LA-UR-17-29604
Approved for public release; distribution is unlimited.

Title: Recent Performance Results of VPIC on Trinity

Author(s): Nystrom, William David; Bergen, Benjamin Karl; Bird, Robert Francis; Bowers, Kevin J.; Daughton, William Scott; Fogerty, Shane Patrick; Guo, Fan; Le, Ari Yitzchak; Li, Hui; Nam, Hai Ah; Pang, Xiaoying; Stanier, Adam John; Stark, David James; Rust, William Newton III; Yin, Lin; Albright, Brian James

Intended for: 59th Annual Meeting of the APS Division of Plasma Physics, 2017-10-23/2017-10-27 (Milwaukee, Wisconsin, United States)

Issued: 2017-10-19

Recent Performance Results of VPIC on Trinity

W.D. Nystrom, B. Bergen, R.F. Bird, K.J. Bowers, W. S. Daughton, S. Fogerty, F. Guo, A. Le, H. Li, H. Nam, X. Pang, A. Stanier, D.J. Stark, W.N. Rust III, L. Yin, B.J. Albright

October 24th, 2017

Operated by Los Alamos National Security, LLC for the U.S. Department of Energy's NNSA

VPIC
Overview

VPIC is a 3D explicit, relativistic, charge-conserving electromagnetic particle-in-cell (PIC) code originally developed by Kevin Bowers at LANL

- **Open Source: https://github.com/losalamos/vpic**
- **Single Precision**
- **Structured Cartesian Mesh**
- **Array-of-Structures (AoS) (particle and field data)**
- **Asynchronous MPI (distributed memory)**
- **OpenMP or Pthreads (shared memory)**
- **Explicit short/wide vectorization using hardware intrinsics**

VPIC
Overview (cont)

- Figure 1 shows an example of a VPIC astrophysical calculation performed during the Trinity Phase 1 Open Science Campaign in February, 2016.
- More information on VPIC is available from the following references:
- K. J. Bowers, B. J. Albright, L. Yin, B. Bergen, and T. J. T. Kwan, “Ultrahigh performance three-dimensional electromagnetic relativistic plasma simulation”, Physics of Plasma, vol 15, no. 5, 2008.
- K. J. Bowers, B. J. Albright, B. Bergen, L. Yin, J. Barker, and D. J. Kerbyson, “0.374 Pflop/s Trillion-Particle Kinetic Modeling of Laser Plasma Interaction on Roadrunner”, SC 2008: Proceedings of the 2008 ACM/IEEE Conference on Supercomputing, (Piscataway, NJ, USA: IEEE Press) pp 1-11.
- K. J. Bowers, B. J. Albright, L. Yin, W. Daughton, V. Roytershteyn, B. Bergen, and T. J. T. Kwan, “Advances in petascale kinetic plasma simulation with VPIC and Roadrunner”, Journal of Physics: Conference Series 180 (2009) 012055.

Figure 1: Trinity Phase 1 Open Science Calculation

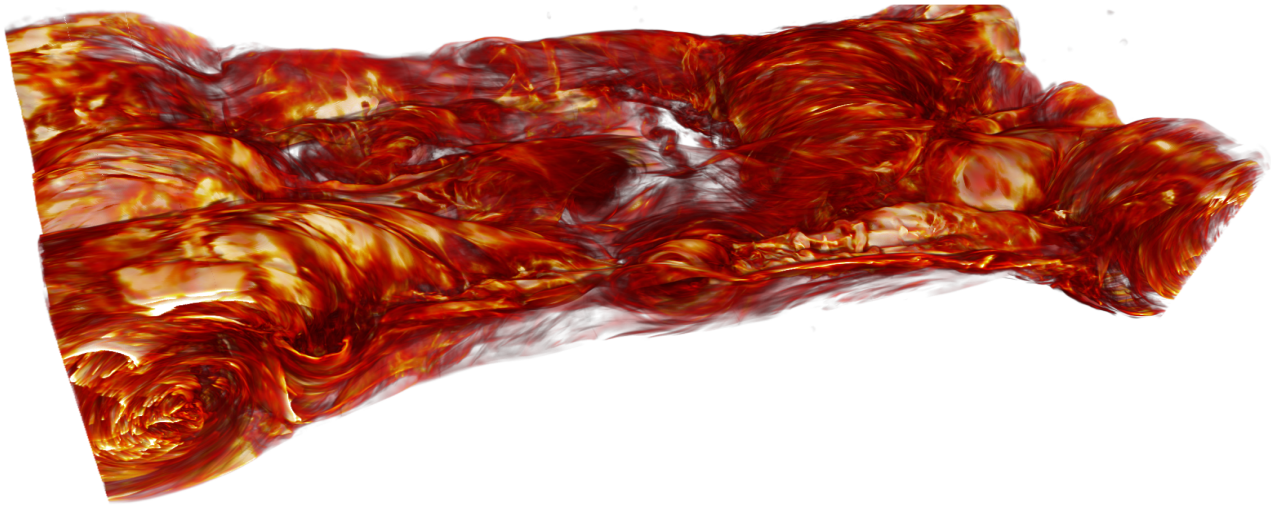


Figure 1: Volume rendering of the current density from one of the reconnection runs performed during the Trinity Phase 1 Open Science Campaign.

Trinity Overview

Trinity Overview

- Trinity is the new capability class HPC machine at LANL that is managed by the LANL/Sandia ACES Consortium and replaces Cielo.
- Trinity has two partitions of roughly equal size, an Intel Haswell partition and an Intel Knights Landing (KNL) partition.
- Trinity Phase 1 consisting of the Haswell partition, half the burst buffers and a parallel file system was delivered and accepted in 2015.
- Trinity Phase 2 consisting of the KNL partition and the remainder of the burst buffers is currently undergoing acceptance.
- Unique and new features of Trinity include the Intel Xeon Phi Knights Landing processor which can be configured at run time into 20 different modes, on package High Bandwidth Memory (HBM) for KNL and burst buffer technology to augment performance of I/O.

Trinity by the Numbers

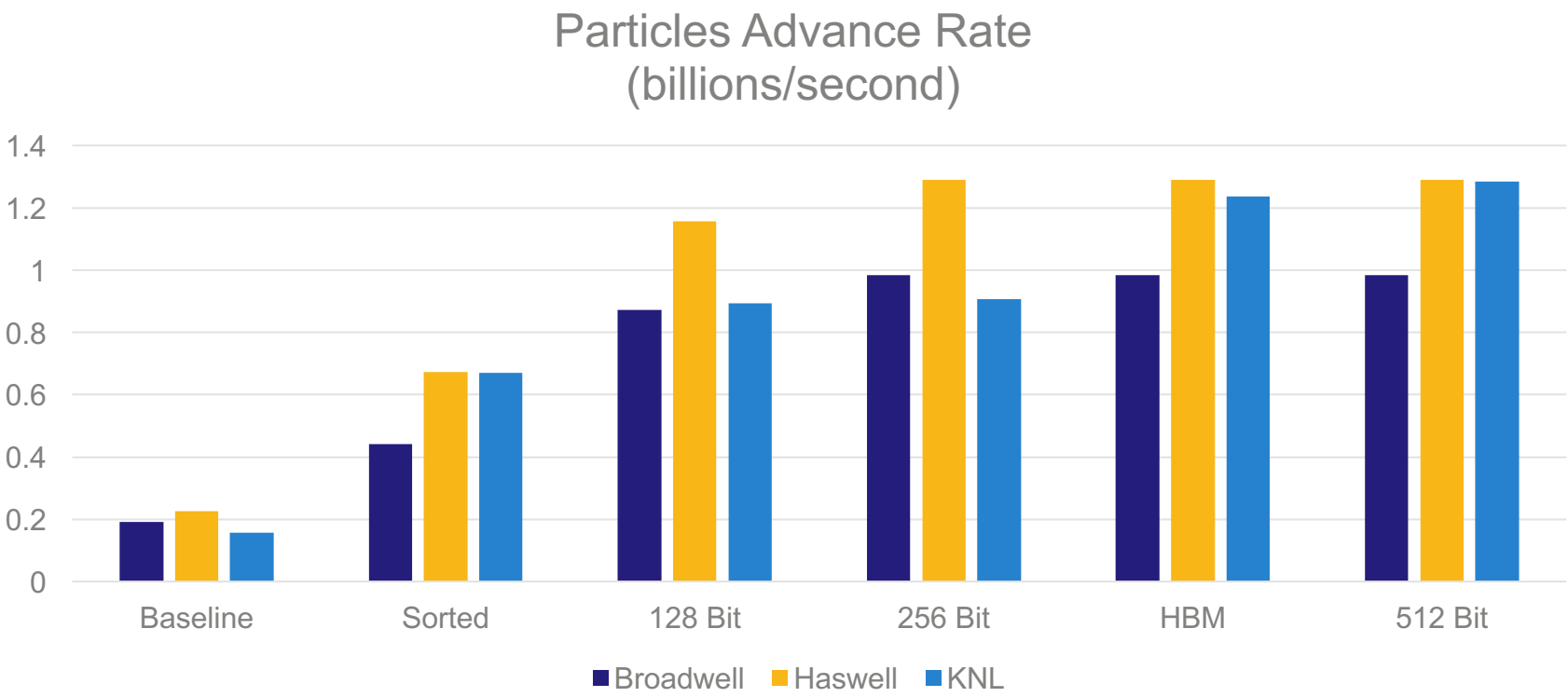
Parameter	Phase 1	Phase 2	Total
Nodes	9436 Haswell	9984 KNL	19420 Nodes
Cores/Node	32	68	
HW Threads/Node	64	272	
Memory/Node	128 GiB	96 GiB (+16G HBM)	
Total Memory	1.15 PiB	0.91 PiB	2.07 PiB
Node Peak Perf	1.18 Tflops	3.01 Tflops	
System Peak	11.1 Pflops	30.7 Pflops	41.8 Pflops
PFS Capacity	78 PB	Unchanged	78 PB
PFS Bandwidth	~ 0.8 TB/S	~ 0.8 TB/S	1.6 TB/S
Burst Buffer Nodes	300	276	576
BB Capacity	1.92 PB	1.77 PB	3.65 PB
BB Bandwidth	1.71 TB/S	1.57 TB/S	3.28 TB/S

Single Node Performance

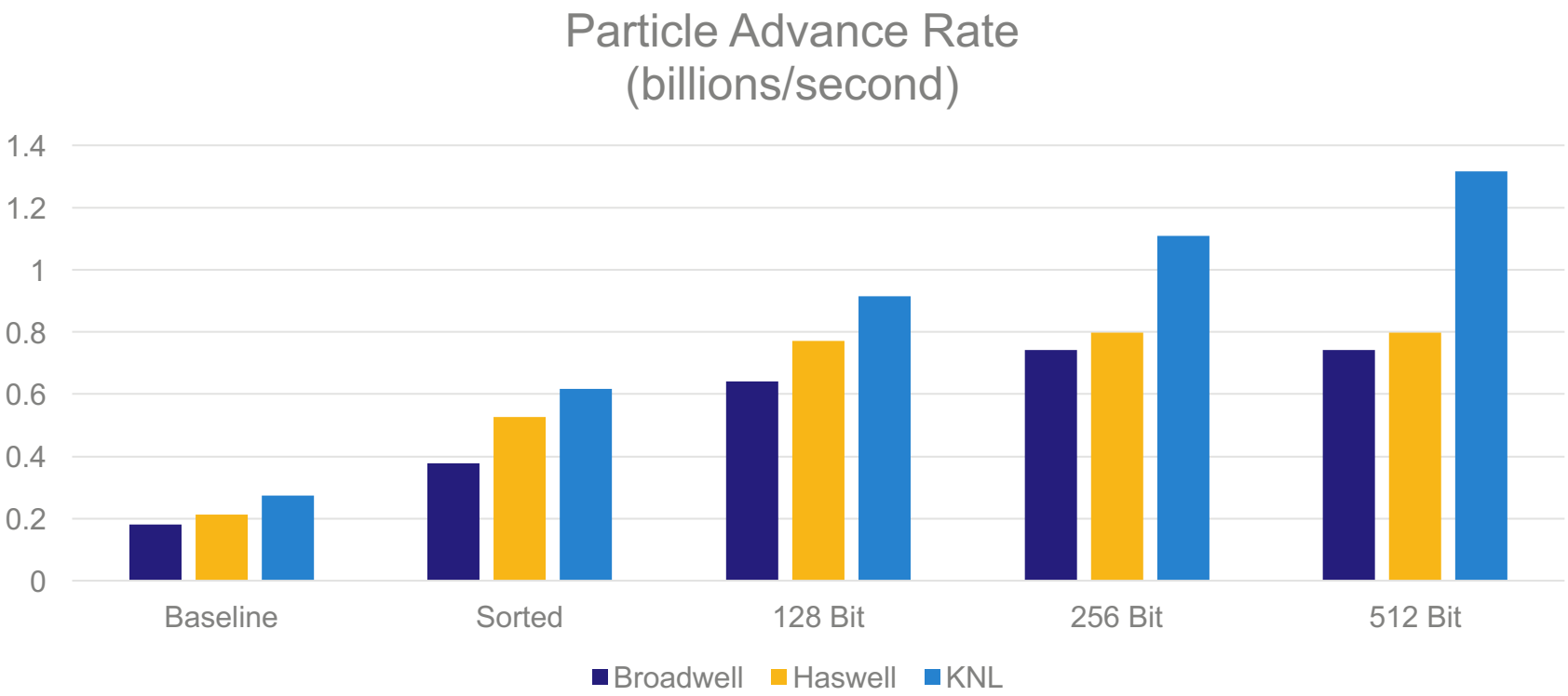
VPIC
Analysis Problems

- **Field Mesh (same for both problems)**
  - 544x96x96 (more cells in the *x*-direction)
  - ~5,000,000 cells
  - Ion and electron species
- **Large Problem (exceeds HBM capacity on KNLs)**
  - 250 particles per cell
  - ~80 GB Memory
- **Small Problem (fits HBM capacity on KNLs)**
  - 25 particles per cell
  - ~8 GB Memory

VPIC
Performance – Large Problem

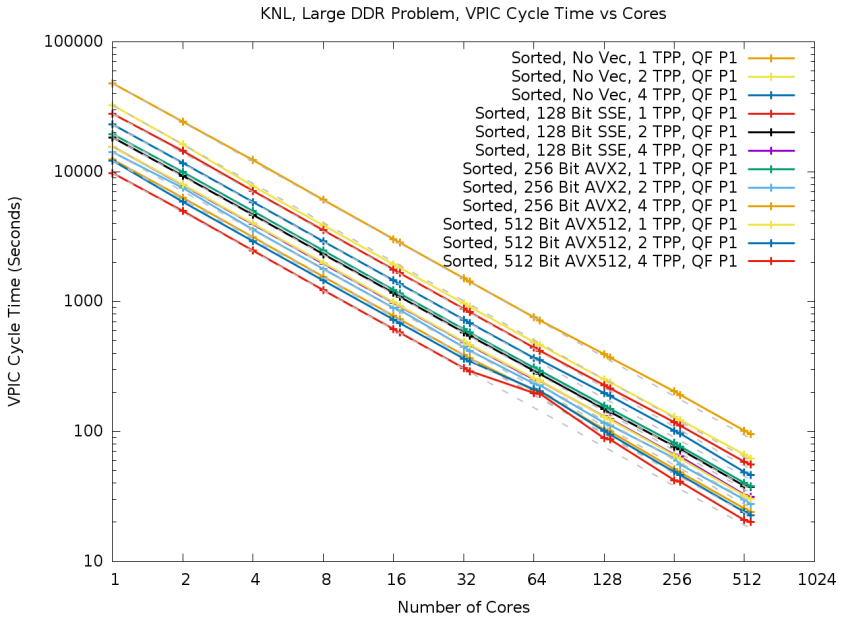


VPIC
Performance – Small Problem

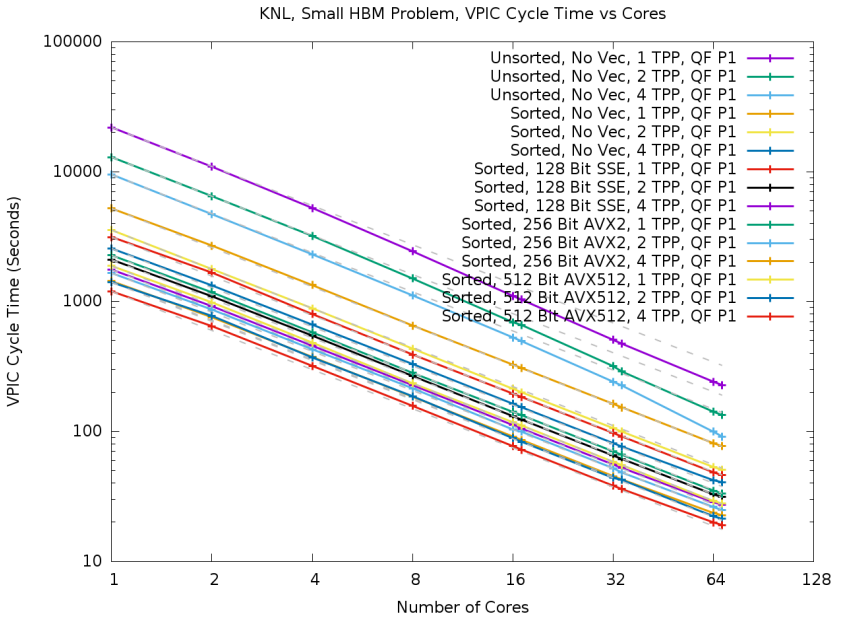


VPIC
Strong Scalability

Large Problem

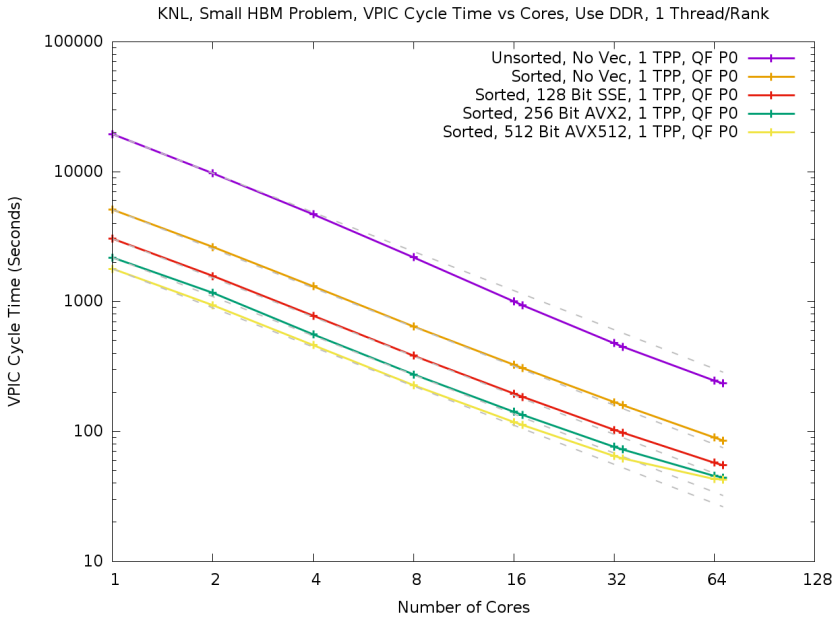


Small Problem

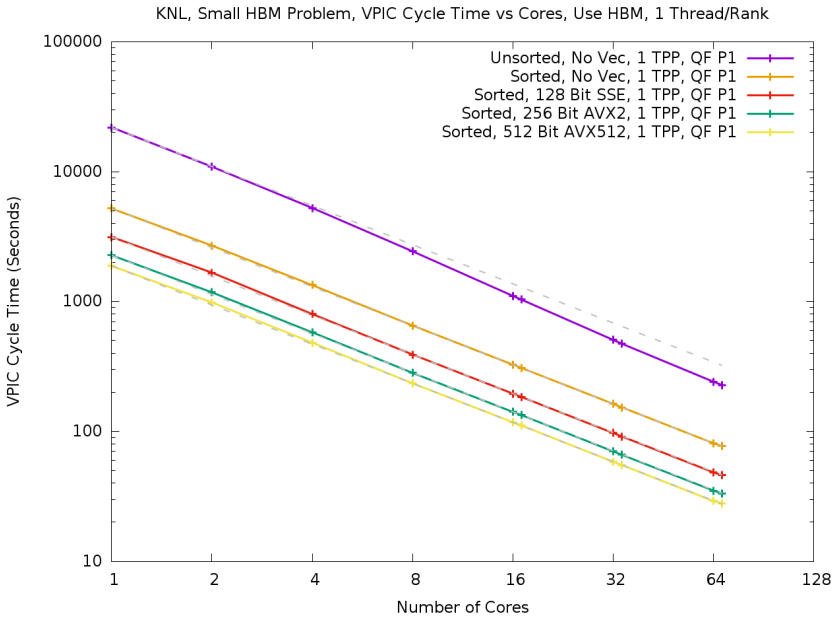


VPIC
Memory Bandwidth Limitations – Small Problem

DDR Memory

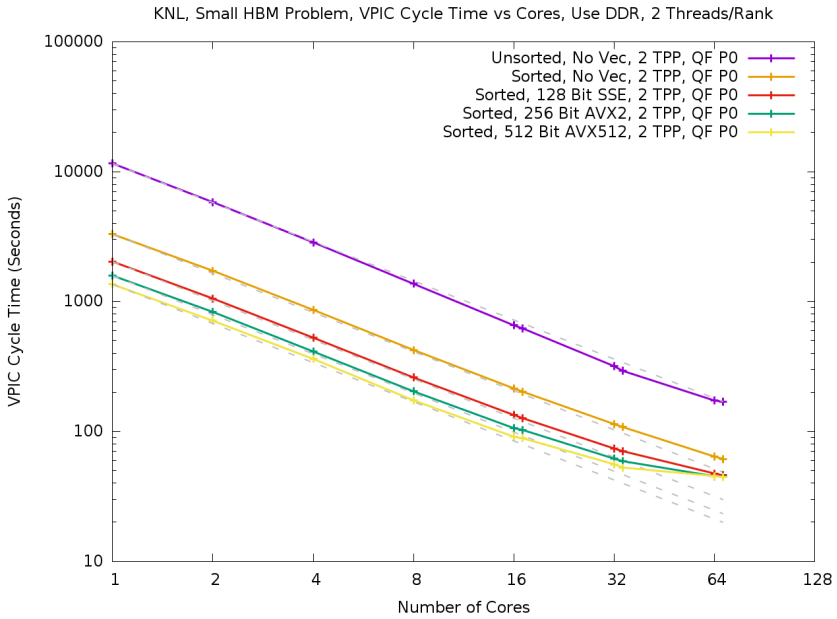


HBM Memory

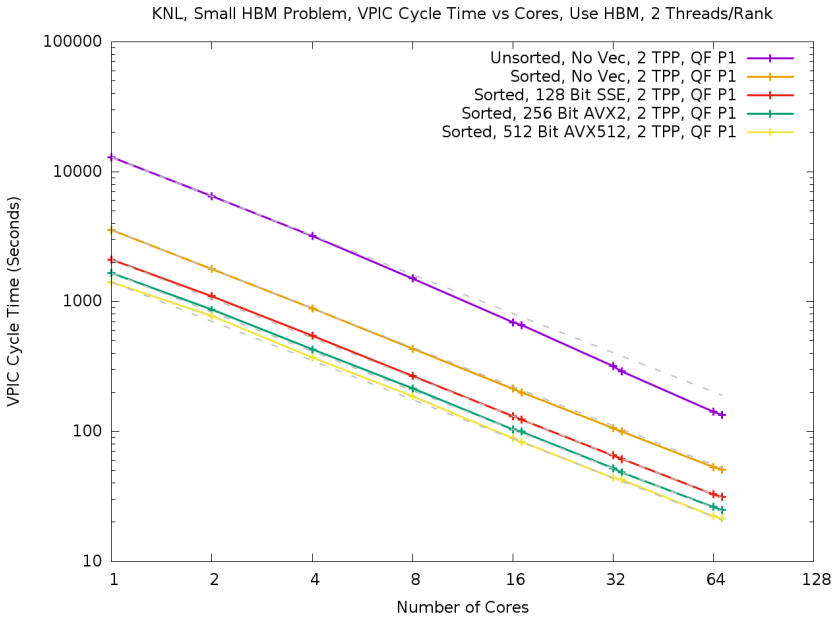


VPIC
Memory Bandwidth Limitations – Small Problem

DDR Memory

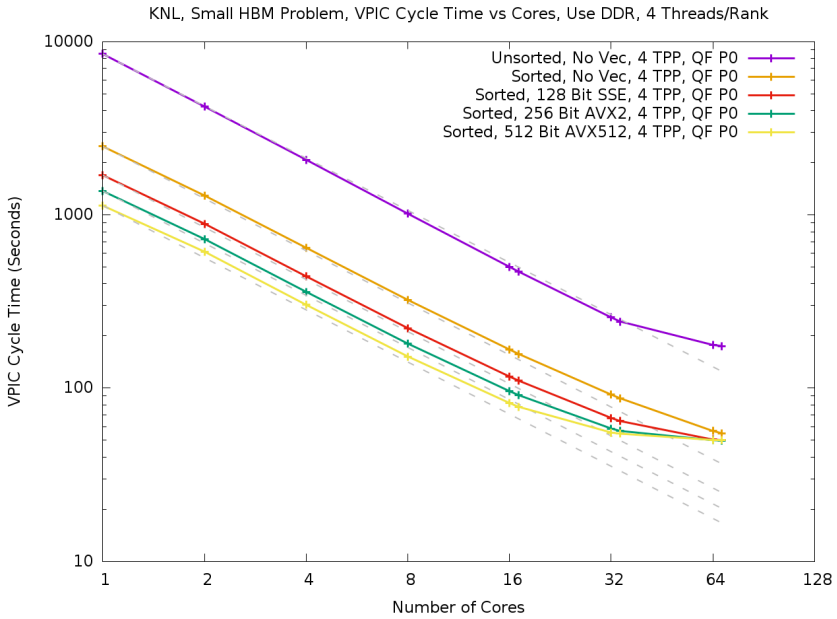


HBM Memory

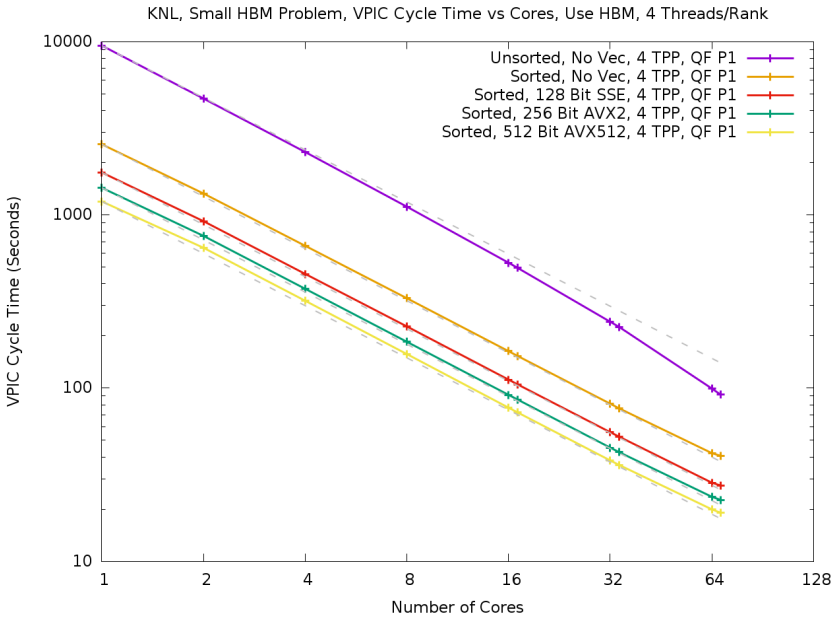


VPIC
Memory Bandwidth Limitations – Small Problem

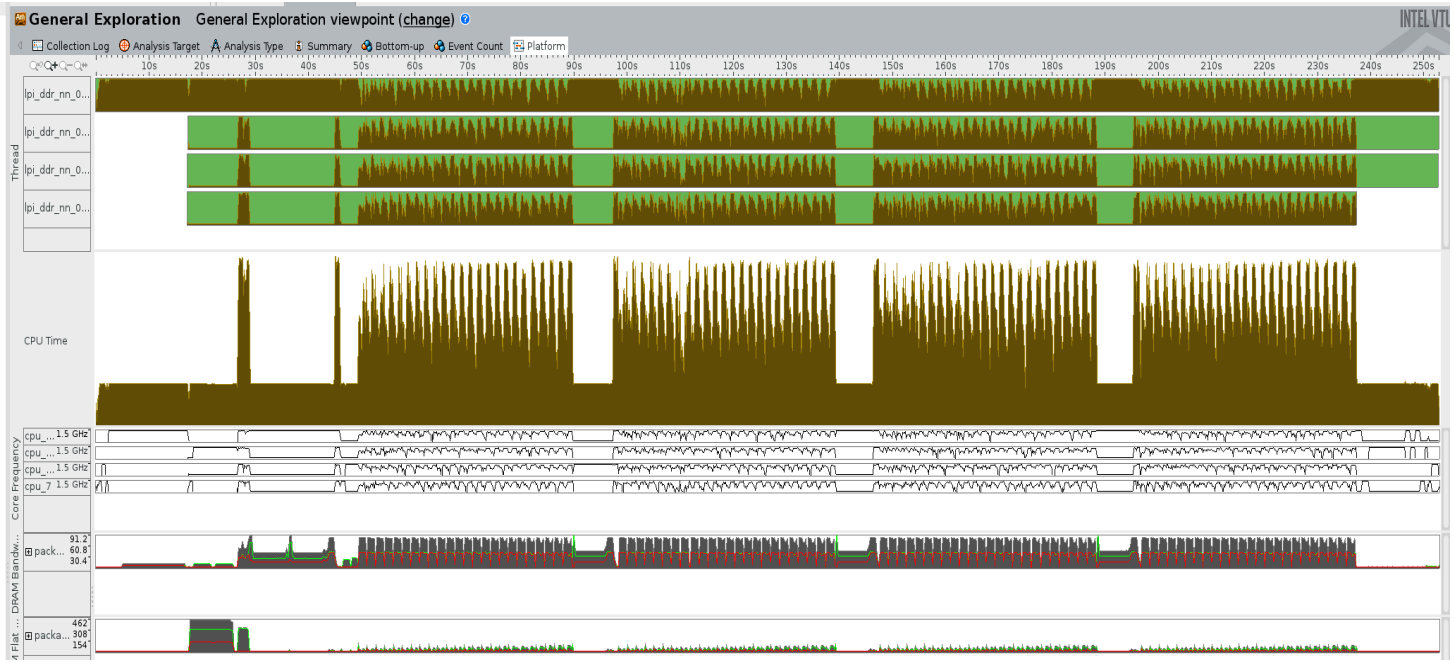
DDR Memory



HBM Memory



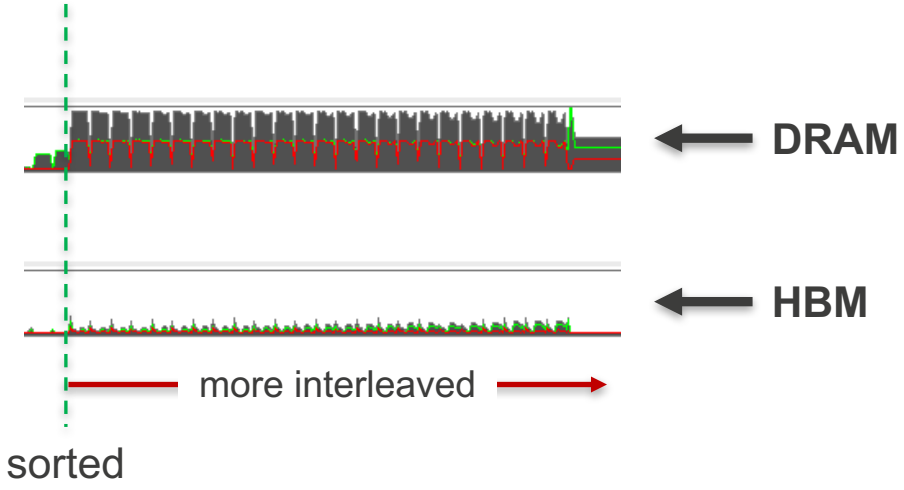
VPIC
Memory Bandwidth Utilization – Large Problem



DRAM

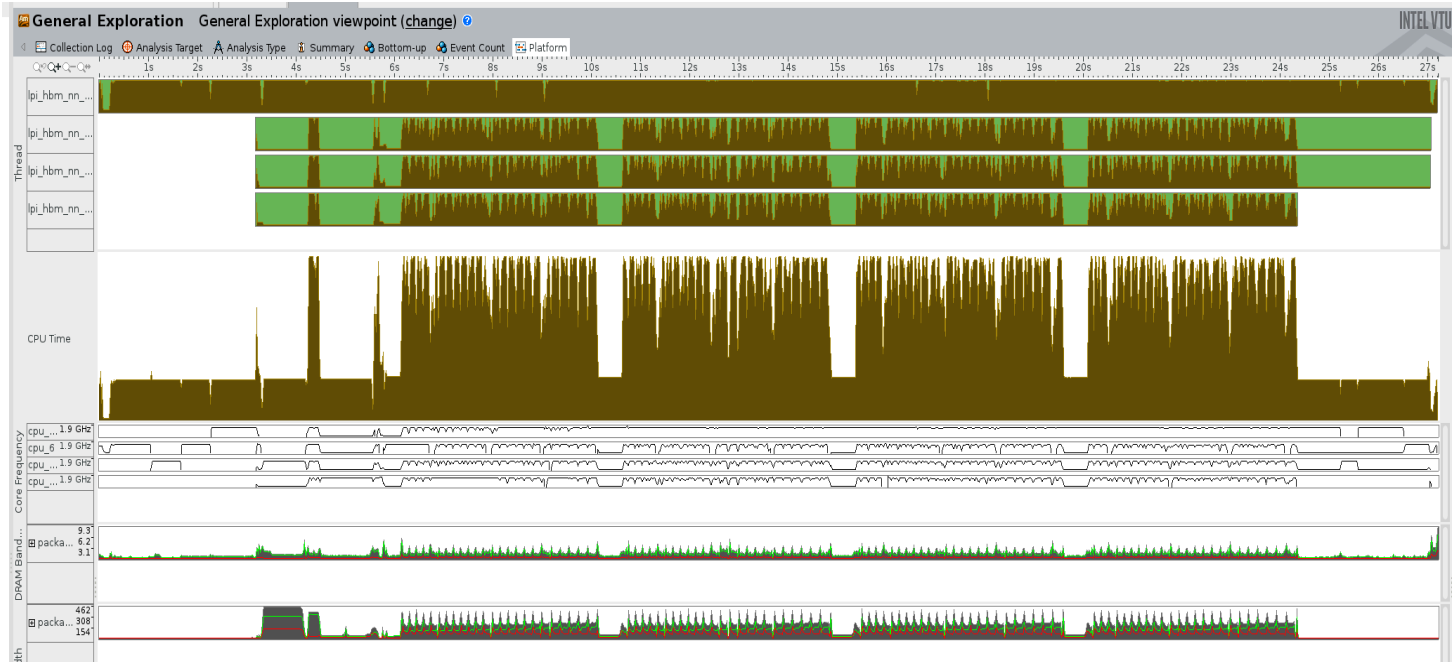
HBM

VPIC
Memory Bandwidth Utilization – Large Problem



- **Sorted particles essentially get full memory bandwidth utilization**
- **As interleaving occurs, HBM utilization increases due to field data accesses**

VPIC
Memory Bandwidth Utilization – Small Problem



VPIC
Single Node Summary

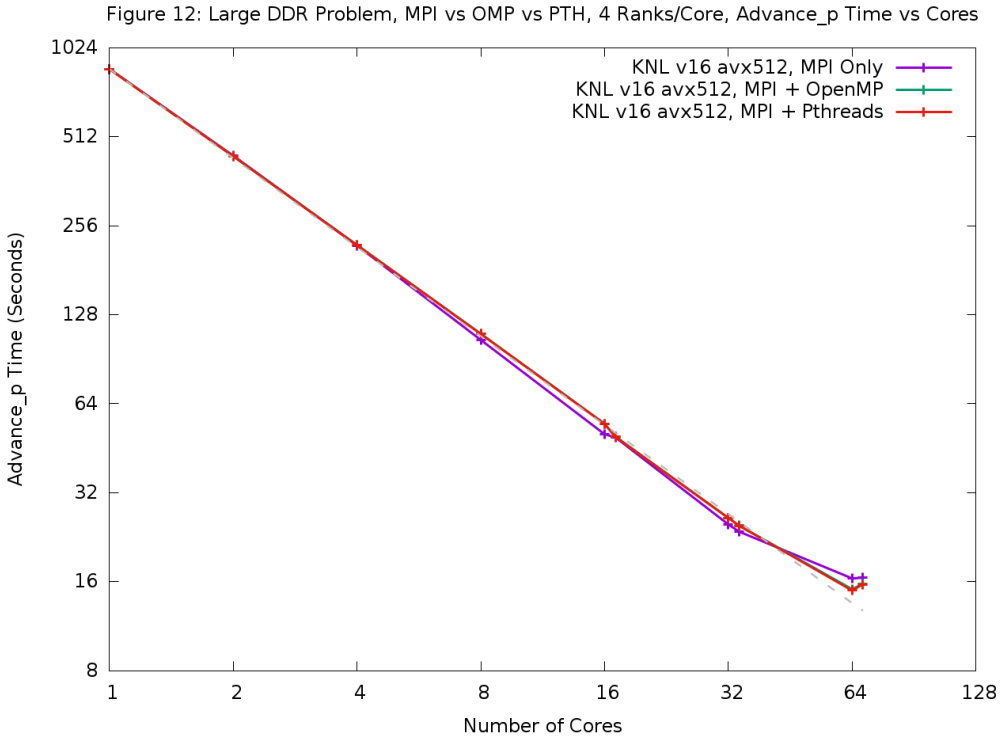
- **Large problem is currently memory-bandwidth limited**
- **Small problem is limited by instruction latency**
- **We have several strategies to improve HBM bandwidth utilization**
  - Re-implementation of transpose operation to use different intrinsics
  - Try gather/scatter support to re-order data by hand
  - Change data layout: Array-of-Structure-of-Vector (AoSoV)
- **Better HBM utilization may provide an impetus to try triple buffering approach**

MPI vs Pthreads vs OpenMP

OpenMP and Pthreads Performance

- **Figure 13 presents strong scaling study of performance of VPIC OpenMP and Pthreads implementations compared to MPI only performance when using V16 implementation which has highest computational intensity.**
- **Both OpenMP and Pthreads give slightly better performance, ~10 percent, for a fully utilized KNL processor running the large DDR problem.**
- **For other cases not plotted, such as for running the small HBM problem or running with the no vectorization implementation or the V4 implementation, OpenMP and Pthreads perform slightly worse than the MPI only case.**
- **These performance results are not understood. One speculation is that MPI only results in more contention for shared resources for the 4 execution threads/core case compared to either OpenMP or Pthreads.**

Figure 13: MPI vs OpenMP vs Pthreads, 4 Ranks/Core



Weak Scaling at Large Scale

Weak Scaling Results at Scale

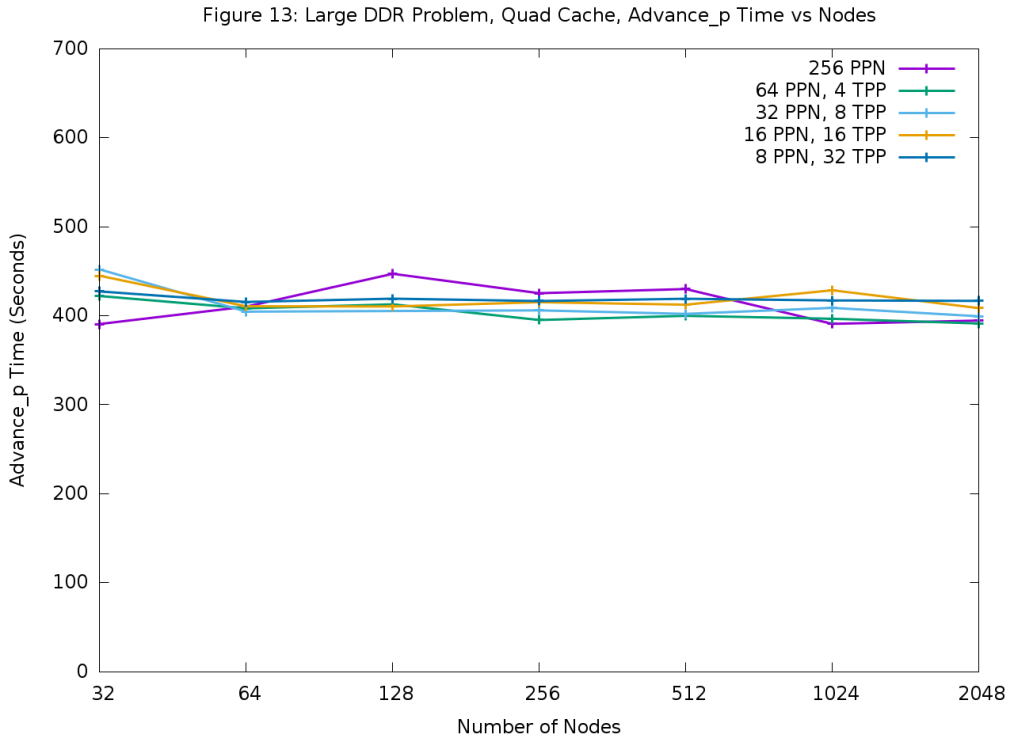
- **Figures 14 and 15 show performance as measured by time spent in the particle advance function versus number of nodes for several sets of KNL runs which were sized to use most of the DDR memory available on a node and were weak scaled across nodes.**
- **Runs were performed at 32, 64, 128, 256, 512, 1024 and 2048 nodes using 256 execution threads/node and 64 cores/node.**
- **Figure 14 shows runs performed using KNL quad cache mode.**
- **Figure 15 shows runs performed using KNL quad flat mode and using numactl –preferred=1 option to put any data structures in HBM that would fit. With 300 particles/cell, most or all of the grid sized data structures should fit in HBM.**
- **Both sets of runs included cases for MPI only and also MPI+Pthreads with 2, 4, 8, 16 and 32 Pthreads/rank.**

Weak Scaling Results at Scale (cont)

- Quad cache mode runs showed good weak scaling but larger variability in performance compared to runs in quad flat mode.
- Quad flat mode runs showed even better weak scaling than quad cache and much less variability in performance.
- The poor performance of MPI only for quad flat mode for 1024 and 2048 nodes is attributed to large memory allocations by Cray MPI getting preferentially allocated to HBM so that important VPIC data structures that reside in HBM for smaller node counts instead reside in DDR. Cray MPI use of HBM can be suppressed by an environment variable.
- The performance of quad flat runs is ~10 percent better on average than runs using quad cache mode.
- Using more than 4 Pthreads/rank in quad flat mode results in degraded performance for these runs. Better results may be achieved with more attention to controlling MPI rank placement and Pthread binding at higher thread counts/rank.

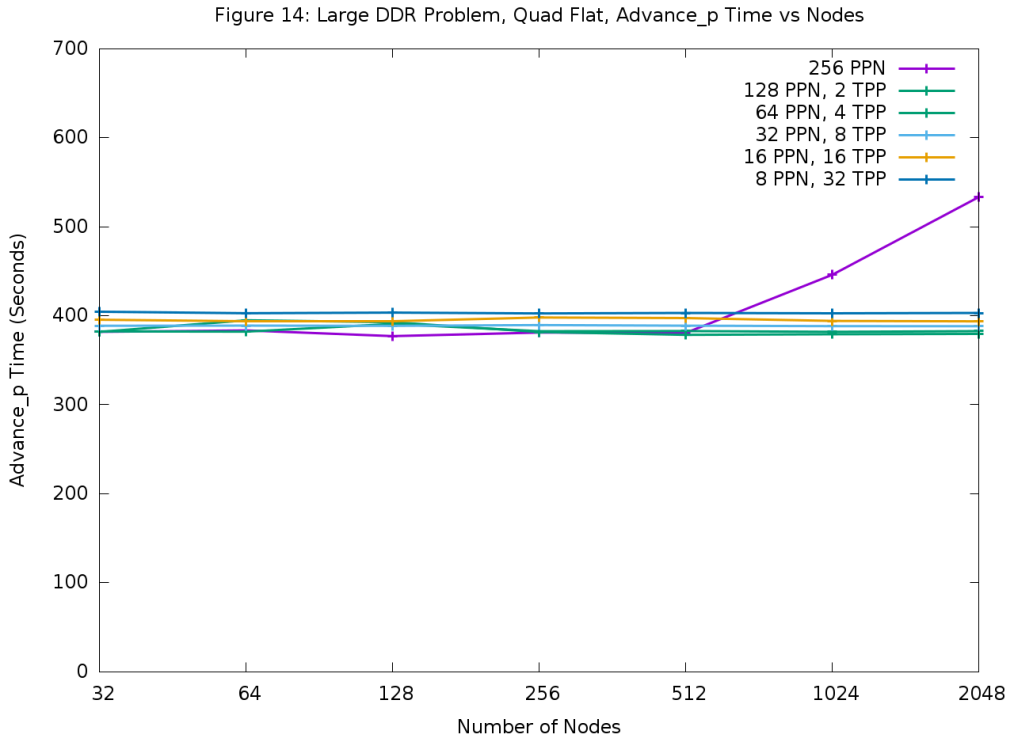
Slide 25

Figure 14: Weak Scaled DDR Problem, Quad Cache



Slide 26

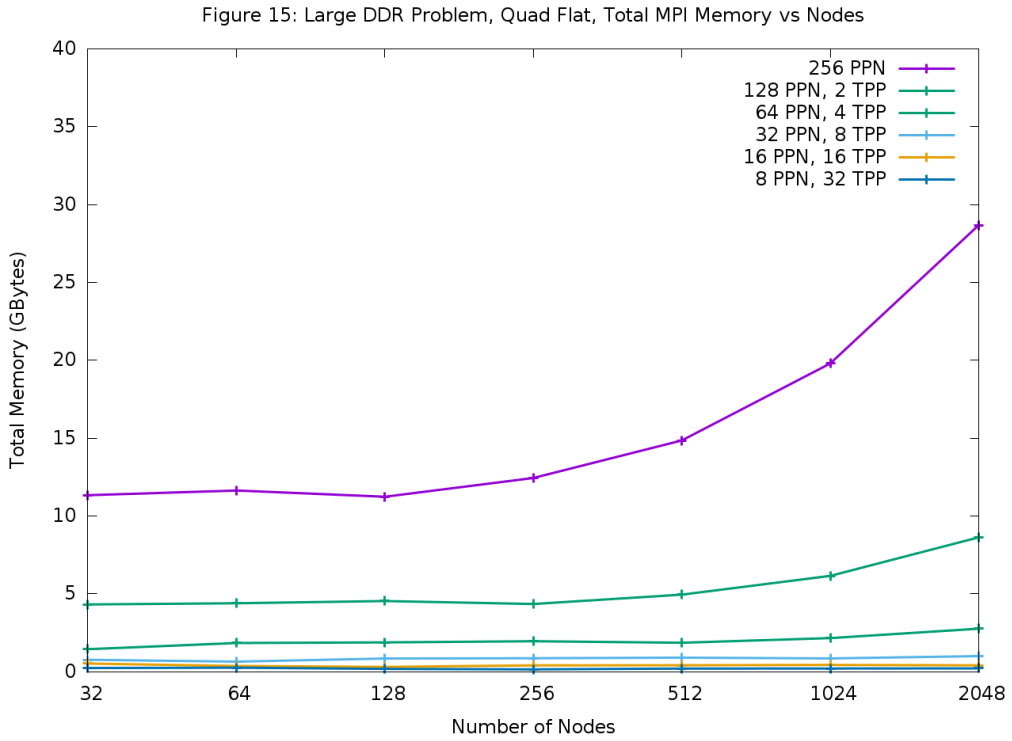
Figure 15: Weak Scaled DDR Problem, Quad Flat



Cray MPI Memory Usage on KNL

- Figure 16 shows total memory usage by Cray MPI for VPIC runs on KNL as a function of nodes for the quad flat runs shown in Figure 15.
- Cray MPI memory usage reported by Cray MPI using environment variable MPICH_MEMORY_REPORT=3.
- At 2048 nodes for MPI only case which uses 256 MPI ranks/node, MPI uses ~28 GB of memory which is ~30 percent of DDR memory per node on Trinity KNL nodes.
- At 2048 nodes, MPI memory usage scales nearly quadratically with total number of MPI ranks.
- VPIC gets significant performance benefit from using 4 execution threads per core versus 2. Other applications may as well.
- VPIC can avoid this issue by using MPI+threads. MPI only applications cannot. To run at large scale on Trinity KNL, MPI only applications may have to sacrifice some performance to allow fitting into memory.

Figure 16: Cray MPI Memory Usage vs Nodes



Slide 29

Acknowledgements

- **Work performed under the auspices of the U. S. Dept. of Energy by the Los Alamos National Security, LLC Los Alamos National Laboratory under contract DE-AC52-06NA25396 and supported by the LANL LDRD program.**