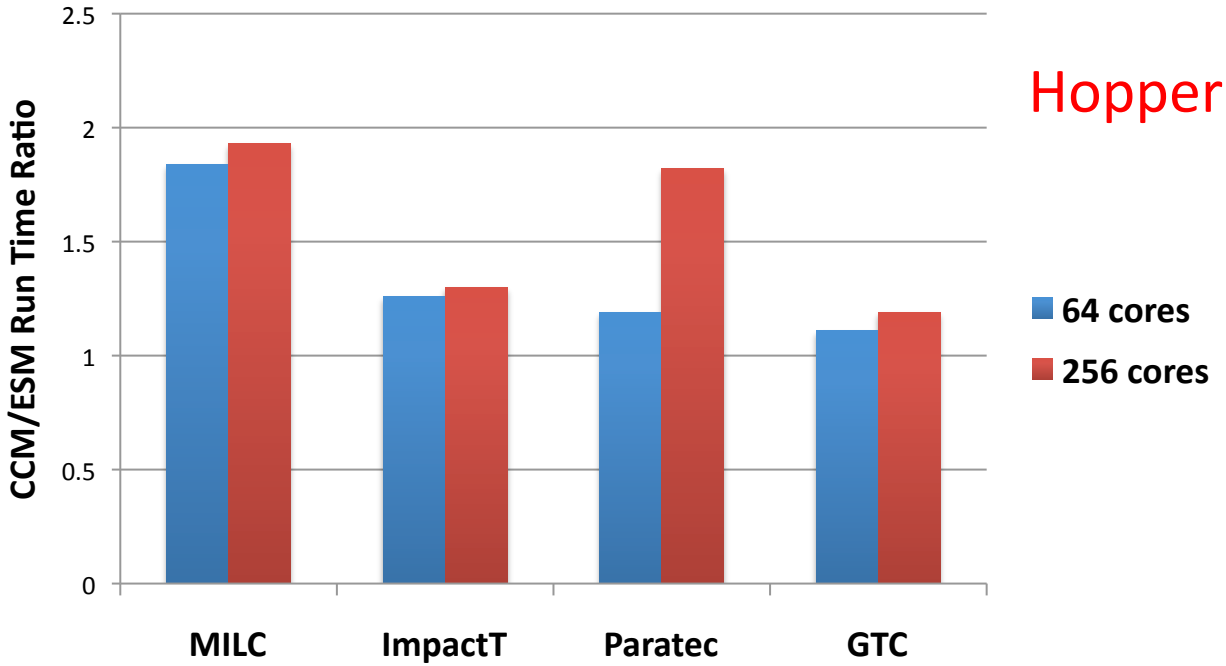
N6 Benchmarks Run Time Comparison between CCM (without IAA) and ESM



1. CCM slows down more with 256 cores than with 64 cores.
2. CCM/ESM run time ratio is between 1.02 times with GTC 64 cores to 2.4 times with MILC 256 cores.

N6 Benchmarks Run Time Comparison between CCM and ESM

Lower is better

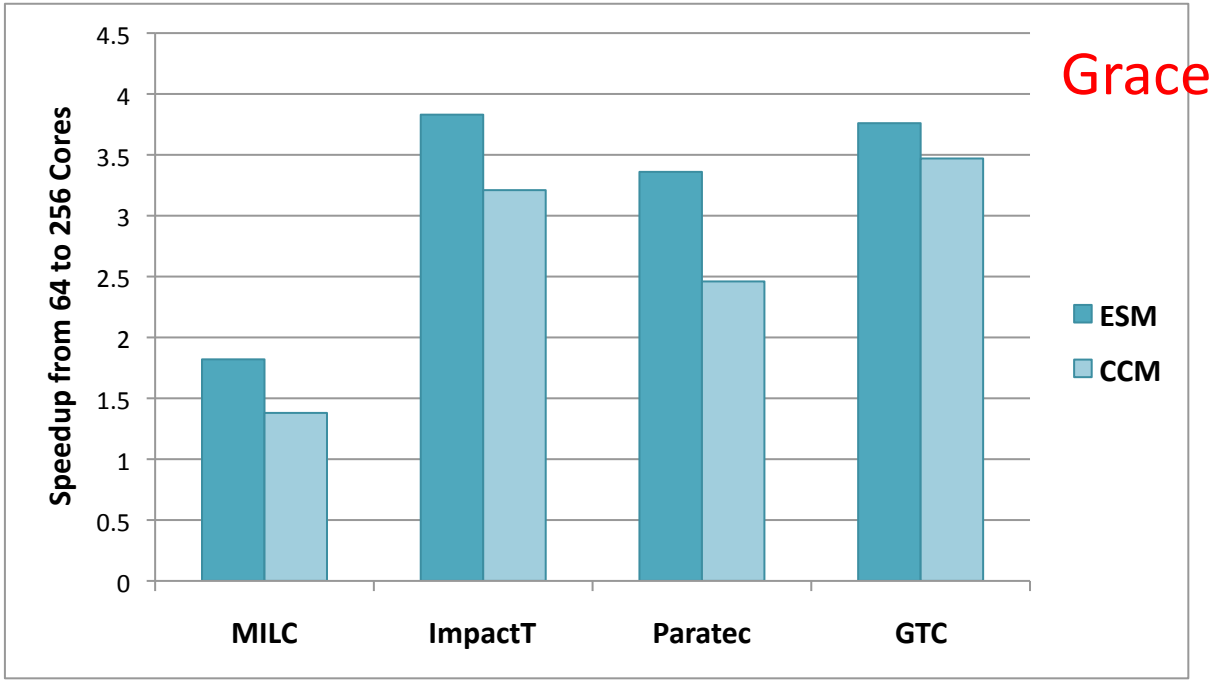


1. CCM slows down more with 256 cores than with 64 cores.
2. CCM/ESM run time ratio is between 1.1 times with GTC 64 cores to 1.9 times with MILC 256 cores.

* Paratec 256 CCM run used Intel compiler. Other Paratec runs used PGI compiler.
* ImpactT failed to run at 256 cores with CCM, so Grace data was used for this data point
* Hopper CCM utilizes IAA the OFED interconnect protocol over Gemini HSN

ESM and CCM (without IAA) Scaling Comparison

Higher is better

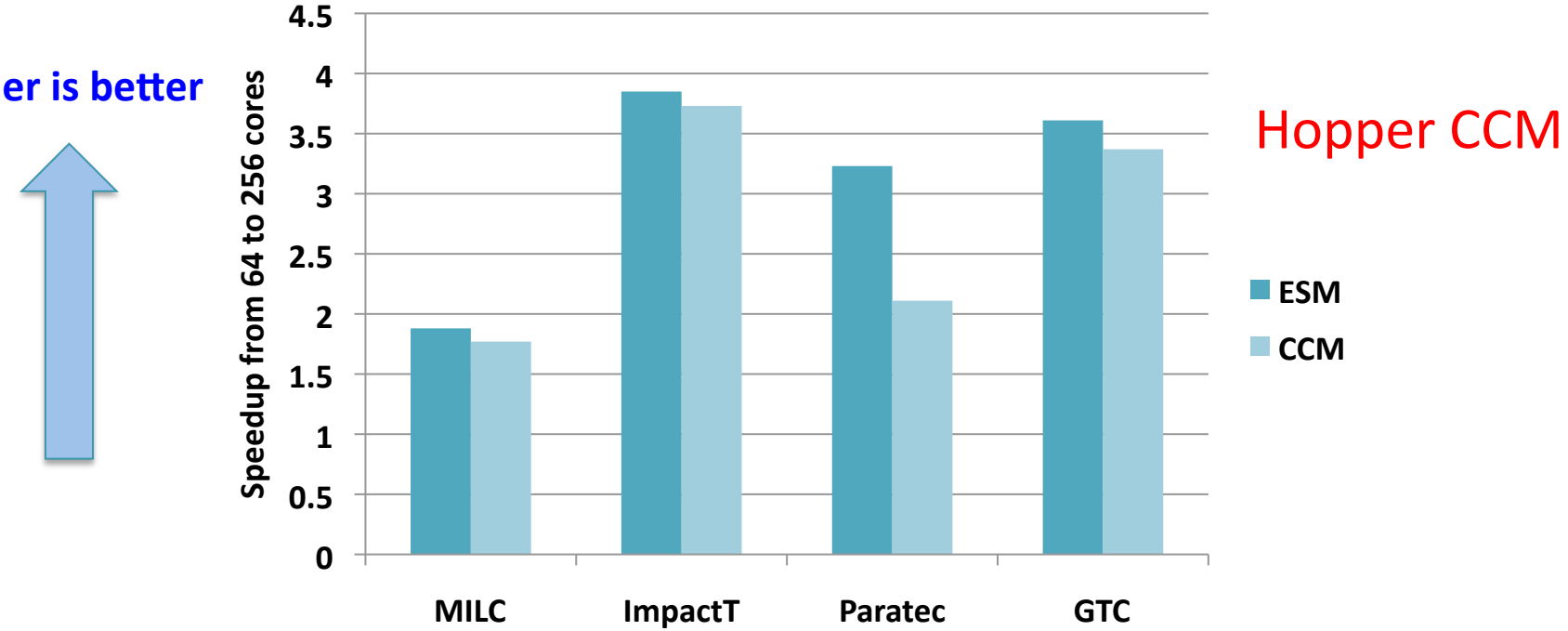


Grace CCM

1. ESM speedup from 64 to 256 cores ranges from 1.82 to 3.83.
2. CCM speedup from 64 to 256 cores ranges from 1.38 to 3.47.
3. ESM has better speedup than CCM. CCM is not a lot worse.

ESM and CCM Scaling Comparison

Higher is better



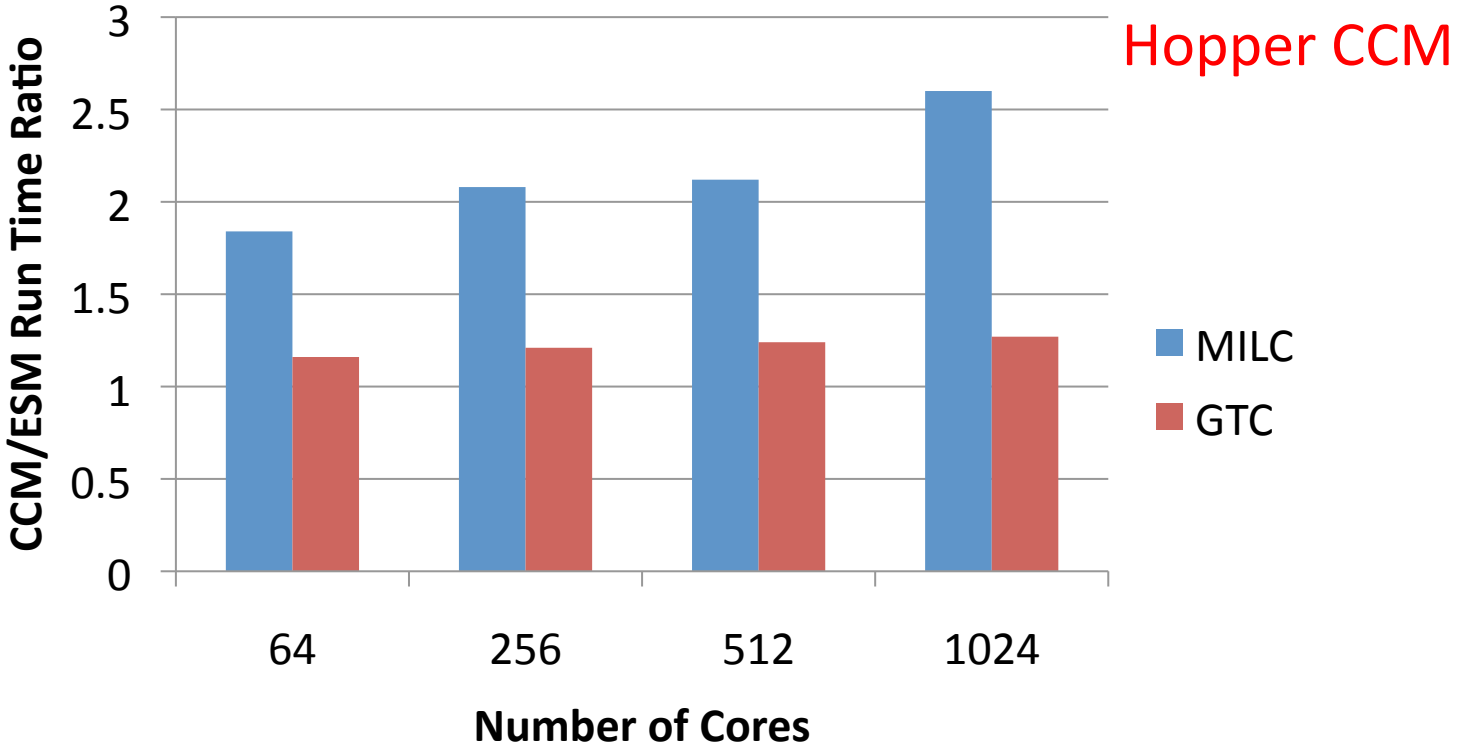
Hopper CCM

1. ESM speedup from 64 to 256 cores ranges from 1.88 to 3.85.
2. CCM speedup from 64 to 256 cores ranges from 1.77 to 3.73.
3. ESM has better speedup than CCM. CCM is not a lot worse.

* Paratec 256 CCM runs used intel compiler, while other Paratec runs used PGI compiler
* ImpactT failed to run at 256 cores with CCM, Grace data was used for this data point

ESM and CCM Run Time Comparison with Larger Runs

Lower is better

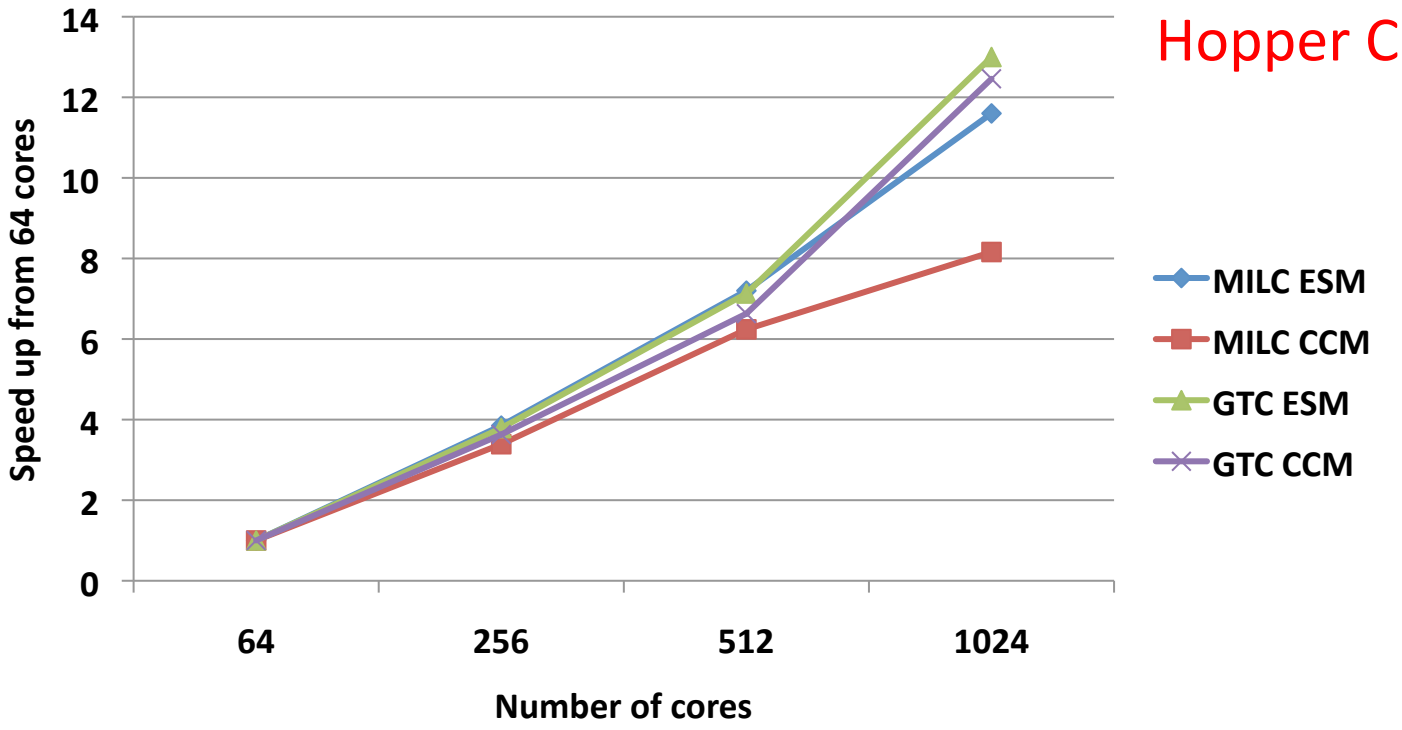


1. MILC CCM/ESM run time ratio ranges from 1.8 (64 cores) to 2.6 (1024 cores).
2. GTC CCM/ESM run time ratio ranges from 1.2 (64 cores) to 1.27 (1024 cores).

* A larger data set for MILC is used
* GTC uses weak scaling.

ESM and CCM Scaling Comparison with Larger Runs

Higher is better

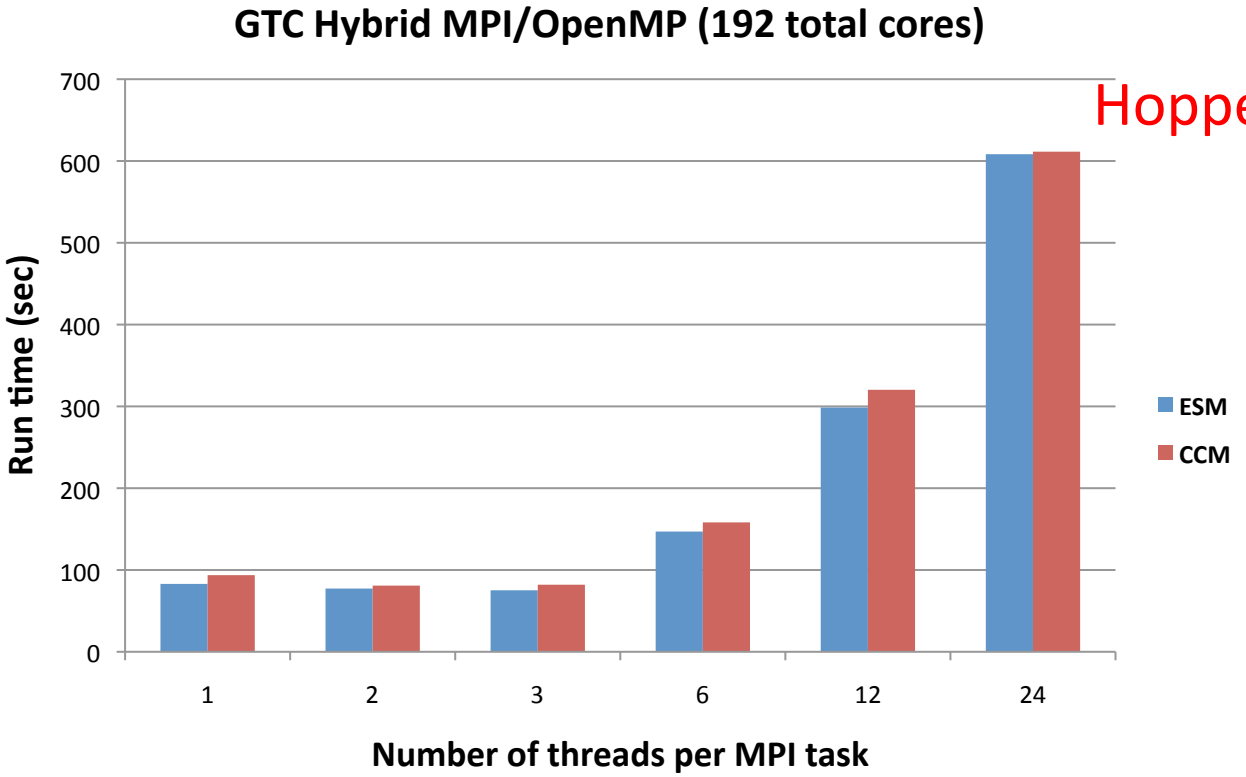


1. MILC speedup from 64 to 1024 cores: ESM 11.6, CCM 8.2. Ideal is 16.
2. GTC speedup from 64 to 256: ESM 13, CCM 12.5. Ideal is 16.

* A larger data set for MILC is used.
* GTC uses weak scaling.

GTC hybrid MPI/OpenMP runs

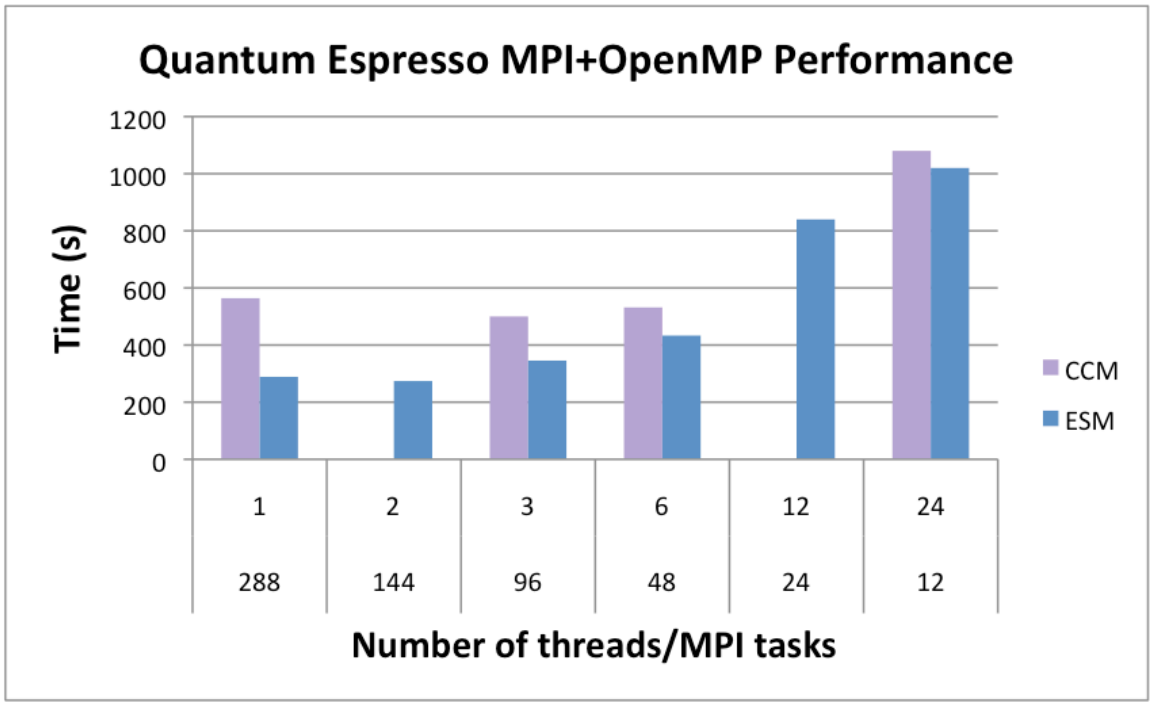
Lower is better



Hopper CCM

1. Both ESM and CCM have a sweet spot at 3 OpenMP threads per MPI task.
2. CCM results are almost identical with ESM results.
3. Actual run time with CCM ranges from 1.01 times (24 threads) to 1.11 times (1 thread) as of ESM.

Quantum Espresso MPI+OpenMP runs



Test case AUSURF112:
112 Au atoms
2 k-points.

- Hybrid QE runs fastest at thread=3 under CCM (among the successful runs) while it runs fastest at threads=2 under ESM.
- Hybrid QE runs 6 % slower under CCM than on ESM at thread 24 90% slower at threads 1.
- Hybrid QE fails at threads=2 and threads=12 (seg fault, hang)

How is CCM utilized at NERSC

- Announced G09 to users on 2/15/2012, contacted individual users to try out CCM.
  - 60 non-staff users have tried g09 on Hopper, only a few users stayed (5-8 users)
  - Performance was the main concern
  - 48 hours of wall limit was not long enough
- From 2/15 to 4/11/2012, 13,508 CCM jobs were run, and used 1.36 million MPP hours.
- A few use cases other than g09
  - Complicated workload management (1 user, user code + multiple g09 +script)
  - NAMD replica jobs (3 users up to 720 cores)
  - Serial workload (3 users) - running serial jobs over multiple nodes using ssh- filed bug for OOM and job hang at exit

How is CCM utilized at NERSC --continued

- Queue policy changes to promote the CCM usage
  - Increased wall limit and raised queue priority to compensate the slower performance and to allow shorter queue wait time
  - Created the ccm_int queue with the same debug/interactive queue priority for interactive CCM workloads
- Announced CCM availability to all NERSC users 4/10/2012
- Expected usage
  - More g09 users (due to the queue configuration changes)
  - More NAMD replica and WIEN2k users
  - Matlab and IDL usage- interactive data analysis (local data, exclusive memory access (more license seats purchased due to the increasing user demand)
  - Serial workloads
  - MPMD that requires sharing a single node between job instances
  - Other TCP/IP workload

Current issues with CCM

- Performance issues
  - IAA improves performance, but not to the extent of user satisfaction
  - Process/memory binding issue impacts performance (g09).
  - Cray shared root file system is slow, extra delay at job start and exit
  - Jobs often run into various errors or hang
- Cray provides execution environment rather than development environment
  - Optimized ESM features don't work under CCM
  - Users need to deal with the runtime environment manually due to the job launching mechanism of Torque is not supported by Cray.
  - CCM environment is not as friendly and complete as ESM environment

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, and enables serial workloads.
- CCM helps the scientific productivity of our g09 users and other users who need TCP/IP services in their workflow by utilizing a shorter queue turnaround on Hopper.
- CCM dynamic queue configuration fits in hopper existing workload seamlessly, while it can accommodate as large as possible CCM workload, it also avoids resource waste when CCM demand is low

Conclusion --continued

- Hopper N6 application benchmark suite run 10% -90% slower in CCM than in ESM at 256 cores.
- Additional work needed to make CCM a more pleasant user environment with better performance.
  - Better scaling, Cray's goal is to scale up to 2048 PE's per job
  - Performance improvement for a fixed core job
  - Provide modules for CCM environment, eg., PrgEnv-pgi-ccm
  - Resolve various runtime issues, job hang, libibgni error, OOM, segmentation fault
  - More robust X11 support

Users are encouraged to try out CCM on Hopper

Acknowledgement

- Gary Lowell, Randell Palmer, and Tara Fly for their technical support and help.
- Tina Butler, Nick Wright, Jay Srinivasan, Jack Deslippe and other NERSC staff for their support and useful discussions
- NERSC users who provided the test cases, Guoping Zhang, Guoxiong Su and Jun-Wei Luo.