LA-UR-15-26182
Approved for public release; distribution is unlimited.

Title:	Progress on Optimizing VPIC for LLNL's Sequoia Platform
Author(s):	Nystrom, William David
Intended for:	International Conference on the Numerical Simulation of Plasmas, 2015-08-12/2015-08-14 (Golden, Colorado, United States)
Issued:	2015-08-04

Disclaimer:
Los Alamos National Laboratory, an affirmative action/equal opportunity employer,is operated by the Los Alamos National Security, LLC for the National NuclearSecurity Administration of the U.S. Department of Energy under contract DE-AC52-06NA25396. By approving this article, the publisher recognizes that the U.S. Government retains nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or to allow others to do so, for U.S. Government purposes. Los Alamos National Laboratory requests that the publisher identify this article as work performed under the auspices of the U.S. Departmentof Energy. Los Alamos National Laboratory strongly supports academic freedom and a researcher's right to publish; as an institution, however, the Laboratory does not endorse the viewpoint of a publication or guarantee its technical correctness.

Progress on Optimizing VPIC for LLNL's Sequoia Platform
Poster P1-6

Dave Nystrom
HPC-5
Los Alamos National Laboratory
wdn@lanl.gov

August 12, 2015

Overview

- VPIC is 3D explicit, relativistic, charge-conserving, electromagnetic PIC code developed at Los Alamos National Laboratory.
- VPIC was highly optimized for Roadrunner and was able to achieve 0.374 Pflops/s (single precision) overall and 0.488 Pflops/s (single precision) for the main particle processing loop.
- This level of optimization and performance for VPIC allowed for significant calculations to be performed with numbers of particles and cells that had not been achievable previously.
- 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 Plasmas, 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.

Porting and Optimizing VPIC for Sequoia

- Sequoia has approximately 15x the performance of Roadrunner. If VPIC could achieve comparable performance on Sequoia relative to peak as was obtained on Roadrunner, this would allow for calculations an order of magnitude more challenging to be performed. This provides a significant motivation to port and optimize VPIC on Sequoia. Additionally, optimizing VPIC to run efficiently on Sequoia will likely provide useful experience for optimizing VPIC for the Knights Landing portion of the LANL Trinity system.

- One of the optimization design decisions for VPIC was to use a vector class to encapsulate and optimize SIMD vector operations. This removes some of the burden on the compiler to produce good vectorized code by allowing the programmer to explicitly make those choices. VPIC has a portable implementation of the V4 vector class which is not optimal but can serve as a reference point and a guide for the implementation of tuned vector classes that use hardware intrinsics. VPIC used an Altivec implementation of the V4 vector class for Roadrunner and also has an SSE implementation that is used for Intel processors. The SSE implementation give about a 2x improvement in speed on Sandy Bridge processors but is likely not optimal because Sandy Bridge supports vectors of 8 floats but VPIC is currently limited to vectors of 4 floats. VPIC has been carefully designed and implemented to use single precision calculations that minimize data movement on carefully aligned data.

Porting and Optimizing VPIC for Sequoia (cont)

- A natural starting point in producing a version of VPIC which runs optimally on Sequoia is to take advantage of the V4 vector class. This allows optimizing the performance of VPIC by focusing on optimization of a relatively small, self-contained code component and also provides a useful test of the utility of this software design concept.

- However, Sequoia differs from Roadrunner in that the PowerPC A2 processor performs all calculations in double precision and does not support Altivec hardware intrinsics. Single precision calculations are converted to double precision, performed and then the result is converted back to single precision and stored as single precision. The VPIC Altivec V4 vector class used hardware intrinsics that supported various types such as ints, longs, floats and doubles and also supported bitwise operations. The PowerPC A2 processor uses QPX hardware intrinsics that only support vectors of 4 doubles and do not support bitwise operations. This is at odds with the single precision design of VPIC and made the implementation of a QPX version of the V4 class more challenging. It also precluded just porting the Altivec version of the V4 class to run on Sequoia.

- Sequoia currently has two options available for compilers, the 4.7.2 version of the GNU compilers and the IBM compilers. VPIC has been ported to Sequoia so that it can use either of these compilers. However, the GNU compilers do not have support for the QPX hardware intrinsic functions. So the use of a QPX version of the V4 vector class will be limited to compilation with the IBM compilers.

Porting and Optimizing VPIC for Sequoia (cont)

- The PowerPC A2 processor has 16 cores available for computation which each have 4 hardware threads. The theoretical peak flop rate for a PowerPC A2 core is 204.8 GFlops and the peak theoretical memory bandwidth is 42.6 GBytes/second. VPIC can be run on Sequoia using only MPI or MPI + Pthreads. When running using only MPI, the oversubscribe option must be used if running with 64 ranks per node. When running with MPI + Pthreads, VPIC is limited to a max of 3 Pthreads per rank because a 4th Pthread is not able to interrupt the thread associated with the MPI process that launched the Pthreads. Better performance is obtained with VPIC using MPI + Pthreads rather than using only MPI. The results reported in this work used the MPI + Pthreads model for running on Sequoia.

- The first step in developing a QPX version of the V4 vector class is making a portable V4 vector class which stores its member data in double precision. Figures 1 and 2 show performance comparisons between the double precision and single precision versions of the portable V4 class. When compiled with GNU compilers, the single precision portable V4 implementation is faster than the double precision implementation for the cases of 1, 2 and 3 Pthreads per rank for each run. The double precision version is about 13 percent slower than the single precision version. Figure 2 shows the comparison when using the IBM compilers. For the case of 2 and 3 threads per rank, the IBM compiler produces code for the single precision portable V4 case which runs slower than expected. At this time, the cause of this slowdown for 2 and 3 threads at 8 and 16 cores is not known.

Porting and Optimizing VPIC for Sequoia (cont)

- Figures 3 and 4 show performance comparisons for the same code of using either IBM or GNU compilers. For the double precision version of V4, the IBM compiler produces slightly slower code for the case of 3 threads per rank and 16 ranks, 37 seconds versus 35 seconds. Finally, for the case of 16 ranks and 3 threads per rank, the double precision portable V4 compiled with the IBM compilers runs in 37 seconds while the single precision portable V4 compiled with GNU compilers runs in 31 seconds. This provides the worst case comparison between the double precision V4 and the single precision V4. The double precision V4 with IBM compilers is 19 percent slower than the single precision V4 with GNU compilers. The QPX V4 must recover this performance penalty to make the implementation worthwhile.

- Figure 5 shows a comparison of the QPX hardware intrinsics version of V4 versus the double precision portable V4 where the time spent in the main particle processing loop of VPIC is measured. Both cases used IBM compilers. The QPX V4 runs in 17 seconds. The double precision portable V4 runs in 37 seconds. So the speedup for QPX V4 is greater than 2x. Figure 6 shows a plot of the flop rate for the main particle advance loop as measured by the Blue Gene Q hardware counters. The QPX V4 runs at a rate of approximately 30 Gflops. The double precision portable V4 runs at a rate of approximately 17.5 Gflops. To run at the same level of performance as on Roadrunner, the QPX V4 needs to run the main particle loop at a flop rate of 80 - 100 Gflops.

Porting and Optimizing VPIC for Sequoia (cont)

- Figure 7 compares the time spent in the main particle loop of VPIC for the QPX V4 implementation to that of the original single precision portable V4. For 16 ranks and 3 threads per rank, the QPX V4 case runs in 17 seconds and the original single precision portable V4 runs in 31 seconds. Making the comparison this way, the QPX V4 implementation has nearly but not quite achieved a 2x speedup.

Performance Penalty of Double Precision Portable V4 Class Using GNU Compilers

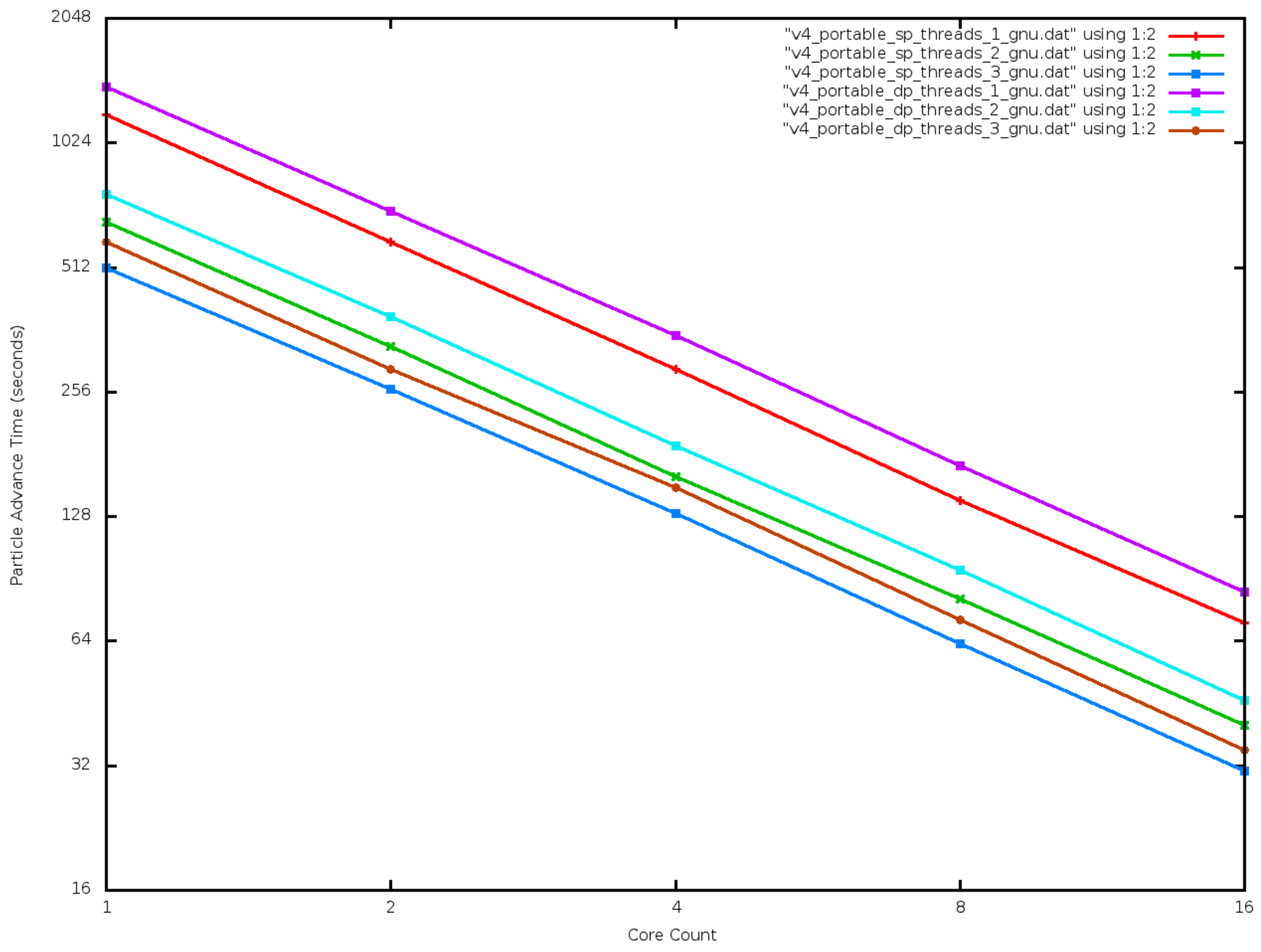

Figure 1: This plot shows the relative performance of the portable V4 vector class when data storage is changed to double precision as a prelude to being able to implement a V4 class which uses the Blue Gene Q QPX hardware intrinsics. This comparison uses GNU compilers for compilation. Results are presented for 1, 2 and 3 threads per MPI rank. There is a performance penalty of about 13 percent for the double precision implementation.

Performance Penalty of Double Precision Portable V4 Class Using IBM Compilers

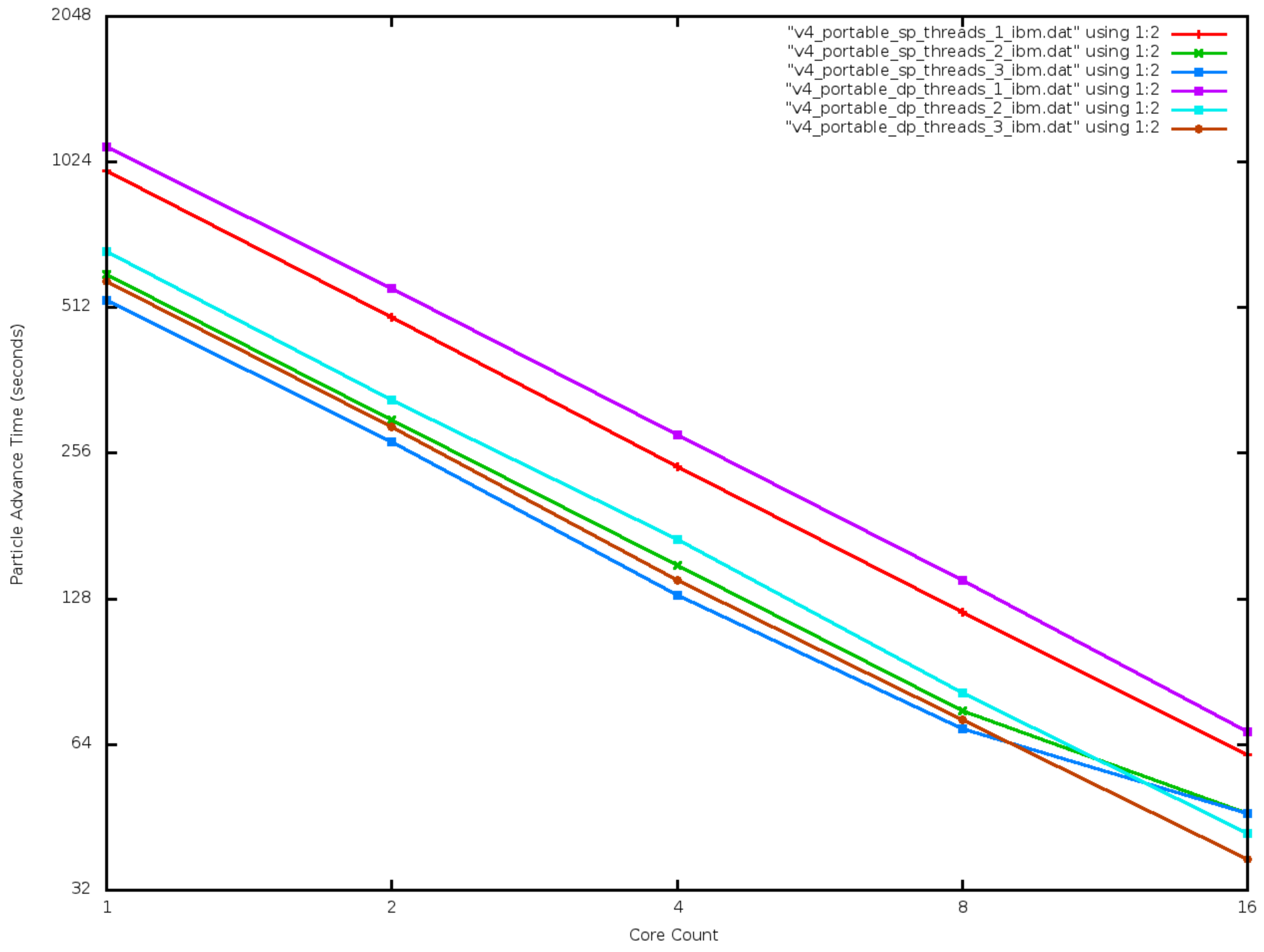

Figure 2: This plot shows the relative performance of the portable V4 vector class when data storage is changed to double precision as a prelude to being able to implement a V4 class which uses the Blue Gene Q QPX hardware intrinsics. This comparison uses IBM compilers for compilation. Results are presented for 1, 2 and 3 threads per MPI rank. There is an unexplained issue with performance of the single precision V4 class for 8 and 16 ranks and 2 and 3 threads per rank.

Performance Comparison of IBM and GNU Compilers for Single Precision V4

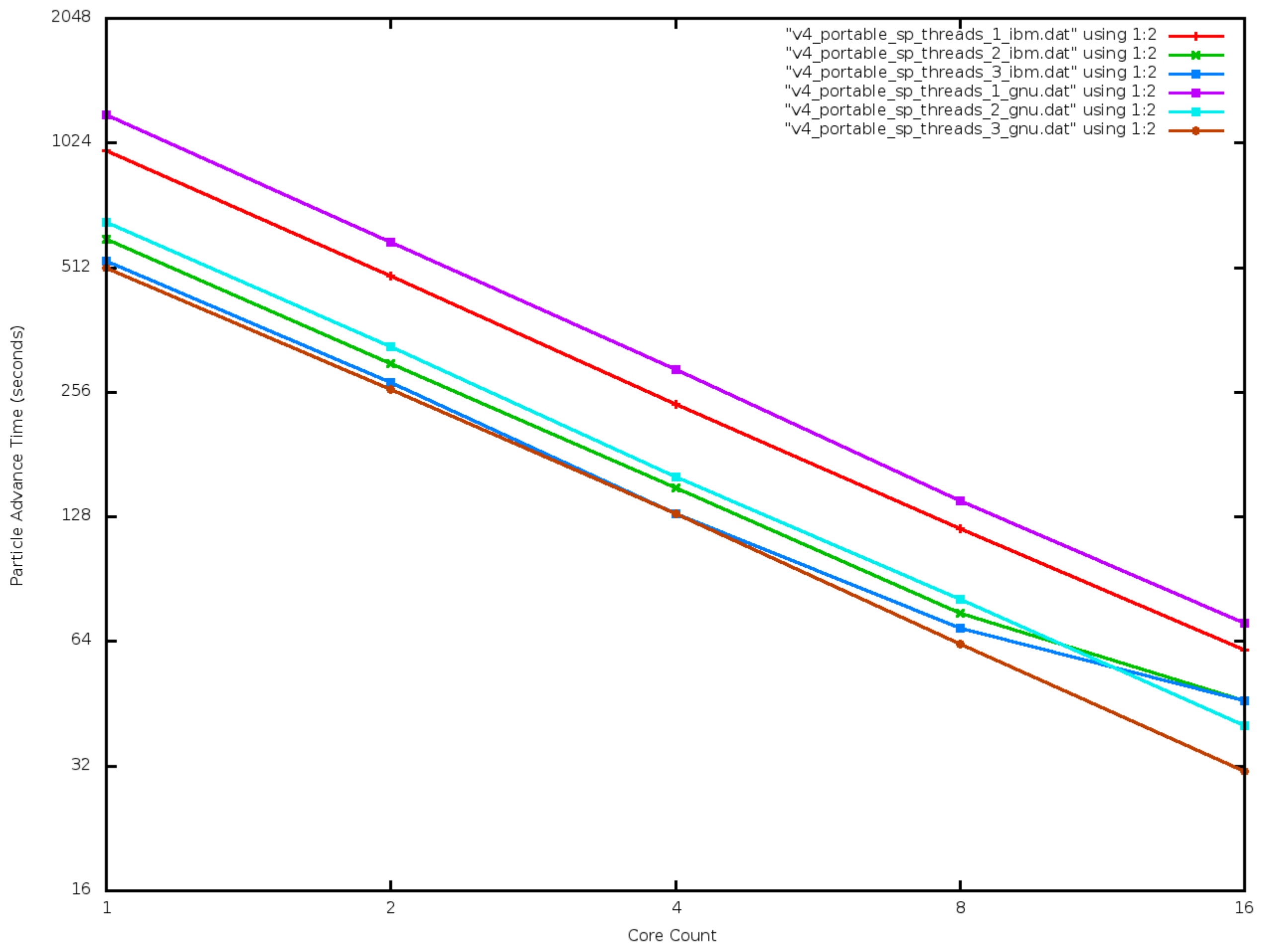

Figure 3: This plot shows the relative performance of the single precision portable V4 vector class for the two different compilers. The IBM compiler produces faster code except for the cases of 8 and 16 ranks with 2 and 3 threads per rank. This performance looks puzzling and has not yet been explained.

Performance Comparison of IBM and GNU Compilers for Double Precision V4

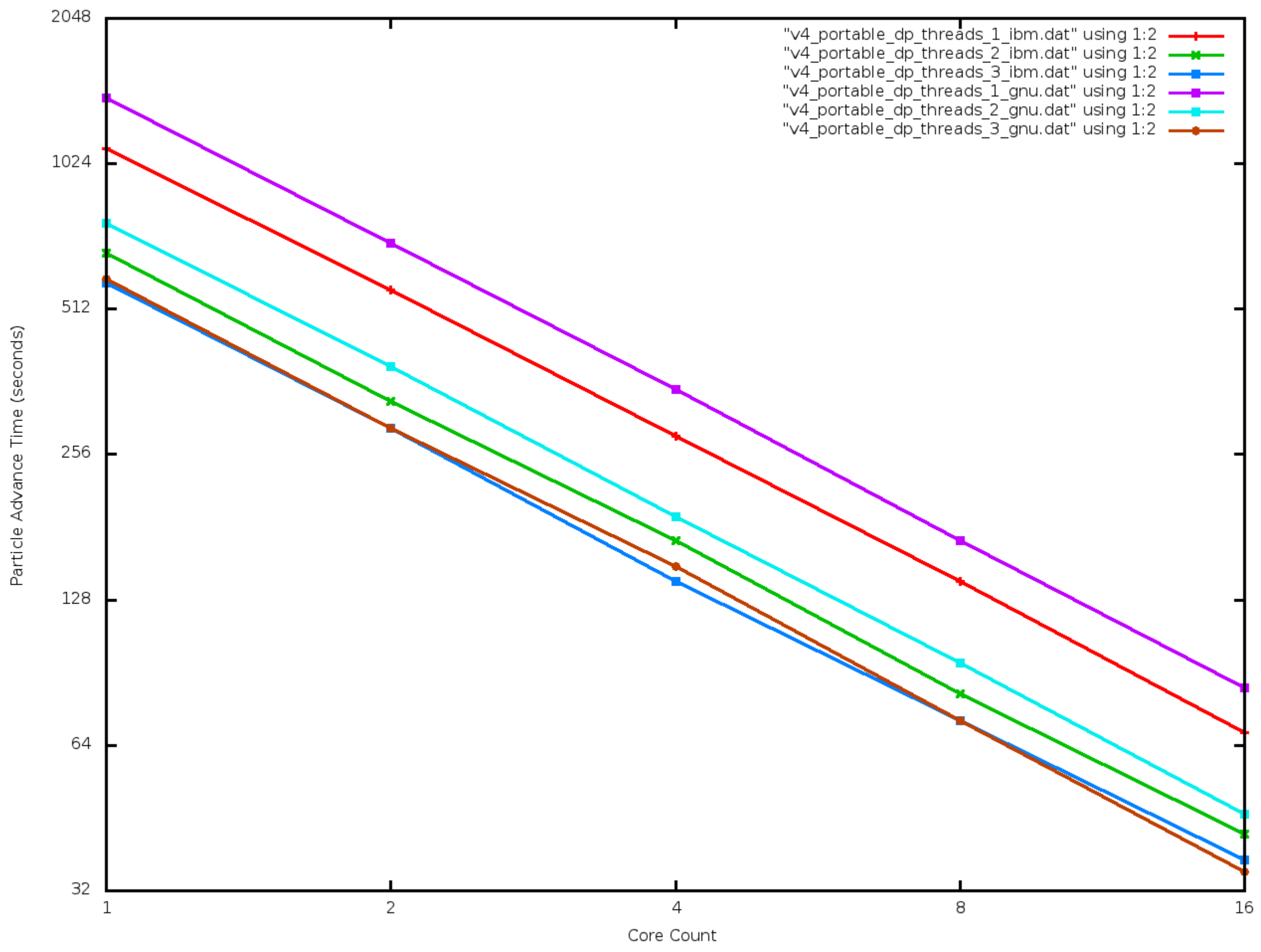

Figure 4: This plot shows the relative performance of the double precision portable V4 vector class for the two different compilers. The IBM compiler produces faster coded for the cases of 1 and 2 threads per MPI rank. The two compilers produce comparable results for the case of 3 threads per MPI rank with GNU slightly faster at 16 ranks and 3 threads per rank.

Performance of Main Particle Loop with QPX V4 versus Portable DP V4, Time

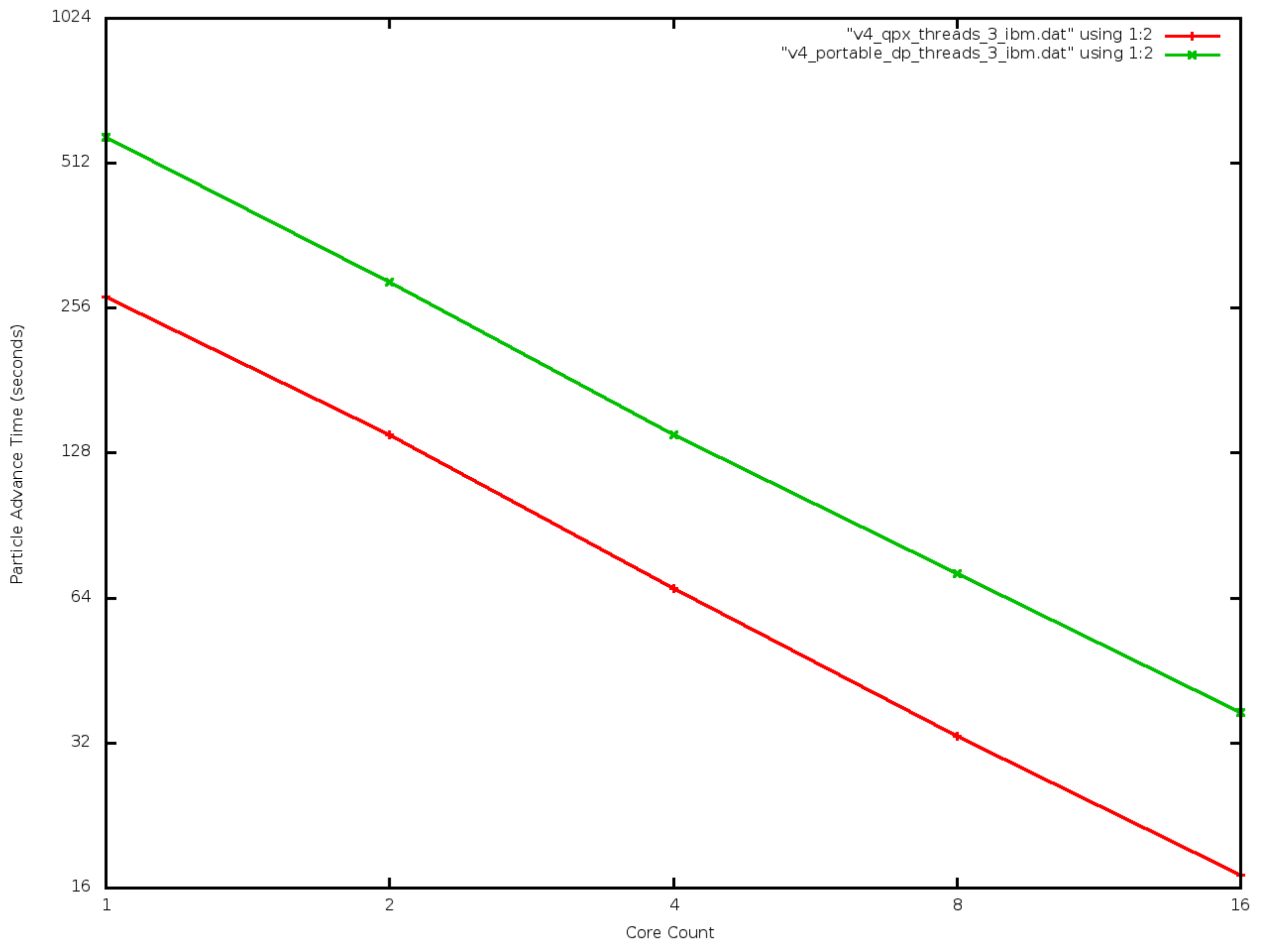

Figure 5: This plot shows the time spent in the main particle loop of VPIC versus the number of ranks or cores. These runs used 3 threads per MPI rank. At 16 ranks the time spent in the main particle loop is 37 seconds for the portable V4 implementation and 17 seconds for the QPX hardware intrinsics V4 implementation. The speedup is a little more than 2x.

Performance of Main Particle Loop with QPX V4 versus Portable DP V4, Flops

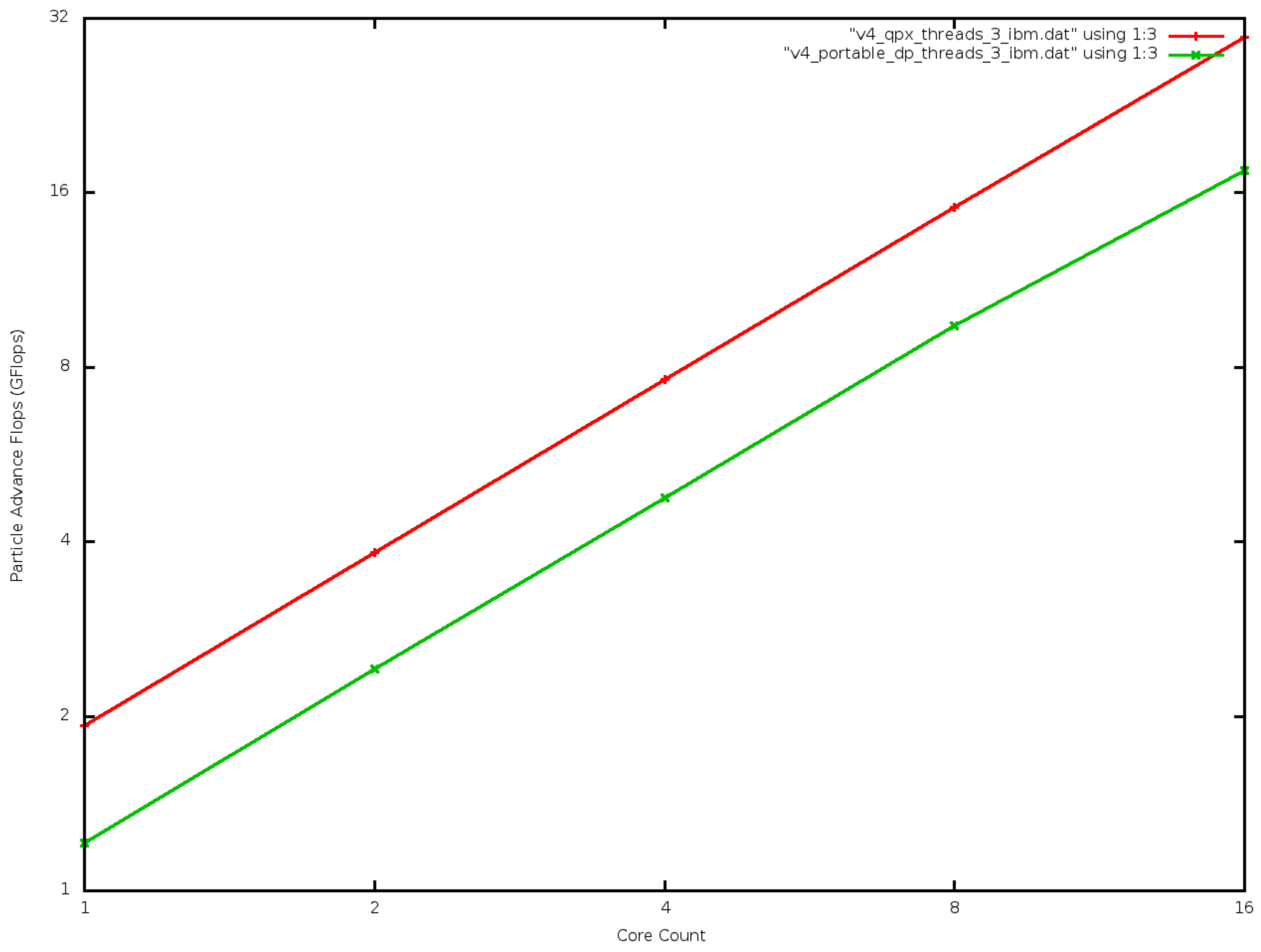

Figure 6: This plot shows the flop rate versus core or rank for the main particle loop of VPIC. The QPX V4 implementation achieves about 30 Gflop/s or about 15 percent of the theoretical peak for a Blue Gene Q processor. The flop rate was measured using Blue Gene Q hardware counters.

Performance of Main Particle Loop with QPX V4 versus Best Portable V4

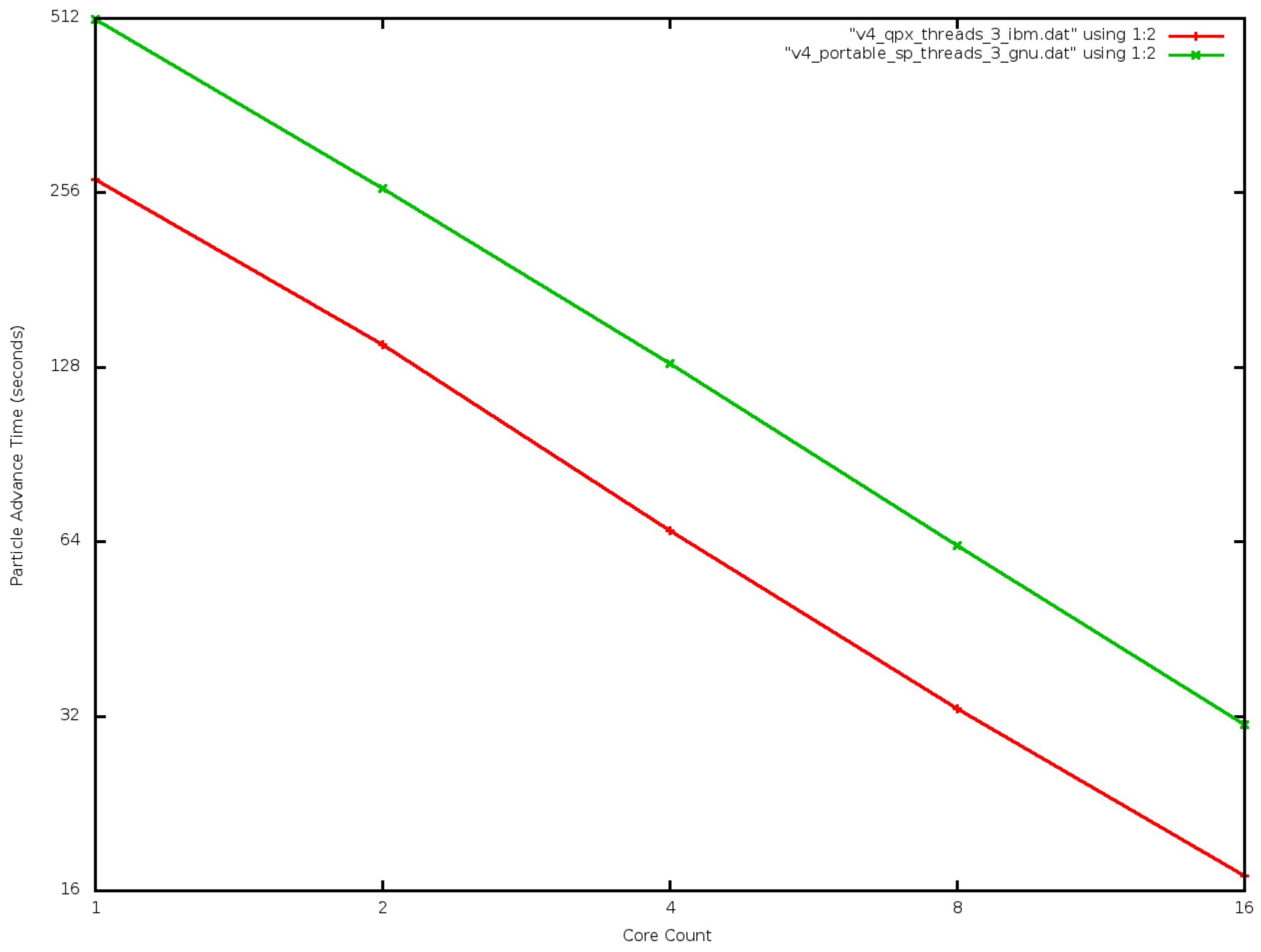

Figure 7: This plot shows the time spent in the main particle loop of VPIC versus the number of ranks or cores using 3 threads per rank. The QPX V4 implementation is compared to the original single precision portable V4 implementation which represents the performance achievable without performing the QPX V4 implementation. The speedup in this comparison is slightly less than 2x.

Conclusions and Future Work

- The strategy of implementing a version of the VPIC V4 vector class that uses QPX hardware intrinsic functions has been successful in providing a significant speedup of the VPIC single node performance on Sequoia, nearly a 2x speedup.

- There is still work to be done on improving the QPX V4 vector class and hopefully this will result in more speedup of the VPIC single node performance on Sequoia because it is currently only achieving 15 percent of the theoretical peak. There is still functionality in the QPX V4 class implementation that is not implemented with QPX intrinsics but hopefully can be even though the functionality provided by QPX is significantly less than that of Altivec.

- When the VPIC single node performance on Sequoia has been maximized, the multi-node performance will then need to be evaluated at scale. And when VPIC is performing well at scale, then new and larger physics problems can be modeled on sequoia.

Acknowledgements

Work supported by the Advanced Simulation and Computing program of the U.S. Department of Energy's NNSA. Los Alamos National Laboratory is operated by Los Alamos National Security, LLC for the NNSA. In addition, the author would like to thank Brian Albright and Ben Bergen of LANL for help in learning how to build and run VPIC, to John Gyllenhaal at LLNL and Roy Musselman of IBM for answering questions and providing assistance on learning how to use Sequoia effectively and finally to Bob Walkup of IBM for assistance in learning how to use the QPX hardware intrinsics and providing tools to measure Blue Gene Q hardware counters including helping to interpret the results from VPIC runs.