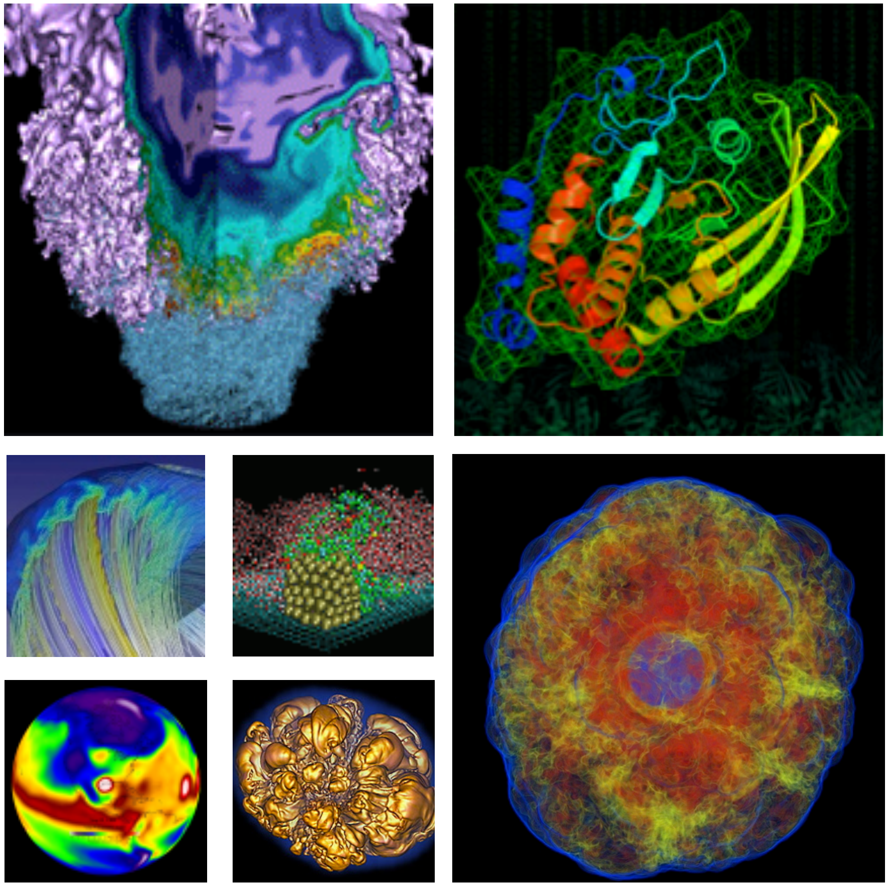
Effects of Hyper-Threading on the NERSC workload on Edison



Zhengji Zhao, Nicholas Wright, and Katie Antypas
NERSC

CUG meeting, May 8, 2013

Edison, Cray XC30, is NERSC's next Petascale machine

Edison will be delivered in two phases

Edison Phase 1 system

Phase 1 system -delivered

664 nodes dual-socket nodes; 8 core 2.6GHz Sandy Bridge processors , 16 cores per node; 64GB memory per node with 1600B/s DDR3;
L1 cache: 32KB; L2 cache: 256KB
L3 cache: 20MB

Aries interconnect with Dragon fly topology

Phase 2 system - coming:

Peak 2.4PFlops, SSP 214;
Ivy Bridge processors, more than 100K cores.

Hyper-Threading is available on Edison.

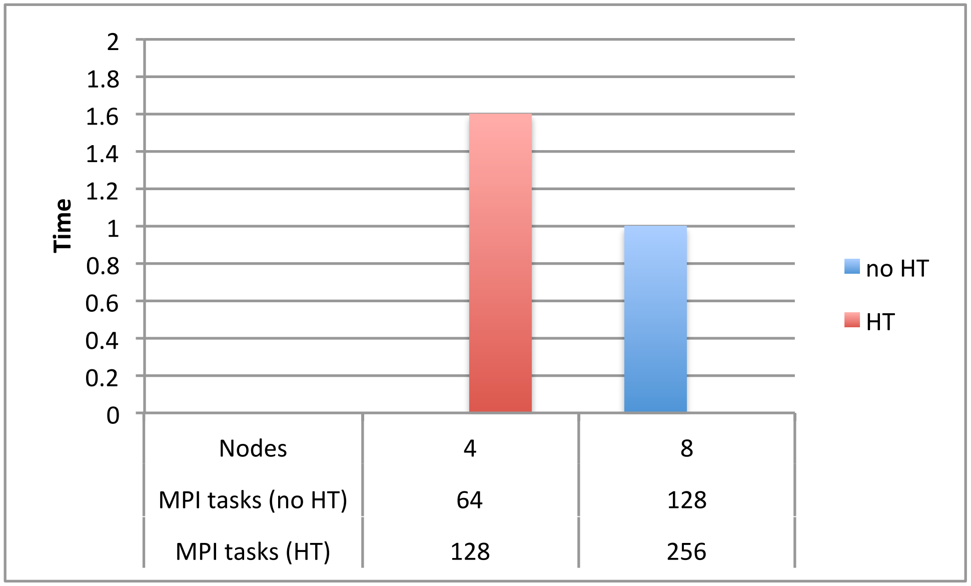
What is Hyper-Threading (HT)?

- HT is Intel's term for its simultaneous multithreading implementation. HT makes each physical processor core appears as two logical cores, which share the physical execution resources.
- HT increases the processor resource utilization by allowing OS to schedule two tasks/threads simultaneously, so that when an interruption, such as cache miss, branch misprediction, or data dependency, occurs with one task/ thread execution, the processor executes another scheduled task/thread.
- According to Intel the first implementation only used 5% more die area than the comparable non-hyperthreaded processor, but the performance was 15–30% better.

How HT works

- **Share resources**
- **Partition, recombine, and scheduling algorithms to make the resource sharing optimal and possible.**
- **OS should also be optimized to avoid consuming the resources if running single stream per physical core**

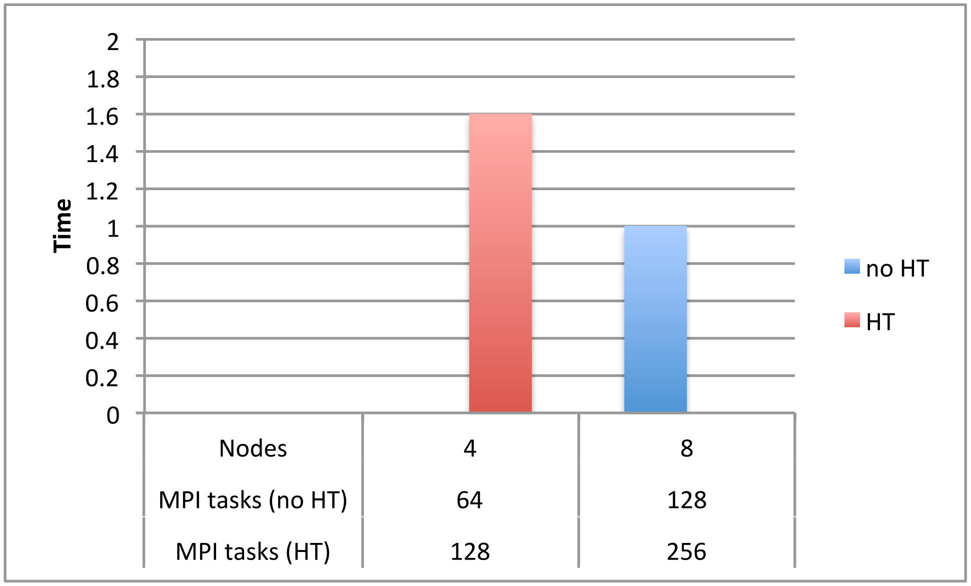
What performance gain we can expect from using HT when running over multiple nodes?



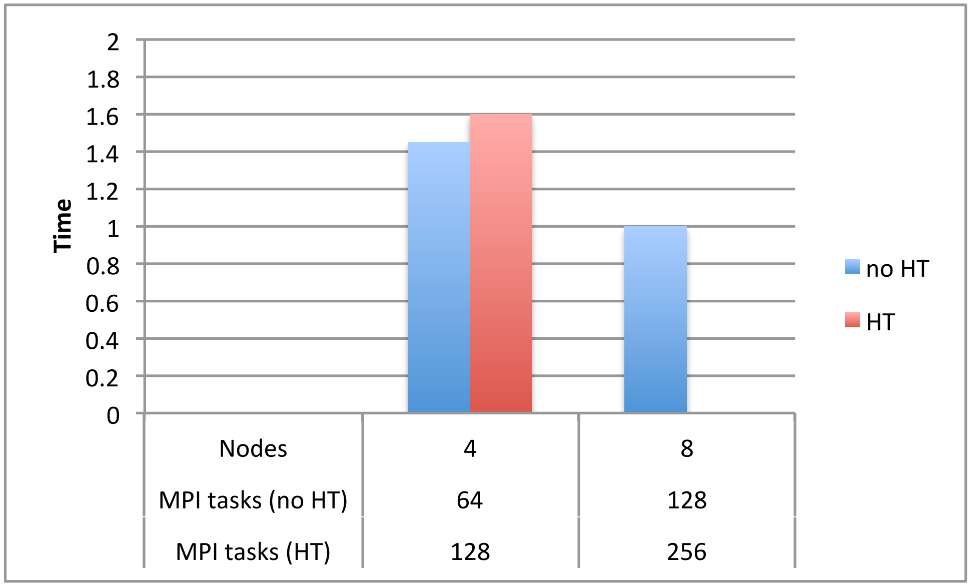
Nodes	4	8
MPI tasks (no HT)	64	128
MPI tasks (HT)	128	256

As long as Time (HT) < 2 x Time (no HT), HT increases throughput. Users get charged less

What performance gain we can expect from using HT when running over multiple nodes?

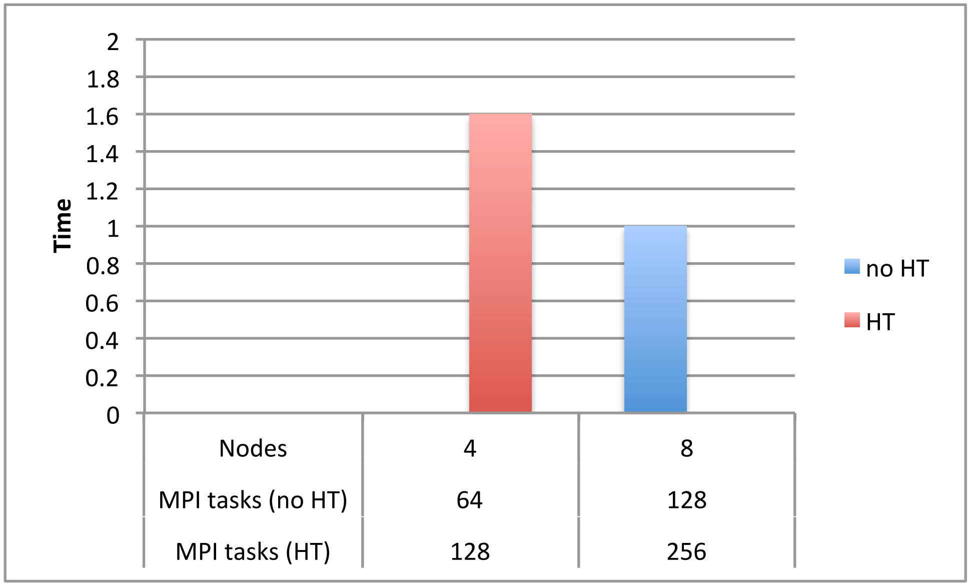


As long as Time (HT) < 2 x Time (no HT), HT increases throughput. Users get charged less

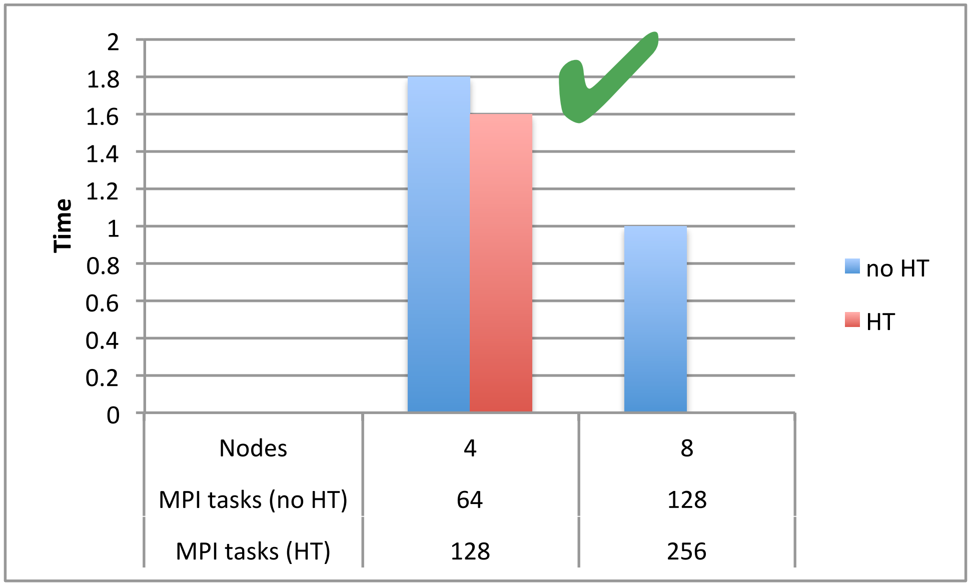


There is no point to run with HT in this scenario.

What performance gain we can expect from using HT when running over multiple nodes?



As long as Time (HT) < 2 x Time (no HT), HT increases throughput. Users get charged less



1) Using the same number of MPI tasks : Time (HT) < 2 x Time (no HT)
2) Using the same number of nodes: Time (HT) < Time (no HT)

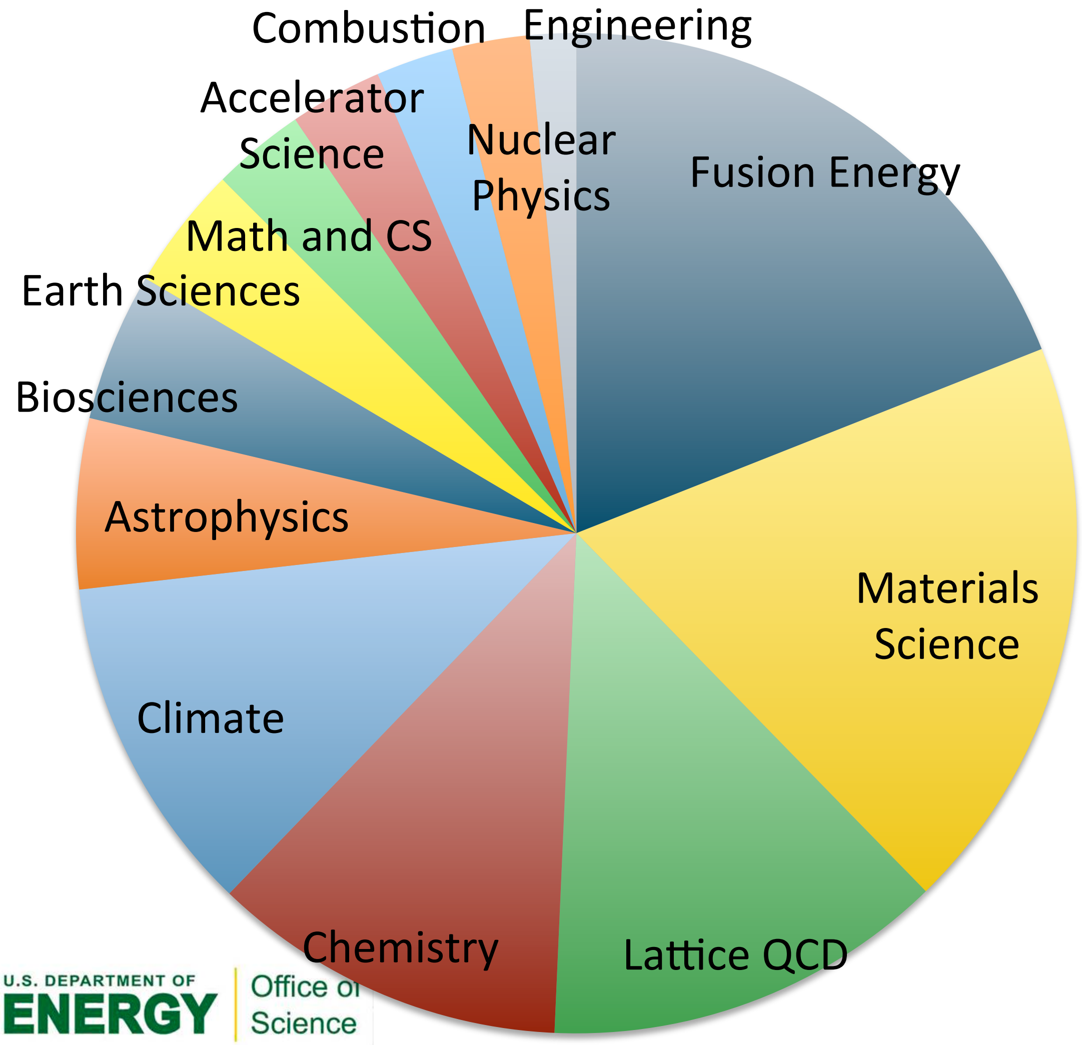
For users, real performance gain from HT should be measured by the runtime at the same node counts, but with doubled MPI tasks with HT.

Why is HT interesting to NERSC?

- Edison processors will be Sandy Bridge/Ivy Bridge where HT is a supported feature.
- HT could allow greater throughput for the entire NERSC workload and bring performance benefit for user applications.
- Users want to do more calculations within the given allocation hours.

NERSC serves a broad range of science disciplines for the DOE Office of Science

2012 Allocation Breakdown



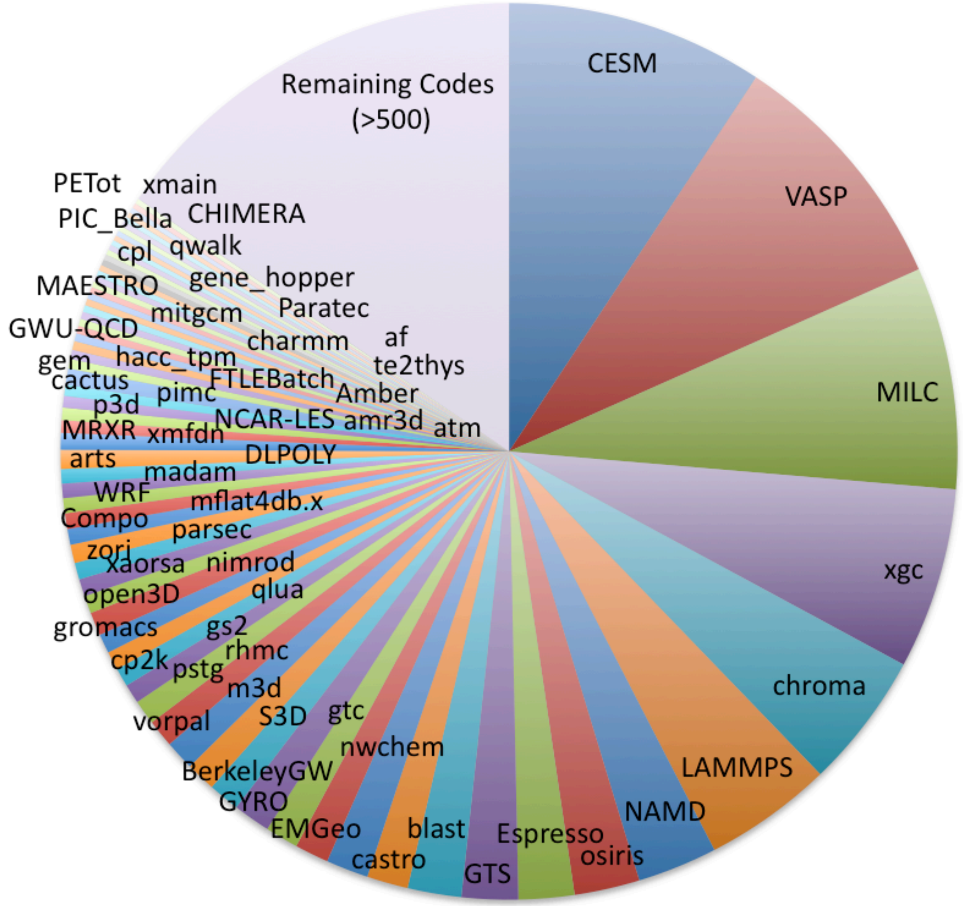
NERSC serves:

- **Over 4500 users**
- **Over 650 projects**

Over 650 applications run on NERSC resources

Top Application Codes on Hopper by Hours Used

Jan – Nov 2012



- 10 codes make up 50% of workload
- 25 codes make up 66% of workload
- 75 codes make up 85% of workload
- remaining codes make up bottom 15% of workload

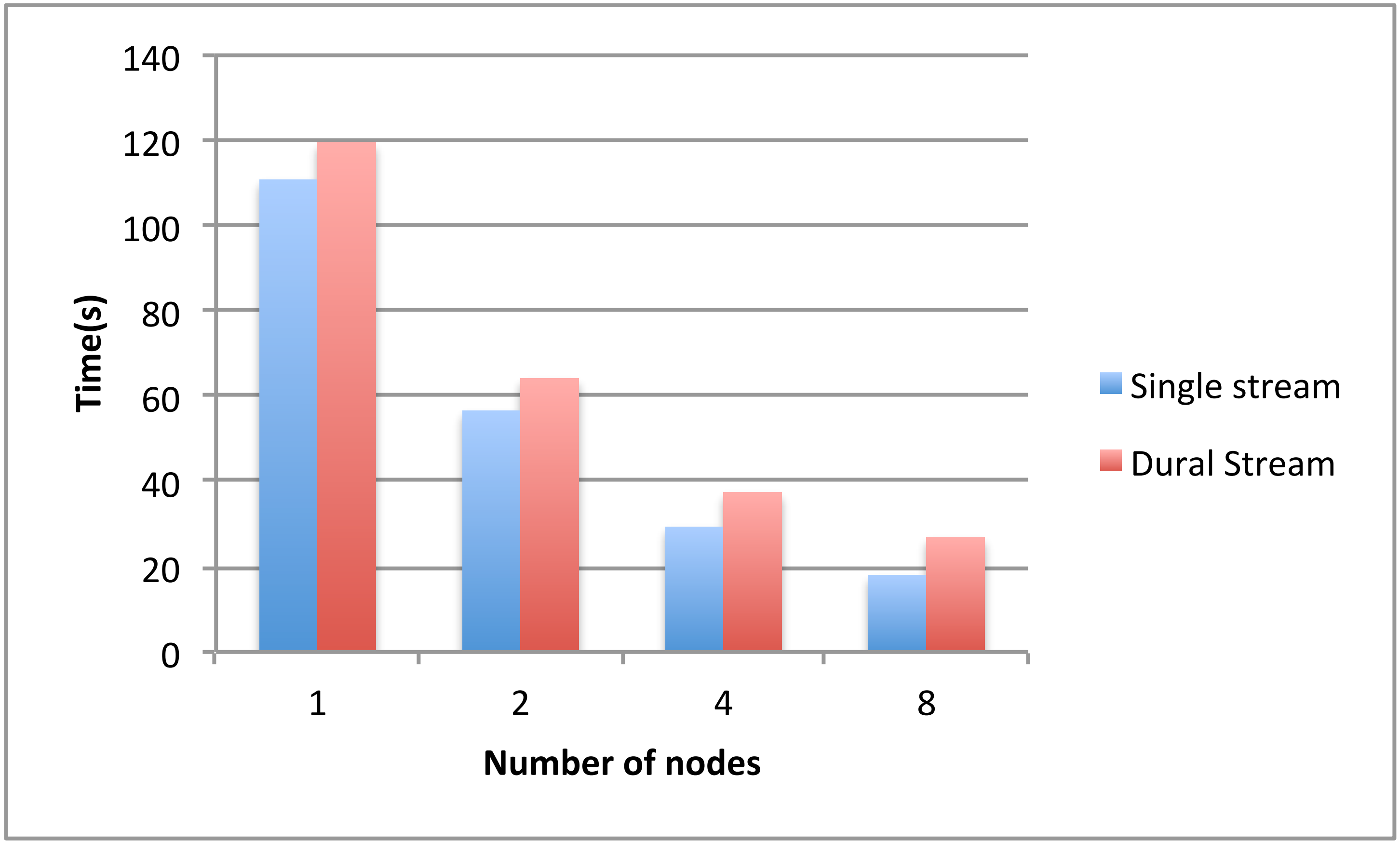
Code selections

Codes	Descriptions	Programming languages and programming models	Libraries used	Rank
VASP	Density Functional Theory	Fortran, C MPI	MPI, MKL, FFTW3	2
NAMD	Molecular Dynamics	C++ Charm++ (MPI)	Charm++, FFTW2	7
QE	Density Functional Theory	Fortran, C; MPI, OpenMP	MPI, MKL, FFTW3	9
NWChem	Chemistry	Fortran, C GA, MPI, ARMCI	MKL, GA	13
GTC	Fusion Plasma code (PIC)	MPI, OpenMP	MPI	15 (7)

Our methods and tests

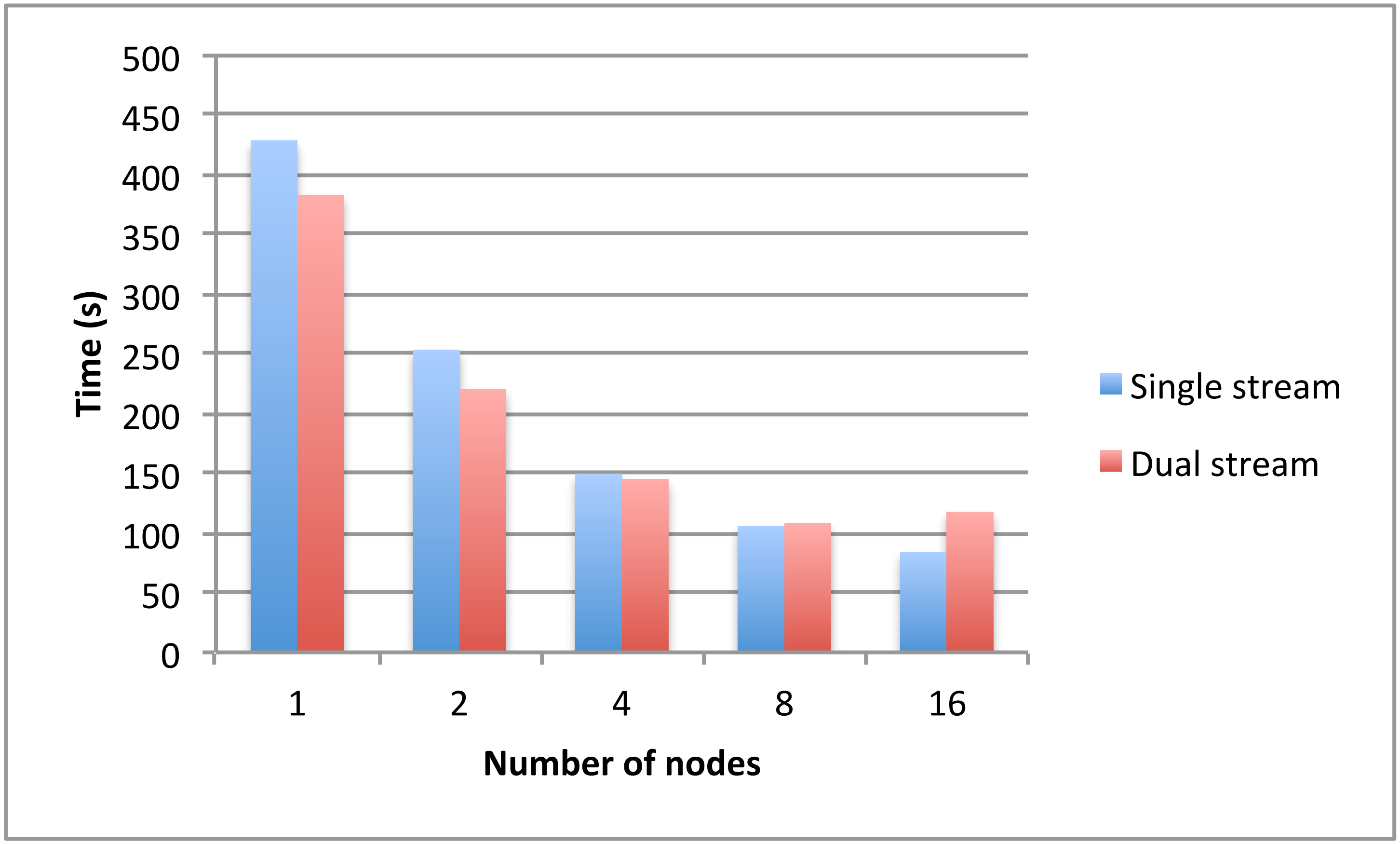
- We compared the same node performance instead of same core performance which is a more appropriate measure of HT effect for user applications, meaning running 2 times of MPI tasks for the HT runs.
- Run each application with selected test case(s) multiple times, and recorded total run time.
- Use the NERSC profiling tool IPM (with PAPI) to measure hardware performance events.
- Sandy Bridge HT enabled system, since the limited number of hardware counters available, the accurate measurement of the Floating point operations was not possible. So we measured the total instructions completed, and reported the cycles used per instruction retired (CPI). We also measured the L3 cache misses, TLB data misses, branch mis-predictions, etc., to see possible stalls occurred in the codes through the different runs (3 events each run).

VASP runtime with HT and without HT



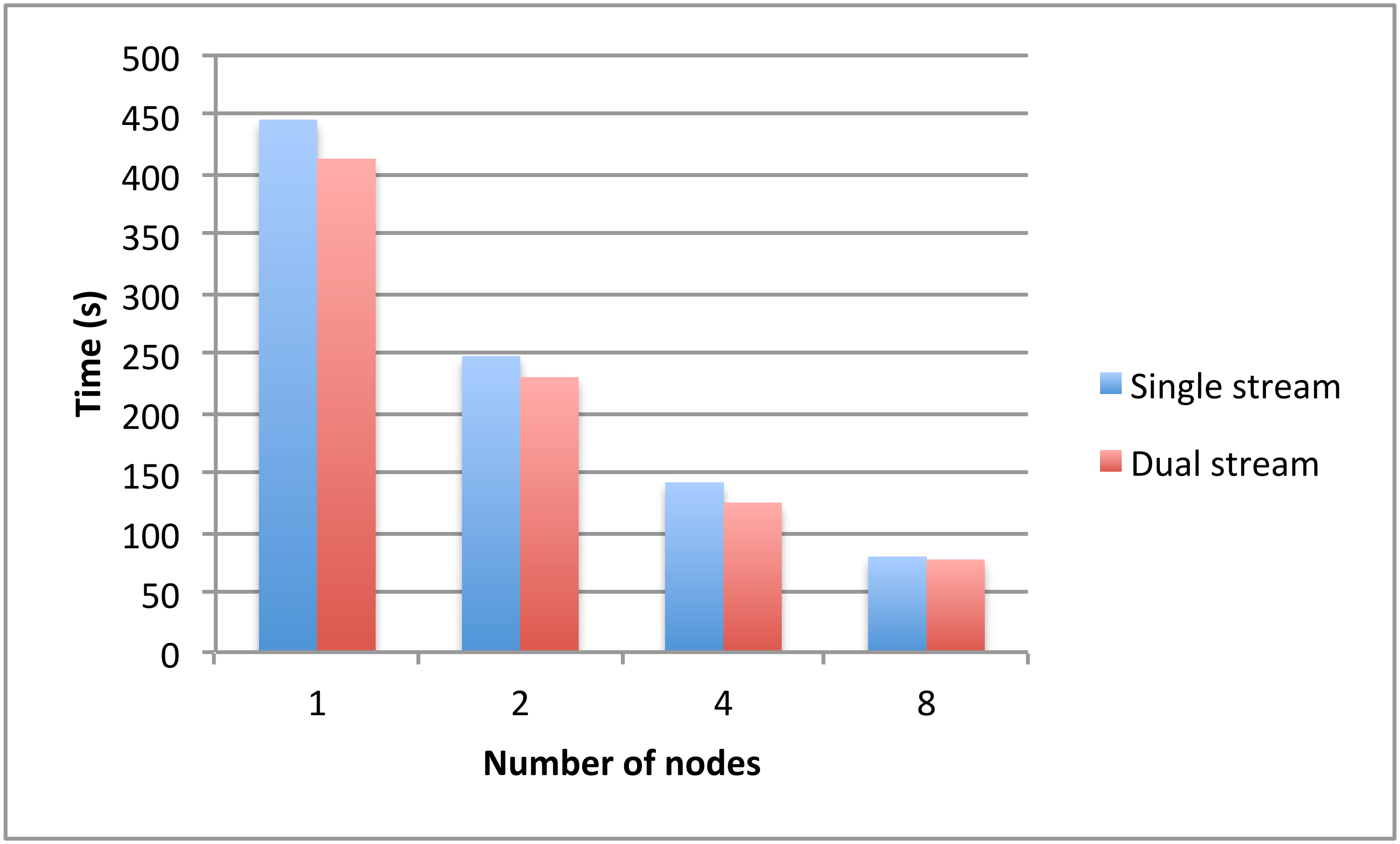
The HT does not help VASP performance at all node counts. The slowdown is 10-50%.

NAMD runtime with HT and without HT



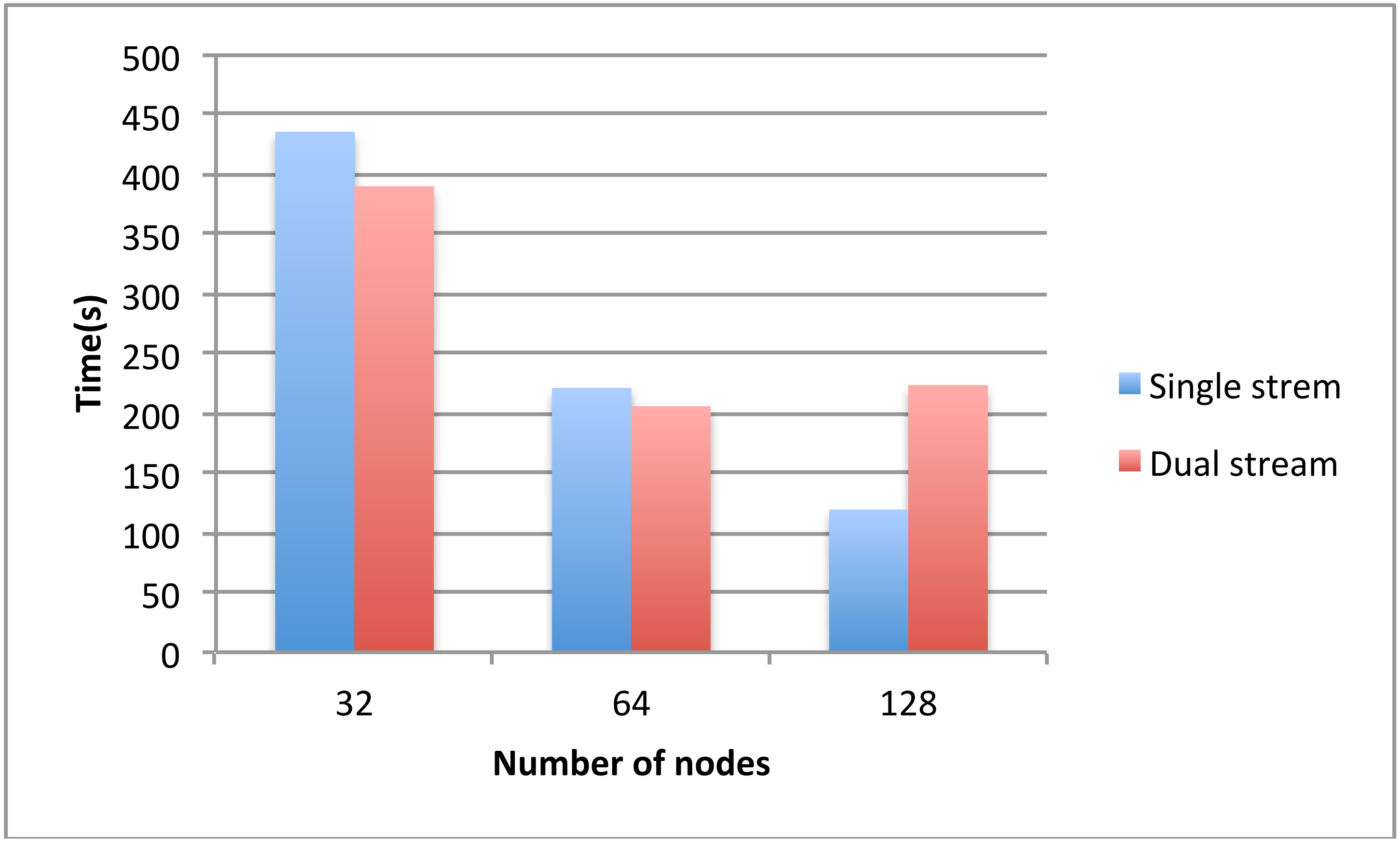
The HT performance benefit occurs at smaller node counts upto 15%, but disappears at larger node counts, gradually hinders the performance. The speedup region from HT does not overlap with the sweet spot of the parallel scaling of the codes.

NWChem runtime with HT and without HT



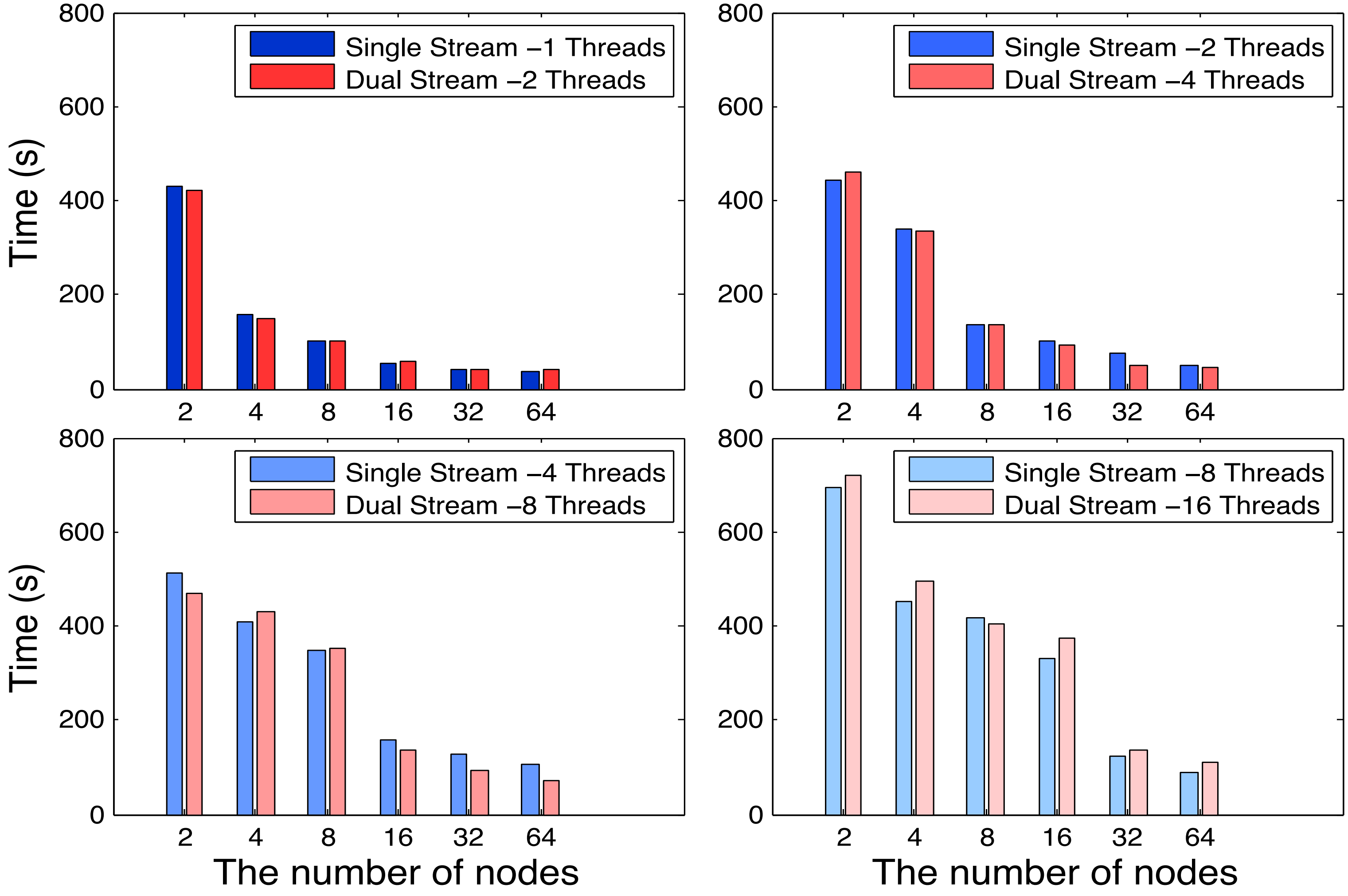
The HT performance benefit occurs at smaller node counts upto 13%, but disappears at larger node counts. The speedup from HT does not overlap with the sweet spot of the parallel scaling of the codes.

GTC runtime with HT and without HT



The HT performance benefit occurs at smaller node counts upto 10%, but disappears at larger node counts and hinders the performance. The slowdown could be at much larger magnitude (80%).

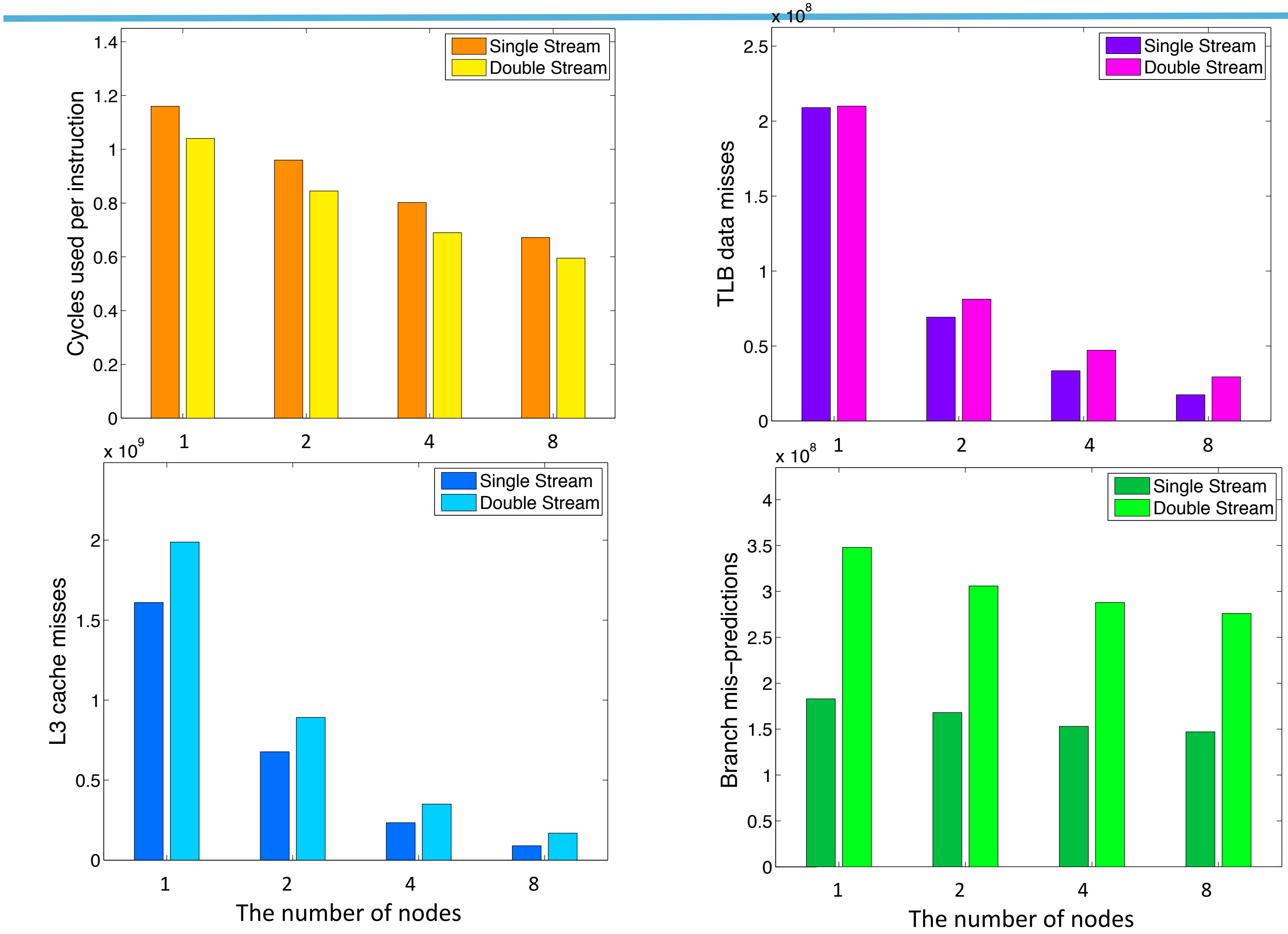
Quantum Espresso runtime with HT and without HT



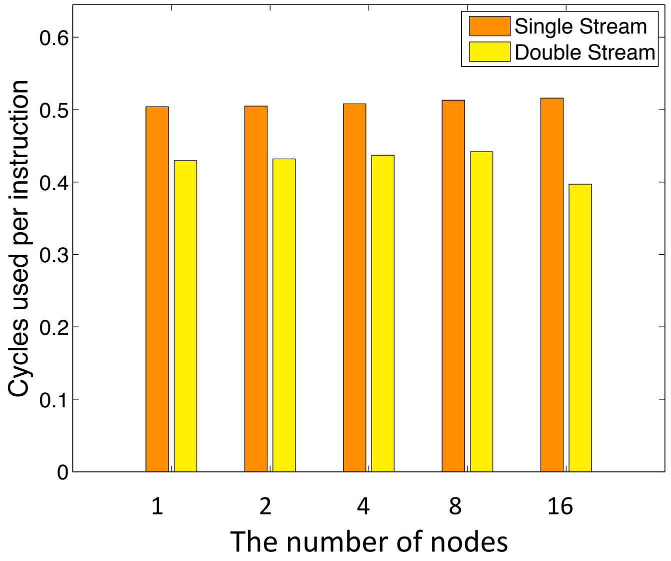
For the most of the MPI task and thread combinations, HT doesn't help the performance.

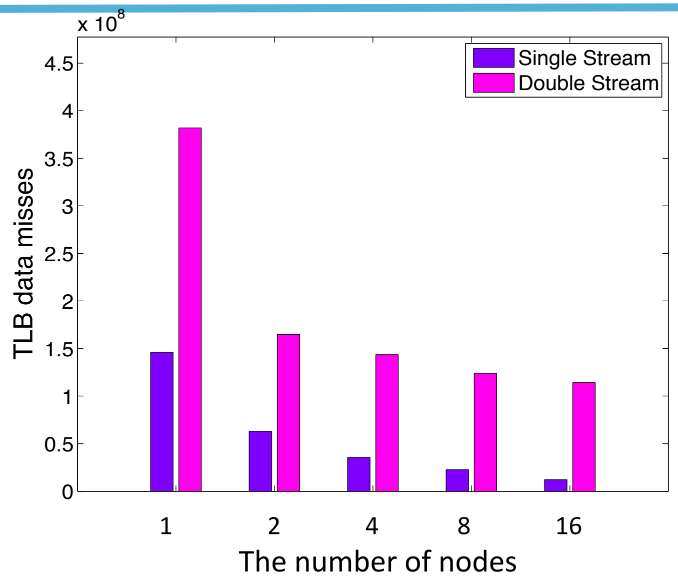
Any hints from profiling regarding when HT benefits code performance?

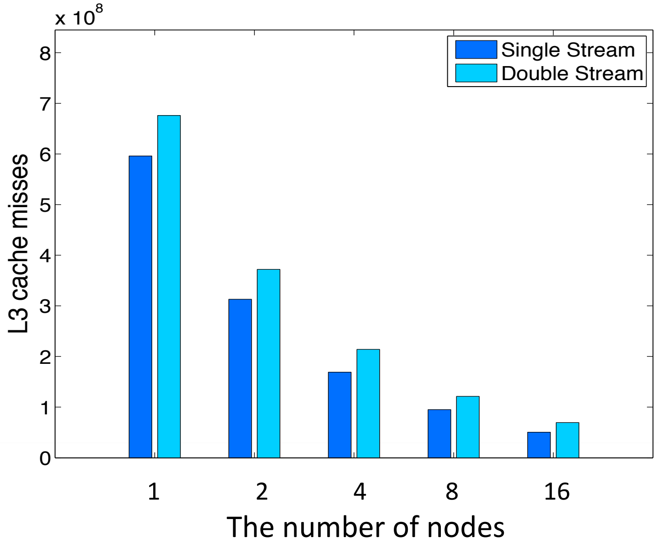
VASP: Cycles used per instruction, L3 cache misses, TLB data misses, and branch mis-predictions per physical core

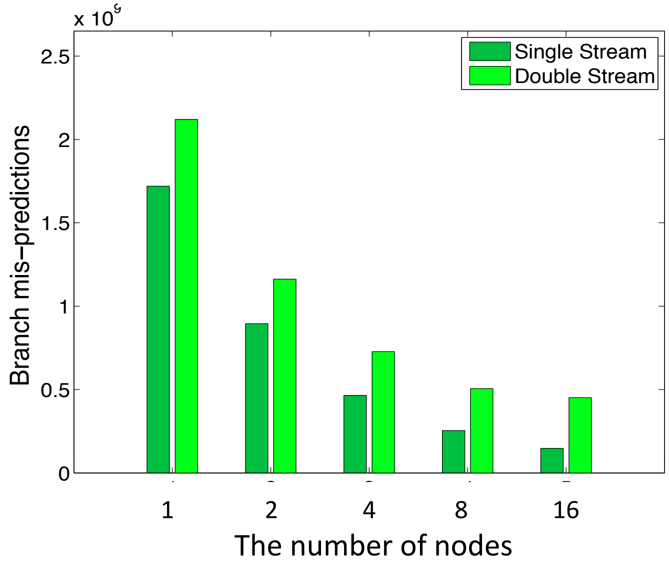


NAMD: Cycles used per instruction, L3 cache misses, TLB data misses, and branch mispredictions per physical core

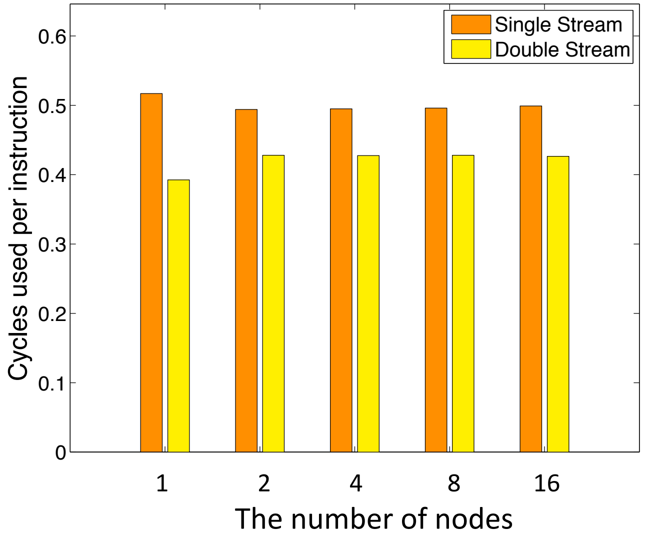


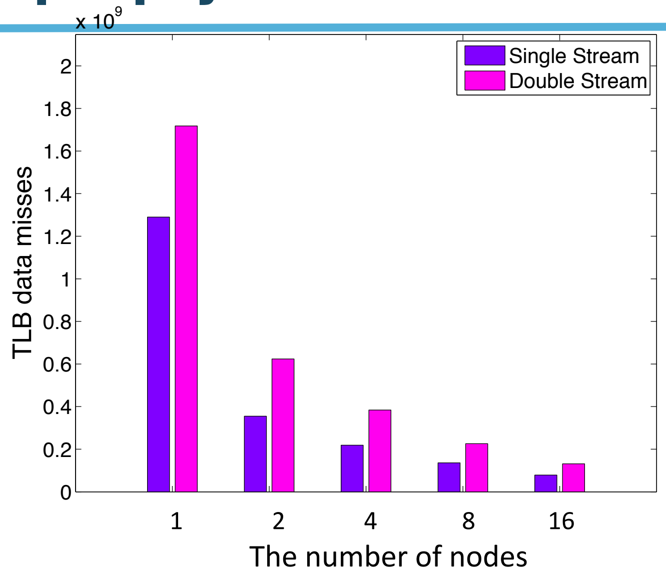


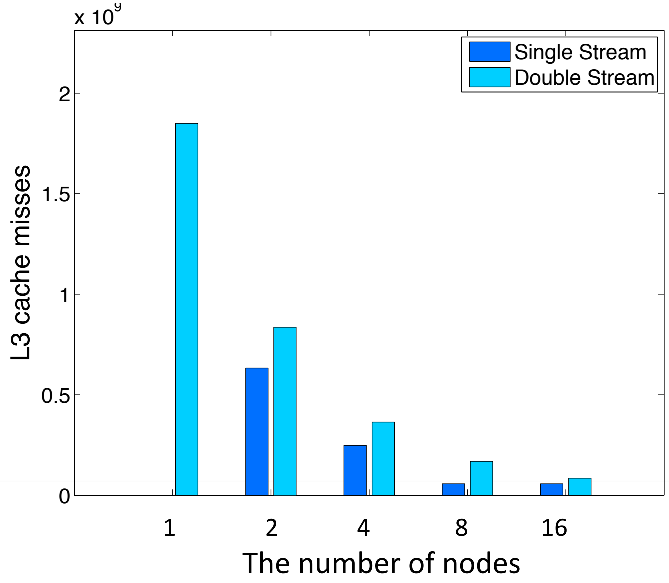


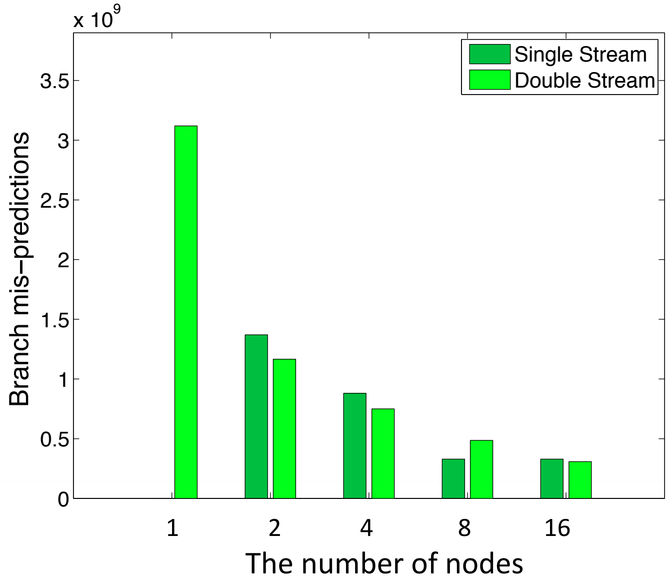


NWChem: Cycles used per instruction, L3 cache misses, TLB data misses, and branch mispredictions per physical core

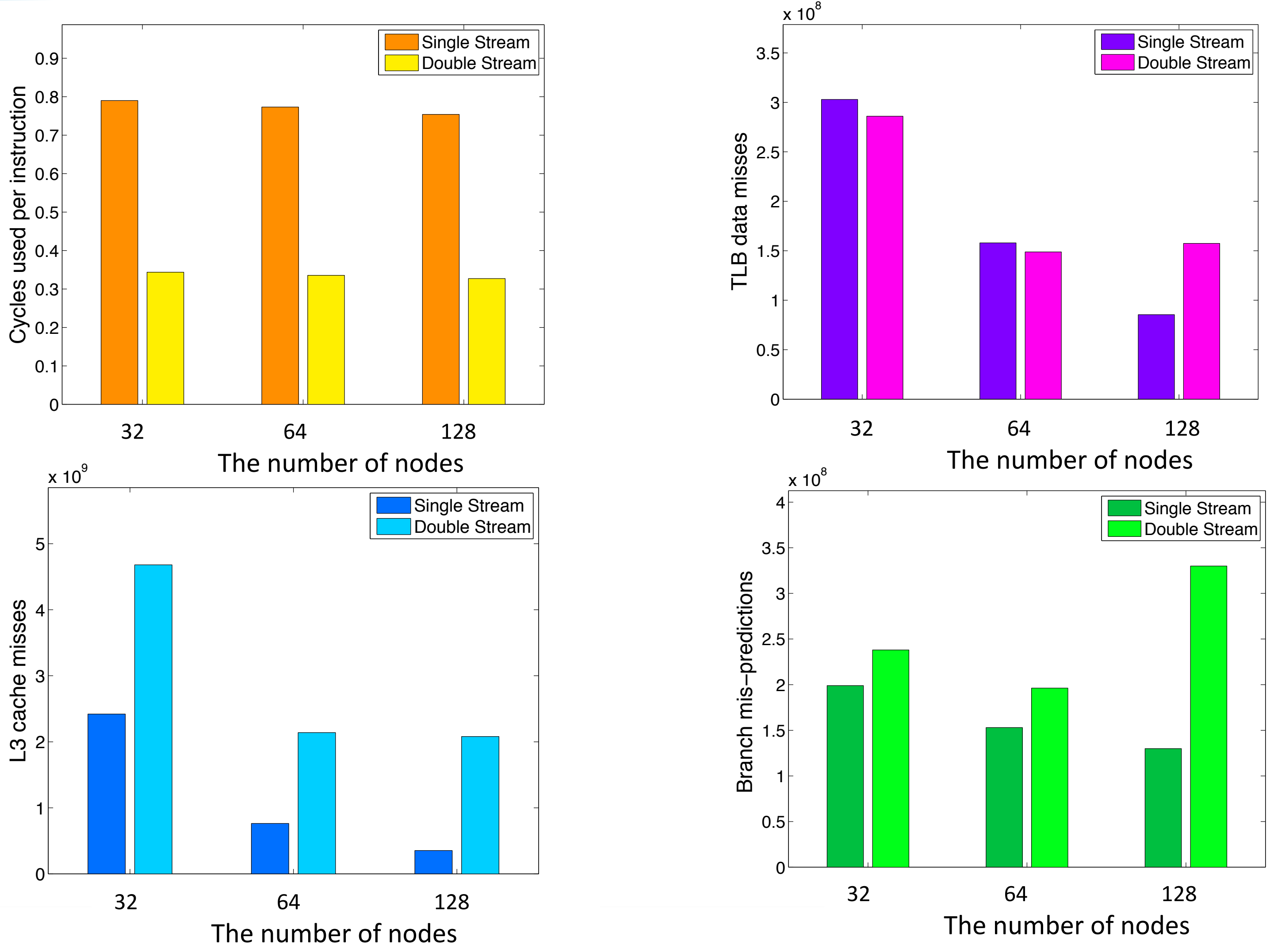








GTC: Cycles used per instruction, L3 cache misses, TLB data misses, and branch mispredictions per physical core



Conclusions

- The HT speedup running at the same node counts is limited to the small node counts, and up to 15% of performance gain was observed from multiple codes.
- However the small speedup quickly disappear at larger node counts, and HT even hinders the code performance with a much larger magnitude. There is no overlap between the HT speedup region and the sweet spot of the parallel scaling.
- Users are recommended to use HT with caution in their production runs
  - The HT speedup region should be well tested for their applications to avoid a large performance penalty from using HT.

Conclusions - continued

- HT performance benefit is realized by the increased resource utilization and sufficient parallel scaling of the application codes (OS should be optimized for HT as well). Whether an application can get performance benefit from using HT depends on if the higher resource utilization overcomes/ nullifies the increased stalls due to resource sharing, communication overhead and other negative contributions introduced by HT itself.
- It is not easy to predict if an application can get performance benefit from profiling data.
- HT as a complementary path in the microprocessor design, we hope to see more HT roles in the HPC workload in the future with continuously improving HT implementations, and application parallel scaling.

Acknowledgement

- Larry Pezzaglia at NERSC for his support and help with early HT study on Mendel clusters at NERSC.
- Hongzhang Shan at Computing Research Division at LBNL, and Haihang You at NICS for insightful discussion and help.
- Richard Gerber and other members of the User Services Group at NERSC for the useful discussions.

Thank you!