SAND2017-4855C

Performance on Trinity Phase 2 (a Cray XC40 utilizing Intel Xeon Phi processors) with Acceptance-Applications and Benchmarks

A. Agelastos*, M. Rajan*, N. Wichmann#, R. Baker+, S. Domino*, E. Draeger@,
S. Anderson#, J. Balma#, S. Behling#, M. Berry#, P. Carrier#, M. Davis#,
K. McMahon#, D. Sandness#, K. Thomas#, S. Warren#, and T. Zhu#

* Sandia National Laboratories, Albuquerque, NM
+ Los Alamos National Laboratory, Los Alamos, NM
@ Lawrence Livermore National Laboratory, Livermore, CA
# Cray, Inc., St. Paul, MN

Cray User Group Meeting, May 7-11, 2017, Redmond, WA, USA

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energys National Nuclear Security Administration under contract DE-NA0003525. This unclassified document is approved for unlimited release (SAND 2017-xxxxx).

NNSA's First Advanced Technology System(ATS-1)

Previous Capability Computing Systems: Cielo, Sequoia

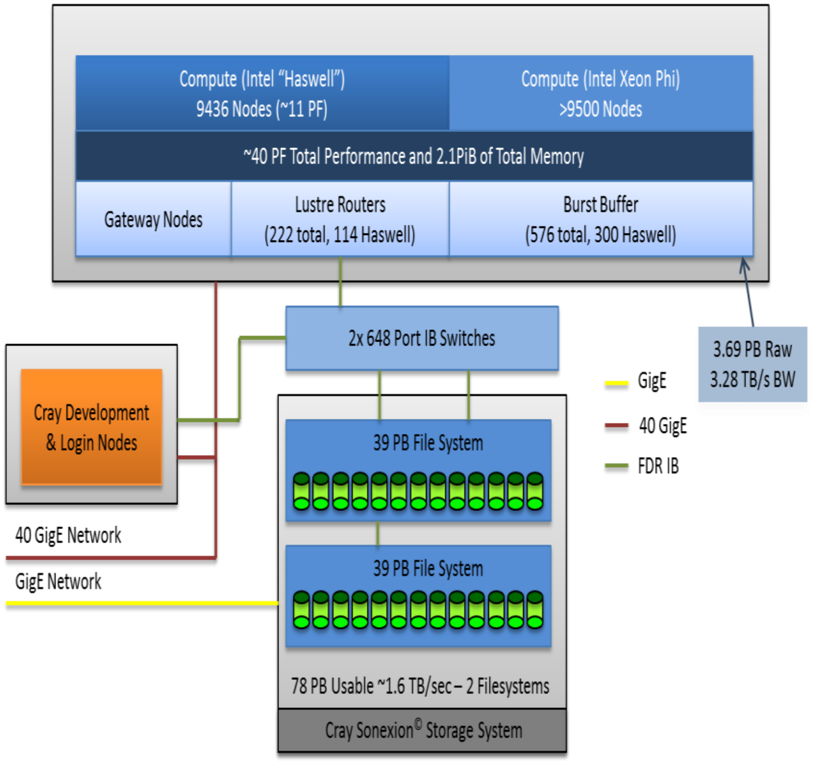
- Trinity (ATS-1) deployed by ACES (New Mexico Alliance for Computing at Extreme Scale, i.e. Los Alamos & Sandia) and sited at Los Alamos. ATS-2 will be led by LLNL, ATS-3 by ACES

Cielo
- Cray XE6
- Nodes =8944
- Memory > 291.5TB
- Peak Performance =1.37 PF
- AMD MagnyCours(16 cores/node)

Sequoia
- IBM BG/Q
- Nodes = 98,304
- Memory = 1.6PB
- Peak Performance = 20PF
- IBM PowerPC A2 (16 cores/node)

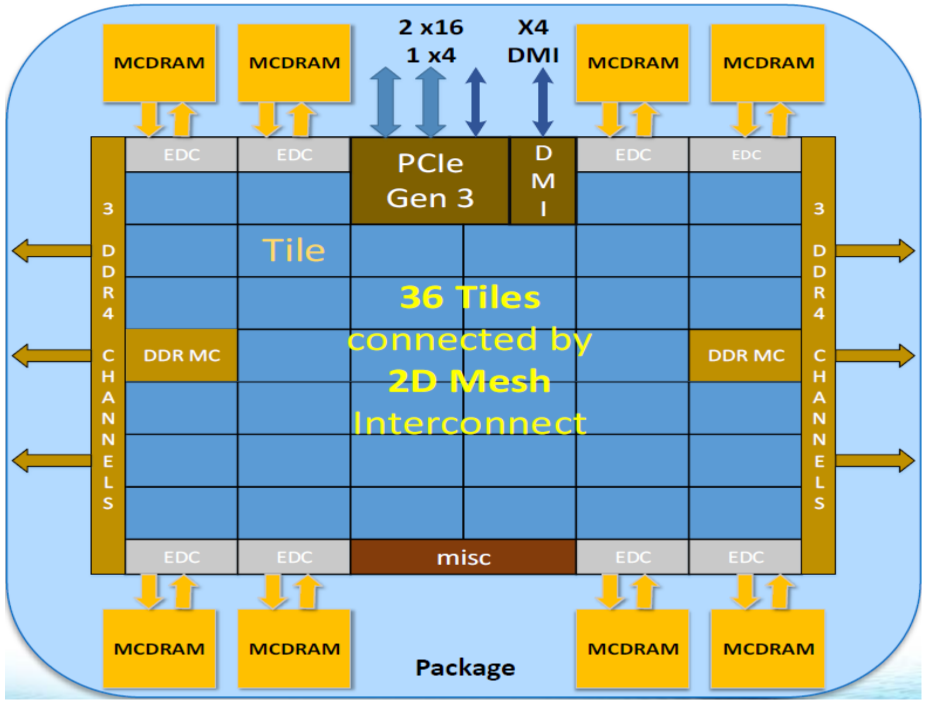
Trinity
- Cray XC40
- Nodes > 19,420
- Memory > 2PB
- Peak Performance > 40PF
- Intel Haswell (32 cores/node) & Knights Landing(68 cores/node)

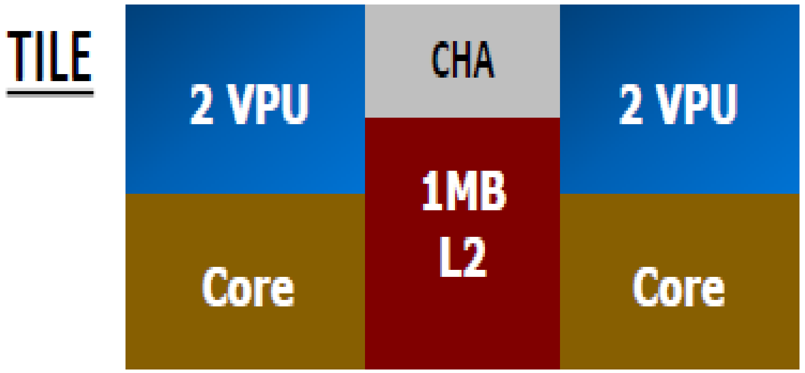
Trinity Architecture: Phase-1 with Haswell Nodes accepted December 2015; Phase-2 KNL nodes accepted December2016



- Peak Haswell Node Performance: 32cores*16FLOPs/cycle*2.3GHz = 1,177.6 GFLOPS/node
- Peak KNL Node Performance: 68cores*32FLOPs/cycle*1.4GHz = 3, 046.4 GFLOPS/node
- Intel Turbo Mode Boost enabled
- Intel Hyper-Threads enabled

Trinity KNL Processor Architecture





- Three Cluster Modes: All-to-All, Quadrant, Sub-NUMA Clustering
- MCDRAM Modes
  - Flat: Exposed as a separate NUMA node
  - Cache: Direct mapped Cache, 64B lines
  - Hybrid: Part Cache, Part Memory; 25% or 50% cache

- Intel Turbo Mode Boost enabled
- 4 Threads per core. Simultaneous Multithreading (SMT)
- 2 VPU per core: 2x AVX 512 units; 32 SP/16DP per unit
- 2D-Mesh connections for tile
- DDR4: 6 channels @ 2.4 GHz

Trinity Phase-2 Acceptance

Completed December 2016

1) Capability Improvement(CI) metric
   - 4X over a baseline performance measured on 2/3rd of the nodes on Cielo
   - runs at near full scale
   - May use appropriately scaled inputs
   - Three applications representative of Tri-lab productions apps ; **Nalu, PARTISN, QBOX**
2) NERSC's Sustained System Performance (**SSP**) target of 489; specified input: "large"
3) Microbenchmarks: Stream, OMB, SMB, mpimemu, psnap, pynamic
4) Run at full scale SSP benchmarks: miniFE, miniGhost, AMG, UMT and SNAP

Cielo, Trinity Architectural Parameter Comparisons

System	Cielo (XE6)	Phase-1	Phase-2
Total Nodes	8,894	9,436	9,975
Total Cores	142,304	301,952	678,300
Processor	AMD MagnyCours	Intel Haswell	Intel Xeon Phi (KNL)
Processor ISA	SSE4a	AVX2	AVX-512
Clock Speed(GHz)	2.40	2.30	1.4
Cores/node	16	32	68
Memory-per-core(GB)	2	4	1.41 (DDR4) 0.235 (MCDRAM)
Memory	DDR3 1,333 MHz	DDR4 2,133 MHz	DDR4 2,400 MHz
Peak node GFLOPS	153.6	1,177.6	3,046.4
DDR Channels/socket	4	4	6

Cache L1(KB)
L2(KB)
L3(MB)
	8 x 64
	8 x 512
	10
		16 x 32
		16 x 256
		40
			68x32
			34 x 1,024
			16 GB MCDRAM (if in Cache

Interconnect Topology	Gemini 3D Torus 18x12x24	Aries Dragonfly	

CI Metric and Applications

SNL App: SIERRA/Nalu:

- Low Mach CFD code for incompressible flows; unstructured mesh; LES/Turbulence Models
- Test Problem:
  - Turbulent open jet (Reynolds number of ~6,000)
  - Weak scaling meshes (R1:268k elements, R2:2.15M elements, ….. R6: 9 billion elements)
- Figure of Merit: Solve time/Linear iteration (66%)& Assemble time/non-linear step(34%)

LANL App: PARTISN:

- PARTISN particle transport code [6] provides neutron transport solutions on orthogonal meshes in one, two, and three dimensions
- Test Problem: MIC_SN (MIC with group-dependent Sn quadrature).
- Figure of Merit: *Solver Iteration Time (should stay constant for weak scaling)*

LLNL App: Qbox:

- first-principles molecular dynamics code used to compute the properties of materials at the atomistic scale
- Test Problem: benchmark problem is the initial self-consistent wave function convergence of a large crystalline gold system (FCC, a0 = 7.71 a.u).
- Figure of Merit: maximum total wall time to run a single *self-consistent iteration* with three non-self-consistent inner iterations)

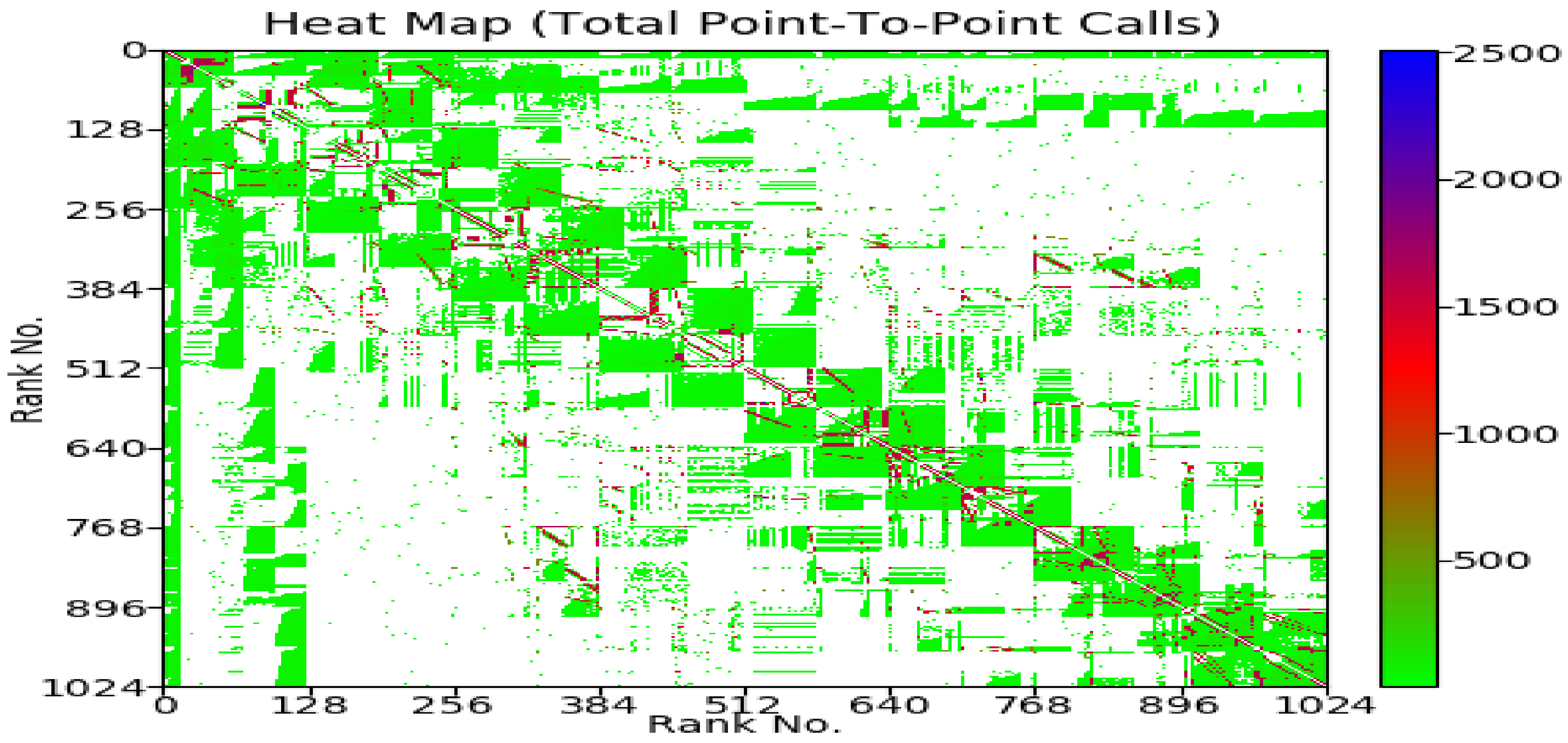
SIERRA/Nalu CI Performance

BASELINE on Cielo				Trinity Phase -2 CI Results			
Nodes	MPI Tasks	Problem Size Complexity Measure	RunTime FOM	Nodes	MPI Tasks	Problem Size Complexity Measure	RunTime FOM
8,192	131,072	*R6*: 9B elements	1.15	4,096	262,144	*R6*: 9B elements	0.689

Running the same problem (9 Billion element mesh) on 2 times the number of PEs in quad-cache mode resulted in a Capability Improvement Metric of =1.15/0.689 = **1.67**

CI for Nalu fell short of the target 4.0 because out-of-memory (OOM) errors prevented runs at 8192 KNL nodes in quad-cache mode

KNL MPI memory usage Investigation; Nalu p2p communications



Heat map of total number of point-to-point (p2p) calls for “R3 Mesh” 1,024 MPI ranks
Rank 14 for the “R3 Mesh” and Rank 13 for the “R4 Mesh” are connected to 1,023 and 7,671 ranks, respectively (horizontal block of green at the top of the Heat Map). This heavy p2p connectivity on few MPI tasks resulted in MPI buffers memory growth leading to the “OOM” error

Nalu hit OOM when run on 8192 KNL nodes; Is MPI memory growth to blame?

MPI Memory measured in GB/node with a benchmark run 8192 KNL nodes

Test Description	524,288 Tasks; 64 tasks/node	393,216 Tasks; 48 tasks/node	262,144 Tasks; 32 tasks/node	131,072 Tasks; 16 tasks/node
Full P2P connectivity	64.4	36.5	16.5	4.3
Full P2P connectivity with MPICH_GNI_MBOXES_PER_BLOCK=<num_connections>	60.4	34.3	15.5	4.1
Full connectivity using MPI_Alltoall	6.7	4.5	2.3	0.8
1024 P2P connections/task	6.2	3.9	2.0	0.7

PARTISN CI Performance

BASELINE on Cielo				Trinity Phase -2 CI Results			
Nodes	MPI Tasks	Problem Size Complexity Measure	RunTime FOM	Nodes	MPI Tasks	Problem Size Complexity Measure	RunTime FOM
8,192	32,768 (4 OMP threads/task)	*2,880 zones/core*	209.4 secs	8,192	262,144 (2 OMP threads/task)	*6,480 zones/core*	332.047 secs

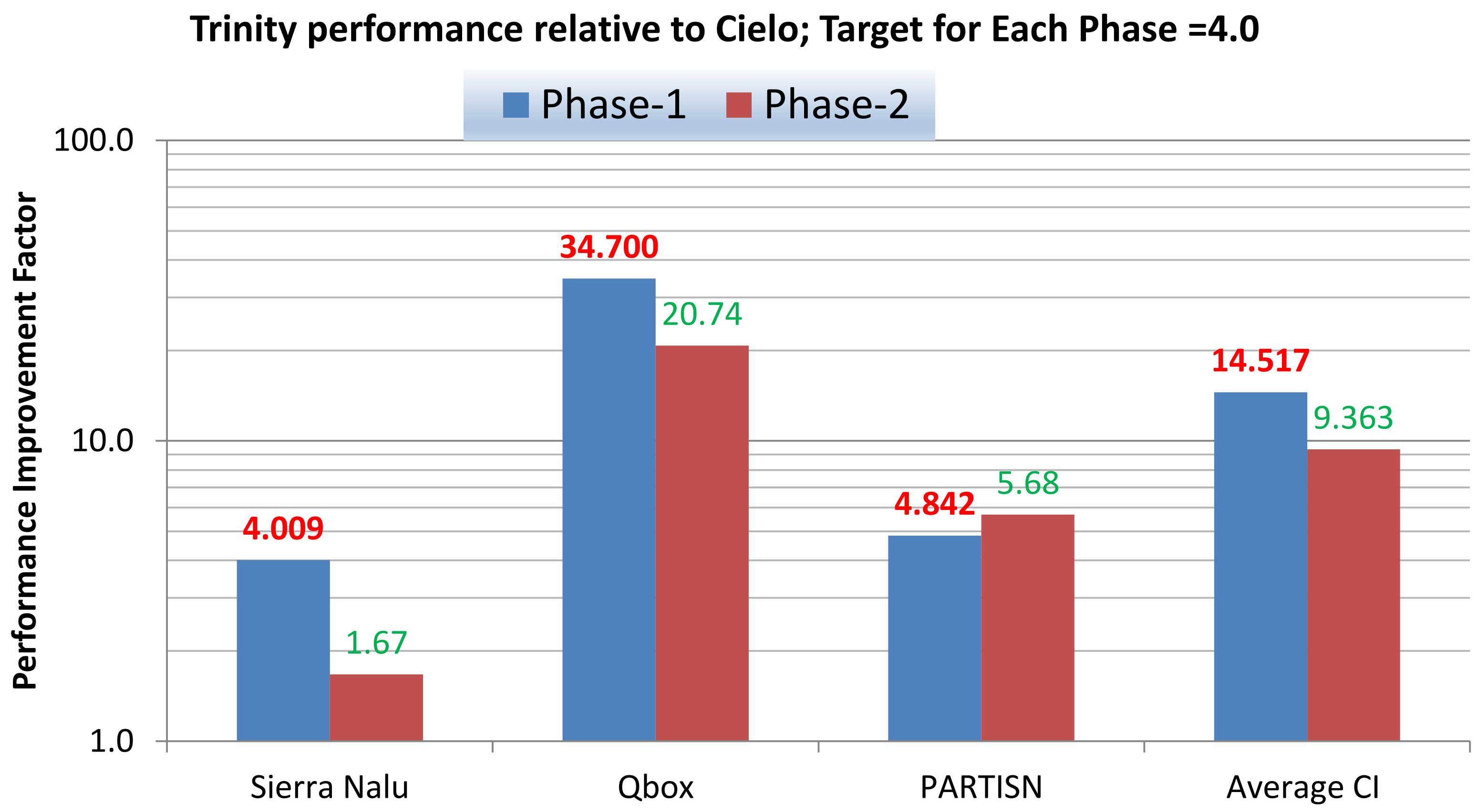
Running a (6480*262144*2)/(2880*32768*4) = 9 times larger problem on 4 times the number of PEs took 1.585 times longer *solver iteration time* leading to a Capability Improvement Metric of = 9/ 1.585 = **5.68**

Qbox CI Performance

BASELINE on Cielo				Trinity Phase -2 CI Results			
Nodes	MPI Tasks 1 thread/task	Problem Size Complexity Measure	RunTime FOM	Nodes	MPI Tasks 4 threads/Task Hyperthreads	Problem Size Complexity Measure	RunTime FOM
6,144	98,304	*1,600 Atoms*	1663 secs	8,504	136,064	*6,000 Atoms*	4,227.4 5 secs

Running a (6000/1600) ** 3 = 52.73 time larger problem on 5.536 times the number of PEs took 2.54 times longer *self-consistent iteration time* leading to a Capability Improvement Metric of = 52.73 / 2.54 = **20.74**

Capability Improvement Summary



With Qbox, KNL is overall about 1.75 times slower than Xeon, per node

PARTISN: Performance tuning

- opt_sweep3d(), which actually performs the KBA sweep that comprises the wave-front algorithm, took 85% of time; optimized for vectorization by Randy Baker and team at LANL
- 8MB huge pages was used
- Hybrid MPI + OpenMP with the following env settings were used:
  - *export OMP_WAIT_POLICY=active*
  - *export OMP_NUM_THREADS=2*
  - *export OMP_PROC_BIND=spread*
- MPI Isend/MPI_Recv communications were frequent on the 2D processor mesh. Cray utility *grid_order* which "repacks" MPI ranks so that Cartesian mesh communication neighbors are more often on node was used to minimize communication overhead
- MPICH_RANK_ORDER generated from *grid_order* utility:

```
# grid_order -R -Z -c 4,4 -g 512,512 -m 262144 -n 32
```

Qbox performance tuning

- Compute time dominated by parallel dense linear algebra and parallel 3D complex-to-complex Fast Fourier Transforms.
- Efficient single-node kernels used and necessary to achieve good peak performance.
- The communication patterns are complex, with nonlocal communication occurring both within the parallel linear algebra library (ScaLAPACK) and in sub-communicator collectives within Qbox, which are primarily MPI_Allreduce and MPI_Alltoallv operations.
- Threading implemented as a mix of OpenMP and threaded single-node linear algebra kernels
- 1,408 atom runs on 1024 nodes of KNL gave the best performance with 32 tasks/node, 2 OMP threads/task and no hyperthreading
- MPICH_RANK_ORDER generated from *grid_order* utility:

```
# grid_order -R -P -c 4,4 -g 128,1063 -m 301376 -n 16
```

NERSC's Sustained System Performance (SSP) Metric

- A set of benchmark programs that represent a workload
- Computed as a geometric mean of the performance of eight Tri-Lab and NERSC benchmarks
  - *miniFE, miniGhost, AMG, UMT, SNAP, miniDFT, GTC and MILC*

Trinity Phase-1 SSP target was 489 Achieved 581

Baseline SSP performance on NERSC's Hopper (Cray XE6)

Hopper Nodes	6384					
Hopper SSP						
Application Name	MPI Tasks	Threads	Nodes Used	Reference Tflops	Time (seconds)	Pi
miniFE	49152	1	2048	1065.151	92.4299	0.0056
miniGhost	49152	1	2048	3350.20032	95.97	0.0170
AMG	49152	1	2048	1364.51	151.187	0.0044
UMT	49152	1	2048	18409.4	1514.28	0.0059
SNAP	49152	1	2048	4729.66	1013.1	0.0023
miniDFT	10000	1	417	9180.11	906.24	0.0243
GTC	19200	1	800	19911.348	2286.822	0.0109
MILC	24576	1	1024	15036.5	1124.802	0.0131
					Geom. Mean=	0.0082
					SSP=	52.1212

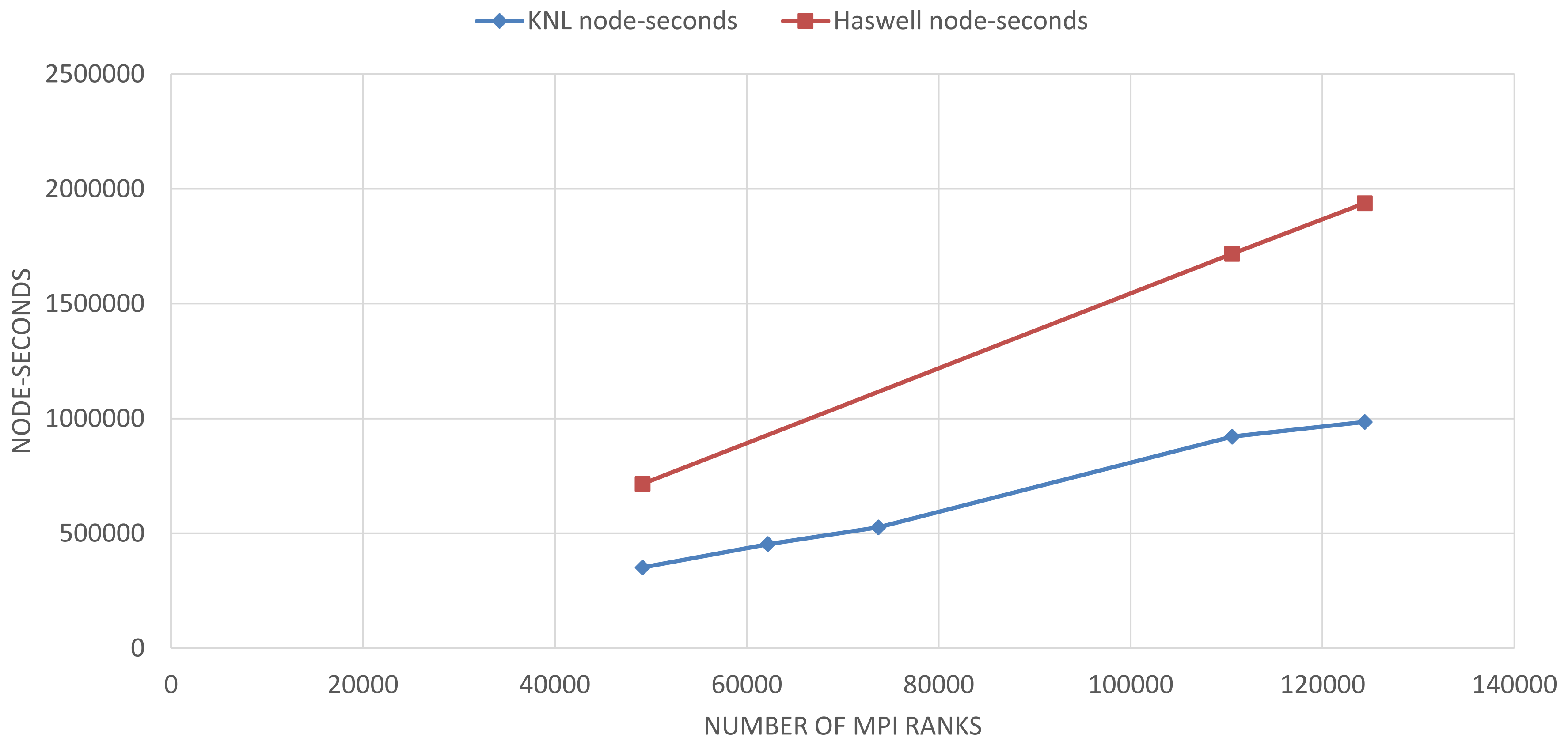
SSP performance on Trinity Phase-2

Application Name	Nodes Used	NERSC Elapsed Time Proposed	Proposed Pi	Elapsed Time	December times	KNL Pi December
miniFE	3840	7.7	0.03602	7.22	7.20	0.0385
miniGhost	768	29.6	0.14737	33.86	33.95	0.1285
AMG	768	165	0.01077	140.70	160.36	0.0111
UMT	769	552	0.04337	449.94	467.26	0.0512
SNAP	768	216	0.02851	216.07	212.00	0.0290
miniDFT	47	1020	0.19149	478.87	471.62	0.4142
GTC	150	2396	0.05540	2195.05	2118.73	0.0627
MILC	384	882	0.04440	612.93	631.27	0.0620
		Geom. Mean=	0.04901	Geom. Mean=		0.0582
				Trinity SSP =		580.92
				Target Trinity SSP =		489

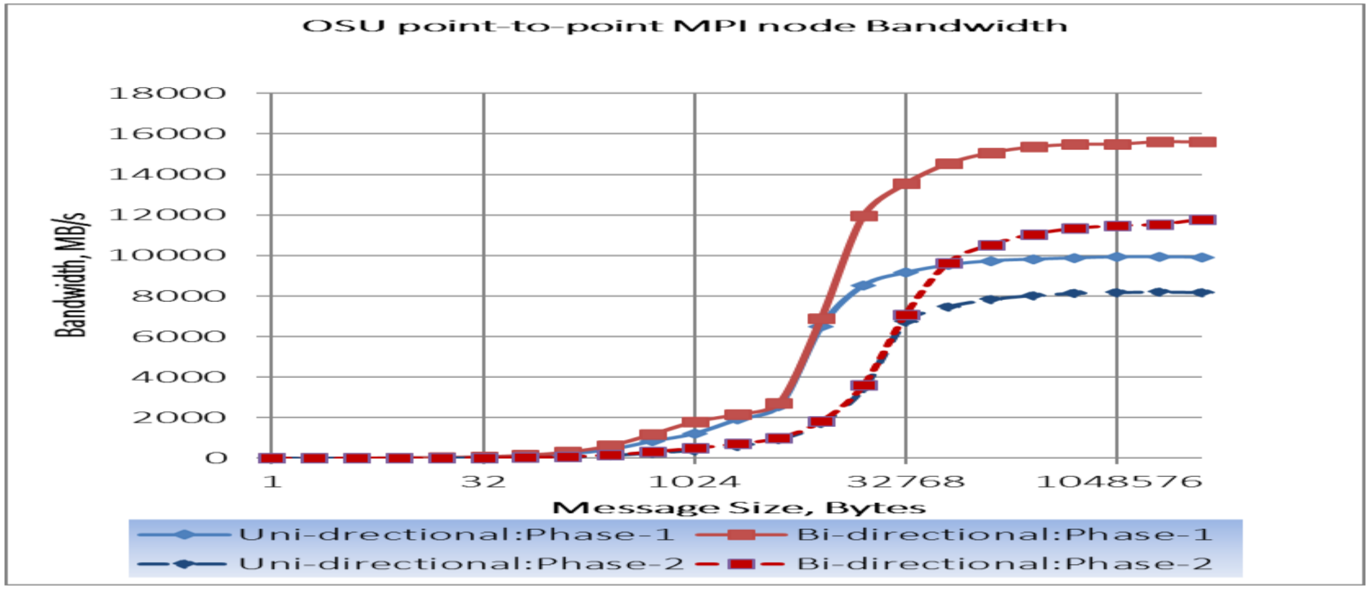
Extra large runs example: UMT

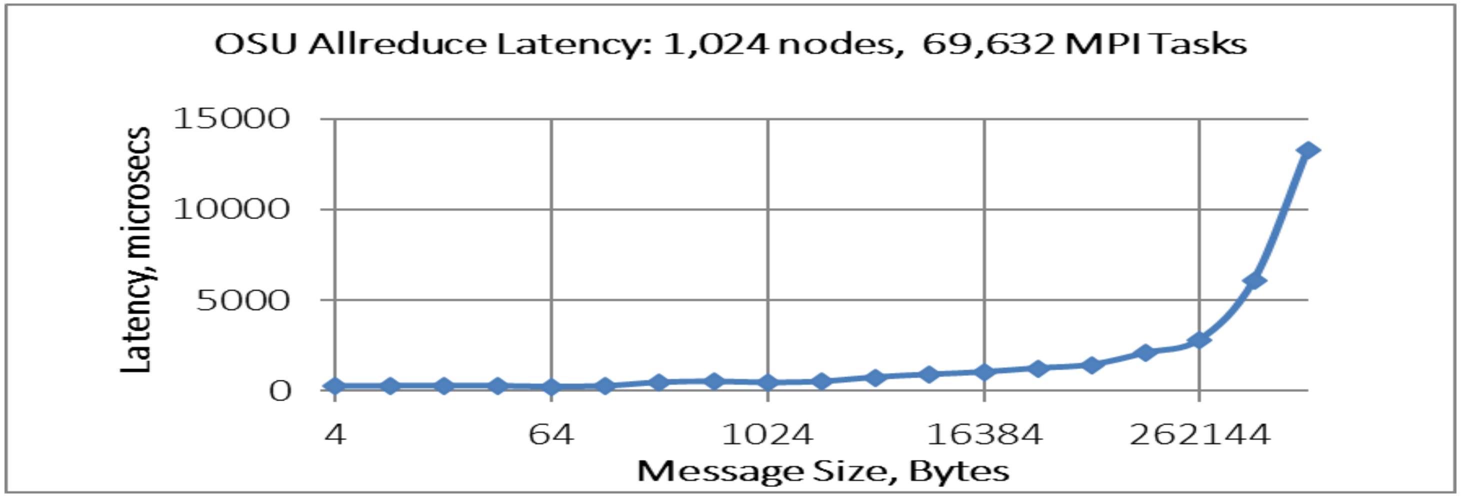
Phase-2 KNL compared to Phase-1 Haswell

Several Optimizations by Cray's Steve Behling: Cray Compiler, Quad-cache for KNL, 64 MPI ranks/KNL-node, 4 OMP threads per KNL-rank (hyperthreading!); MPMD used; 4M hugepages, Vectorization



Sample Micro-benchmark Results





Conclusions-I

- Several months of effort by the Cray and Tri-Lab teams resulted in exceeding performance acceptance requirements
- Based on benchmark results we anticipate production Trinity apps will see a gain of 2x-6x over Cielo
- Benefit of hybrid MPI + Threads clearly seen with Qbox
- use of *grid_order* resulted in good performance gains for CI and many SSP apps

Conclusions-II

- KNL's Quad/Cache mode is a good-performing, general purpose mode for applications which have not yet directly mapped selected data structures to MCDRAM.
  - highly susceptible to thrashing if important data aliases to the same location
  - This becomes more and more likely as node counts increase (seen with SSP apps: GTC, miniFE)
  - If an application's entire memory footprint fits in MCDRAM Quad/Flat is great!
- Both Intel and Cray compilers were used depending upon the application.
- Static linking, more often than not, achieves better performance than dynamic linking.
- Huge pages typically provides a performance increase and should be investigated for each application.
- Grid ordering improves performance for many applications and should also be investigated for each application. Many benchmarks also benefited from 'Core Specialization'.
- Hybrid parallelism can improve performance over MPI everywhere on KNL, however Cray's MPICH implementation on Trinity scales remarkably well and can be used in the interim while developing hybrid parallelism within an application.
- It is important to closely monitor application communication patterns at larger scales, to ensure that point-to- point connections does not lead to large per-node MPI memory (buffers), minimizing available memory for the application