Performance Metrics and Application Experiences on a Cray CS300-AC^{TM}Cluster Supercomputer Equipped with Intel$^{®}$ Xeon Phi^{TM} Coprocessors

Vincent C. Betro, Robert P. Harkness*, Bilel Hadri, Haihang You,
Ryan C. Hulguin, R. Glenn Brook, and Lonnie D. Crosby
University of Tennessee National Institute for Computational Sciences
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA
Email: vbetro@utk.edu, harkness@sdsc.edu, bhadri@utk.edu, hyou@utk.edu,*
ryan-hulguin@tennessee.edu, rgbrook@utk.edu, lcrosby1@utk.edu
**Posthumous publication of work with Beacon Project at AACE/NICS*

***Abstract*—Given the growing popularity of accelerator-based supercomputing systems, it is beneficial for applications software programmers to have cognizance of the underlying platform and its workings while writing or porting their codes to a new architecture. In this work, the authors highlight experiences and knowledge gained from porting such codes as ENZO, H3D, GYRO, a BGK Boltzmann solver, HOMME-CAM, PSC, AWP-ODC, TRANSIMS, and ASCAPE to the Intel Xeon Phi architecture running on a Cray CS300-AC^{TM}Cluster Supercomputer named Beacon. Beacon achieved 2.449 GFLOP/W in High Performance LINPACK (HPL) testing and a number one ranking on the November 2012 Green500 list [1]. Areas of optimization that bore the most performance gain are highlighted, and a set of metrics for comparison and lessons learned by the team at the National Institute for Computational Sciences Application Acceleration Center of Excellence is presented, with the intention that it can give new developers a head start in porting as well as a baseline for comparison of their own code's exploitation of fine and medium-grained parallelism.**

***Keywords*-Intel Xeon Phi, MIC, accelerators, performance metrics, CS300-AC**

I. Introduction

Given the growing popularity of accelerator-based supercomputing systems, it is beneficial for applications software programmers to have cognizance of the underlying platform and its workings while writing or porting their codes to a new architecture [2]. This is becoming even more relevant with the variety of "accelerator" architectures available today which include GPGPUs, Intel$^{®}$ Xeon Phi^{TM} s, FPGAs, and a plethora of other choices. In this work, the authors highlight experiences and knowledge gained from porting such codes as ENZO, H3D, GYRO, a BGK Boltzmann solver, HOMME-CAM, PSC, AWP-ODC, TRANSIMS, and ASCAPE to the Intel$^{®}$ May Integrated Core (MIC) architecture running on a Cray CS300-AC^{TM}Cluster Supercomputer named Beacon.

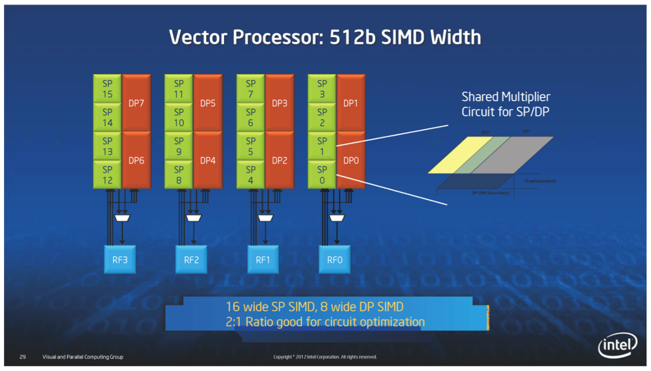

Vector Processor: 512b SIMD Width
SP 15
SP 14
SP 13
SP 12
DP7
DP6
SP 11
SP 10
SP 9
SP 8
DP5
DP4
SP 7
SP 6
SP 5
SP 4
DP3
DP2
SP 3
SP 2
SP 1
SP 0
DP1
DP0
RF3
RF2
RF1
RF0
Shared Multiplier Circuit for SP/DP
16 wide SP SIMD, 8 wide DP SIMD
2:1 Ratio good for circuit optimization
intel

Figure 1: The Intel Xeon Phi Coprocessor's Wide Vector Unit [5]

One reason why the Intel Xeon Phi coprocessor is chosen for this work is that the lessons learned on it regarding vectorization and memory alignment, among other things, apply directly not just to other architectures but also to bolstering performance on the Intel$^{®}$ Xeon$^{®}$ CPU itself. For instance, while the vector width on an Intel Xeon Phi is twice that of an Intel Xeon (512 bits vs 256 bits), the method of harnessing the fused multiply add functionality along with the vector processing functionality can still yield eight times the performance on the Intel Xeon (and sixteen times the performance on the Intel Xeon Phi). For more information on how this vector processing works, please see Figure 1.

Additionally, the Intel MIC architecture is x86 based, which means that third party libraries and conventional languages port directly to the architecture, though further optimization might be necessary to achieve the best performance. This also allows users to run natively on the Intel Xeon Phi itself, saving the need for using the PCIe bus to transfer "kernels" over to the accelerator causing

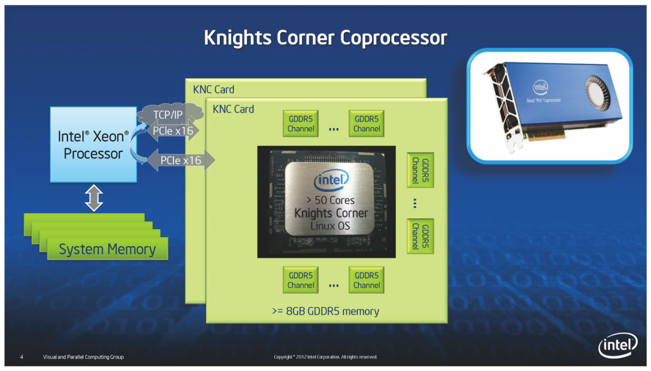


Figure 2: The Intel Xeon Phi Coprocessor within System Architecture [5]

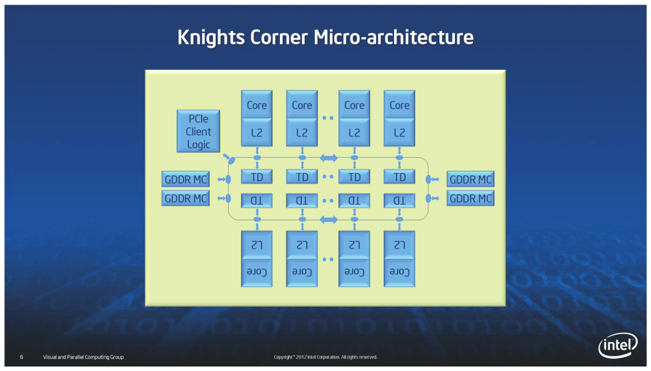
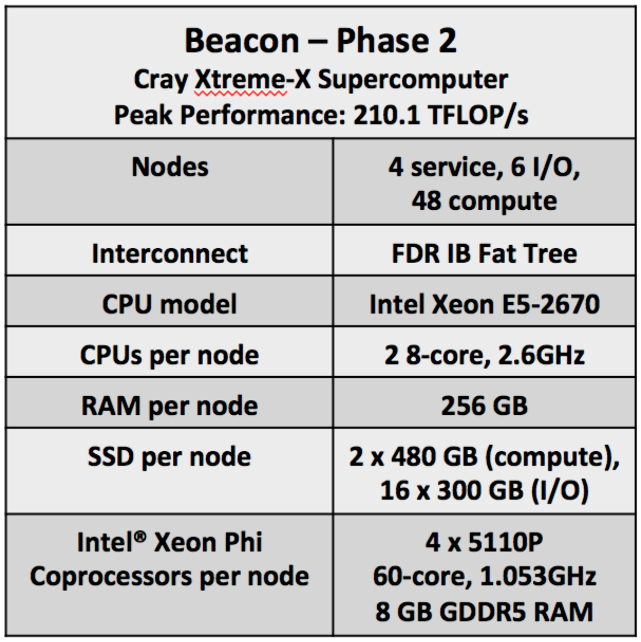


Figure 3: The Intel Xeon Phi Coprocessor On-Chip Architecture [5]



Beacon – Phase 2
Cray Xtreme-X Supercomputer
Peak Performance: 210.1 TFLOP/s

Nodes	4 service, 6 I/O, 48 compute
Interconnect	FDR IB Fat Tree
CPU model	Intel Xeon E5-2670
CPUs per node	2 8-core, 2.6GHz
RAM per node	256 GB
SSD per node	2 x 480 GB (compute), 16 x 300 GB (I/O)
Intel® Xeon Phi Coprocessors per node	4 x 5110P 60-core, 1.053GHz 8 GB GDDR5 RAM

Figure 4: The Beacon Project System Specifications

latency over the bus to often outstrip the gains made with the extra threads and cores of the Intel Xeon Phi (see Figure 2 for full schematic of architecture). While this offload method can still be employed where it has been optimized on the Intel Xeon Phi , it is not the only model. In fact, a heterogeneous approach with the Intel Xeon Phi and Intel Xeon working simultaneously on different facets of the problem and likely even employing hybrid MPI and OpenMP parallelism seems to be one of the most promising paths for applications programmers and uses the Intel Xeon Phi to its greatest potential. The Intel Xeon Phi even has an On Chip Interconnect (OCI) which allows MPI tasks to be run across cores on the card along with threading (see Figure 3.

II. The Beacon Project

Beacon, an experimental cluster within the University of Tennessee Application Acceleration Center of Excellence (AACE), was funded through a NSF Strategic Technologies for Cyberinfrastructre (STCI) grant and the State of Tennessee. Beacon was first deployed in a 16 node cluster with two MICs per node in August 2012. In April 2013, the production Beacon machine was deployed, consisting of 48 compute nodes, 6 I/O nodes, 2 login nodes, and a management node (see Figure 4). Each compute node is based on two dual socket Intel Xeon E5-2670 processors (with 256 GB of RAM) and four Intel Xeon Phi 5110P coprocessors, each having 8 GB of memory. Beacon has an FDR InfiniBand interconnect providing 56 Gb/s of bi-directional bandwidth and contains 960 GB of SSD storage per node; each I/O node provides access to an additional 4.8 TB of SSD storage. In total, Beacon has 768 conventional cores and 11,520 coprocessor cores that provide over 210 TFLOP/s of combined computational performance, 12 TB of system memory, 1.5 TB of coprocessor memory, and over 73 TB of SSD storage. This cluster is being used to prepare NSF application teams and their applications for future systems based on the Intel MIC architecture.

Additionally, Beacon has proven to be tops in the category of green computing, which is one reason for the push behind accelerator/coprocessor technologies. In October 2012, Beacon achieved 2.449 GFLOP/W in High Performance LINPACK (HPL) testing and a number one ranking on the November 2012 Green500 list [1]. This represents a large step forward in green computing from previous BlueGene systems as well as an obvious call for programmers to employ accelerator and coprocessor friendly coding paradigms.

III. Ported and Optimized Applications

The following applications have been being ported and optimized over the first year of the Beacon project by Dr. Robert Harkness (ENZO); Dr. Homayun Karimabadi (H3D); General Atomics, Dr. Mark Fahey and Dr. Vincent

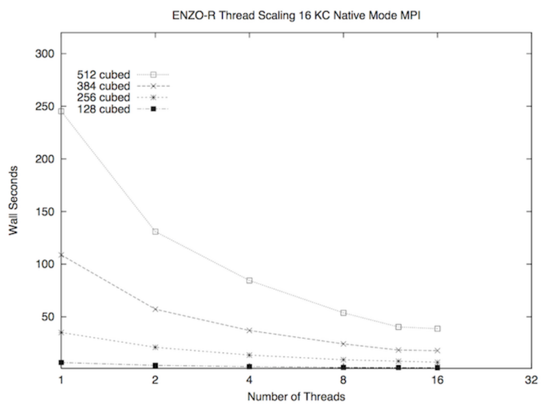


Figure 5: ENZO-R Thread Scaling vs walltime on an Intel Xeon Phi Coprocessor running in Native Mode with 16 MPI ranks

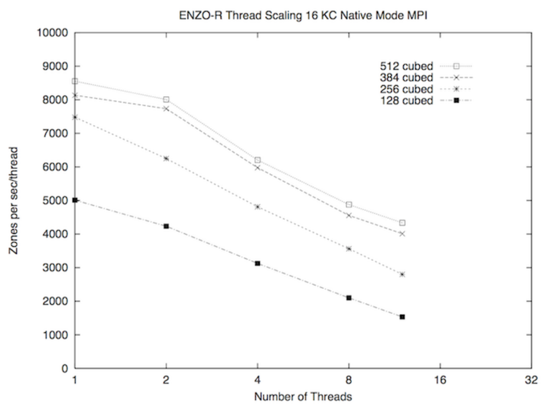


Figure 6: ENZO-R Thread Scaling vs zones/second on an Intel Xeon Phi Coprocessor running in Native Mode with 16 MPI ranks

Betro (GYRO); Ryan Hulguin and Rob VanDerWingaart (Boltzmann BGK). They represent a variety of optimizations including hybrid MPI/OpenMP parallelism, MPI over the coprocessor OCI, and memory alignment.

A. *Enzo*

The Enzo multi-physics hydrodynamic astrophysical code is a freely-available, community-developed adaptive mesh refinement simulation routine, designed for rich calculations [6]. The Enzo code used in this paper was ported and optimized for the Intel Xeon Phi Coprocessor by Dr. Robert Harkness of the University of California at San Diego before his untimely passing in January 2013 from a short bout with cancer at age 56 [15].

Despite Enzo being over 1 million lines of code, it was instantly ported to the Intel Xeon Phi with a simple addition of "-mmic" to the compile line. Also, it was able to use the HDF5 and HYPRE libraries that had been compiled for the Intel Xeon Phi in the same manner. Then, hybrid MPI/OpenMP parallelism was added in several routines, allowing for scalability and optimization for the Intel Xeon Phi .

By using multiple MPI tasks per processor and several threads per MPI rank, Enzo was shown to scale well as threads were increased, especially for larger problem sizes. Information about zones/second and walltimes at each thread count for each case run on 16 MPI ranks on one Intel Xeon Phi coprocessor can be found in Figures 5 and 6.

B. *H3D*

The H3D magneto hydrodynamics code used in this publication was ported and optimized by Dr. Homayun Karimabadi at the University of California at San Diego [7]. This was again done simply with adding "-mmic" to the compile line and running in MPI across the OCI on the Intel Xeon Phi coprocessor.

H3D is used in global modeling of solar wind interaction with the Earth‘s magnetosphere. Currently global simulations are generally based on single-fluid magnetohydrodynamics (MHD). MHD simulations are useful in studies of the global dynamics of the magnetosphere in so far as predicting substorms and other global events. However, spacecraft observations have established that most critical plasma processes regulating mass and energy transfer in the magnetosphere take place at thin boundaries/discontinuities between regions of geospace where ion kinetic effects control the physics. Thus, it is desirable to retain the full ion kinetic effects while treating the electrons as fluid, which requires massively parallel computing; this is what H3D is designed to do.

Fortunately, H3D scales nearly perfectly when run in native MPI across the OCI of an Intel Xeon Phi as seen in Figure 7. The problem can then be further decomposed across Intel Xeon Phi coprocessors to make the largest of simulations a possibility.

C. *GYRO*

The Gyro tokamak plasma simulation code used in this paper has been optimized by the team at General Atomics as well as Dr. Vincent Betro and Dr. Mark Fahey at the National Institute for Computational Sciences (NICS). Gyro was ported through the simple addition of the "-mmic" compiler flag and optimized through the use of hybrid MPI/OpenMP parallelism, and it only required the netcdf and fftw/2.1.5 libraries to be recompiled for the Intel Xeon

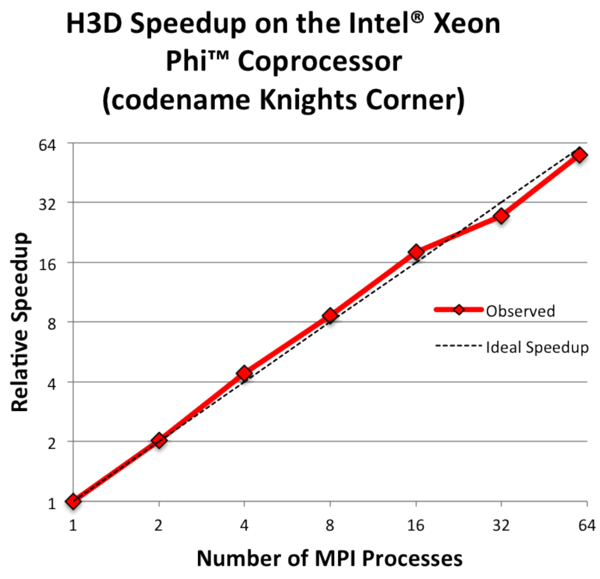


Figure 7: Scaling of the H3D code run in native mode on the Intel Xeon Phi coprocessor

Phi , since MKL was available for the normal calls to libsci or tpsl.

Gyro numerically simulates tokamak plasma microturbulence. It computes the turbulent radial transport of particles and energy in tokamak plasmas and solves 5-D coupled time-dependent nonlinear gyrokinetic Maxwell equations with gyrokinetic ions and electrons. To do so, it utilizes second-order implicit-explicit Runga-Kutta integration with a fourth-order, explicit Eulerian algorithm. It can operate as a flux-tube (local) code, or as a global code, with electrostatic or electromagnetic fluctuations [4].

The MPI only scaling of Gyro seen in Figure 8 shows that the speed of the Intel Xeon still far outweighs the number of cores on the Intel Xeon Phi in so far as time to solution; this is because the clock speeds of the cores of the Intel Xeon (2.6 GHz) currently cannot be matched by the Intel Xeon Phi (1.053 GHz). However, due to memory bandwidth limitations that occur when packing 16 MPI ranks onto the full 16 cores of Intel Xeon available, the Intel Xeon scaling does not continue on the same trajectory as the Intel Xeon Phi scaling does. So, the addition of more threads and better vectorization to utilize those threads could yield walltimes that are shorter than those observed for the same number of MPI ranks on the Intel Xeon , just with less threads being used on the Intel Xeon due to chip size limitations. This will be the subject of future work on optimizing the Gyro code, and one can see preliminary hybrid MPI/OpenMP results for scaling on Intel Xeon Phi in Figure 9.

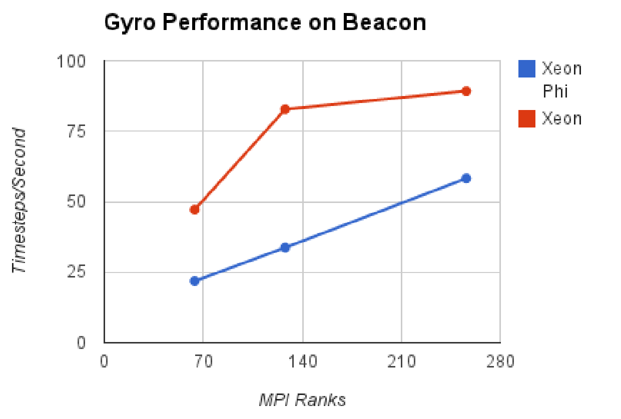


Figure 8: Number of time steps per second versus number of MPI ranks for Gyro runs on the Intel Xeon Phi and Intel Xeon on Beacon

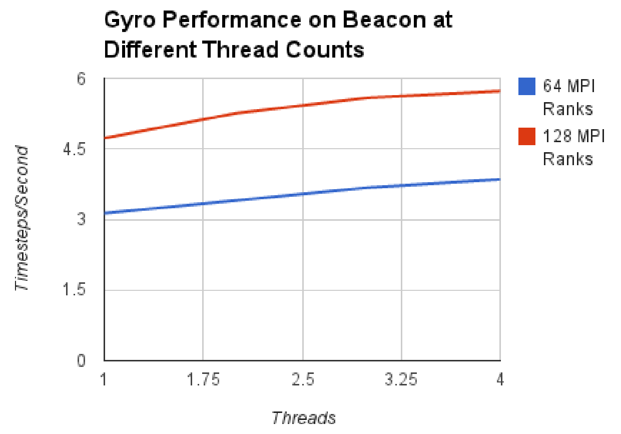


Figure 9: Number of time steps per second versus number of threads for Gyro runs on 64 and 128 MPI ranks on four Intel Xeon Phi coprocessors

D. BGK Boltzmann Solver

The version of the Boltzmann BGK solver was optimized by Ryan Hulguin (NICS) and Rob VanDerWingaart (Intel), and it gets its speed from vector alignment (use of `#pragma SIMD` and `#pragma IVDEP`), precision switching, and low thread overhead as is seen in a previous paper by the authors [9].

The Boltzmann BGK solver is a computational fluid dynamics solver based on the BGK model Boltzmann equation. The BGK model Boltzmann equation is typically used to solve non-continuum rarefied gas flow where the continuum assumptions of the Navier-Stokes equations break down. The BGK model Boltzmann equation could have hundreds of thousands of state variables that need to be solved at each

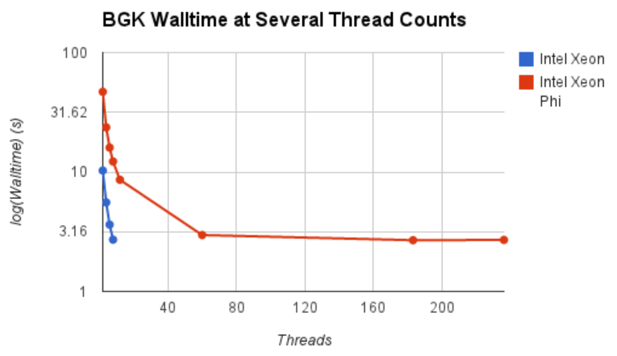


Figure 10: Walltime versus number of threads for BGK Boltzmann solver on one Intel Xeon Phi and one Intel Xeon on Beacon

grid point, making it a great candidate for vectorization and acceleration.

In this case, the solver was run on one Intel Xeon Phi with differing numbers of threads. All runs were run with `KMP_AFFINITY="balanced"`. The physical case (which is a canonical CFD case) has the right plate moving and the left plate stationary and runs until the Argon gas in between the plates reaches a steady state, which was considered to be when the residual reached 10^{-6}, which is approximately 30 pseudotimesteps.

As one can see, the best results were achieved when approximately three threads per core were used, likely due to the "balanced" logical core mapping giving enough wiggle room for OS activity and memory latency to be obscured. Additionally, the wide vector unit, fused multiply add functionality, and single precision intrinsics were employed to their fullest extent to get full performance out of the Intel Xeon Phi . By doing these types of optimizations, the Intel Xeon Phi was able to finish the computation faster than the Intel Xeon by approximately 0.1 second at high thread counts, as is seen in Figure 10, which shows the potential for speed up if the coprocessor is used to its fullest extent.

IV. Codes Ported Simply Through Compilation

It is a worthy discussion to briefly mention the following codes which were ported to the Intel Xeon Phi simply by adding "-mmic" to the compile line and recompiling third party libraries using the "-mmic" compiler flag. This shows the simplicity of porting, and it is inferred that the speedups that are mentioned were garnered from simply recompiling without optimization. A summary of these speedups can be seen in Table I.

Table I: Scaling for Unoptimized Codes on one Intel Xeon Phi on Beacon

Code	MPI Ranks	Walltime	Speed Up	% Peak
HOMME-CAM	32	82.65 s	—	—
HOMME-CAM	64	54.91 s	1.51	76%
PSC	16	894.41 s	—	—
PSC	32	679.58 s	1.32	66%
AWP-ODC	32	932 s	—	—
AWP-ODC	64	524 s	1.78	89%
AWP-ODC	128	287 s	3.25	81%
TRANSIMS	1	5904 s	serial	—
ASCAPE	1	30.197 s	serial	—

A. *HOMME-CAM*

The High Order Method Modeling Environment (HOMME) is a scalable, global hydrostatic atmospheric modeling framework [10], [11]. HOMME-CAM is integrated into the Community Atmospheric Model (CAM), the atmospheric component of the Community Climate System Model (CCSM). HOMME-CAM relies on a cubed-sphere grid, where the planet Earth is tiled with quasi-uniform quadrilateral elements, free from polar singularities. HOMME-CAM is the first ever dynamic code to allow for full two-dimensional domain decomposition in CAM. This was run on Beacon as 32 and 64 MPI ranks on one Intel Xeon Phi , and it achieved 76% efficiency using the OCI on the Intel Xeon Phi .

B. *PSC*

PSC is a particle-in-cell (PIC) code developed by Dr. Kai Germascshewski at the University of New Hampshire that simulates plasma kinetics by solving the collisionless Vlasov-Maxwell PDE (or, with its collision operator, the weakly collisional Fokker-Planck-Maxwell system which describes weakly-coupled, classical plasmas) by tracking the motion of a group of macroparticles (each of which simulates a large number of physical particles). Then, the electric currents due to the macroparticles are summed and the electric and magnetic fields are evolved. Particle positions are updated using the relativistic equations of motion, with particle acceleration specified by the Lorentz force. A Boris-type method in the momentum update ensures that the magnetic field does no work on the particle (to machine precision). The current density is calculated from the charge densities before and after the appropriate position update for each particle in a way that exactly satisfies the discrete continuity equation for charge and current. Fields are then advanced using Maxwells equations and are represented on a staggered Yee grid to maintain the divergence-free constraint of the magnetic field. PSC implements a collision operator, following a classical Monte-Carlo method, wherein the operator simulates binary collisions, and random pairs of particles are selected to represent the full ensemble of collisions.

PSC is currently written mainly in C99, with some optional modules still available in the original Fortran 90, and it relies only on parallel HDF5 and MPI external libraries [8]. When run on Beacon at 16 MPI ranks and 32 MPI ranks on one Intel Xeon Phi coprocessor, it saw a speedup of 1.32, yielding 66% efficiency.

C. AWP-ODC

The Anelastic Wave Propagation (AWP-ODC) code simulates dynamic rupture and wave propagation during an earthquake. Dynamic rupture creates friction, traction, slip, and slip rate information on the fault, and the moment function is created from fault data and used to initialize wave propagation. A finite difference, staggered-grid scheme is used to approximate the 3D velocity-stress elastodynamic equations. Dynamic rupture may be modeled with the Stress Glut (SG) or the Staggered Grid Split Node (SGSN) method, and there are two available external boundary conditions that minimize artificial reflections back into the computational domain: the absorbing boundary conditions (ABC) of Cerjan and the Perfectly Matched Layers (PML) of Berenger. AWP-ODC is written in Fortran 77 and Fortran 90 and utilizes MPI and MPI-IO. [12]

On Beacon, AWP-ODC was used to solve a 3D problem of size $512\times512\times512$ using processor topologies of $4\times4\times2$, $4\times4\times4$, $4\times4\times8$ for 32, 64, and 128 MPI ranks, respectively. Of the run time, an average of 2% was spent in point-to-point communication and an average of 13% was spent in collective communication.

D. TRANSIMS

The Transportation Analysis and Simulation System (TRANSIMS) is an agent-based cellular automata model that creates activity-based travel demand models by individually monitoring several drivers' decisions over the course of a simulation and tracking the routes each driver traverses over the network. The physical act of driving is simulated by having driver agents progress through a series of cells, wherein if a vehicle is already in the approaching cell, traffic jams and congestion can result and be modeled as two cars cannot occupy the same space. [13]

The code is written in serial C++, though a parallel version in MPI is expected soon, and the specific run was based on a traffic model for Alexandria, VA. The main purpose of discussing this serial code is that it is still portable to the Intel Xeon Phi .

E. ASCAPE

Ascape is a Java code for developing and exploring all-purpose agent-based models which offers a broad array of modeling and visualization tools, which were inactive for the runs on Beacon to save wall time that would be used for X forwarding [14]. The model which was run on Beacon was a canonical social science model called"The Prisoner's Dilemma," wherein cooperation is rewarded but not always chosen despite this fact. The results show how the decisions of individual agents affect those of others based on proximity and several other factors.

Ascape was run for 200,000 iterations with 100 agents, a mutation rate 0.2, and the display set to off. The main reason for mentioning Ascape is that it is a non-traditional HPC tool that runs in an interpreted language. This is still compatible with the Intel Xeon Phi , so long as one has a version of Java compiled for the Intel Xeon Phi .

V. Conclusions and Future Work

It is clear that accelerator based systems are the wave of the future based both on their power consumption and variety of programming paradigms to fit the needs of all applications developers. In order to get the most out of this hardware, programmers must be aware of where there code is spending the most time so they may optimize this area; this can be determined through profiling tools and scaling studies, such as this one. In this study, areas of optimization that bore the most performance gain were highlighted, and a set of metrics for comparison and lessons learned by the team at AACE were presented, with the intention that it can give new developers a head start in porting as well as a baseline for comparison of their own code's exploitation of fine and medium-grained parallelism.

The most beneficial optimizations noted in this work were the use of hybrid MPI/OpenMP coding, which allows the fast context switching of the Intel Xeon Phi , along with its massive number of threads (240), to be utilized most effectively. Also, vector memory alignment and the use of `#pragma SIMD` and `#pragma IVDEP` allowed the Boltzmann BGK solver to beat the performance of Intel Xeon with two and a half times faster cores on an Intel Xeon Phi . Finally, the use of thread affinity to assure that memory access is as even as possible between threads and that all cores of the Intel Xeon Phi are being utilized was found to be important in achieving maximum performance.

Acknowledgment

The authors would like to thank Mikhail Smelyanskiy, Karthikeyan Vaidyanathan, Ian Steiner, Joe Curley, Jim Jeffers, Pradeep Dubey, Susan Meredith, Rajesh Agny, and Russ Fromkin at Intel Labs, Alexander Heinecke from Technische Universitat Munchen, and John Lee, David Parks, Edgardo Evangelista, and Danny Tran at Cray, Inc. (formerly Appro) for their invaluable assistance in making Beacon's debut at the top of the November 2012 Green 500 a success. Additionally, the willingness of Intel's software and hardware engineers to work in concert with and listen to the needs of applications engineers at NICS and elsewhere has been invaluable. Finally, the authors wish to thank the National Science Foundation for their support of the Beacon project, along with the State of

Tennessee, and invite you to participate in an open call for research on Beacon, which you may apply for at: http://www.jics.tennessee.edu/aace/beacon/open-call.

References

[1] CompuGreen, LLC. "The Green 500 List - November 2012". $http://www.green500.org/news/green500-list-november-2012$. Accessed December 19, 2012.

[2] Feldman, Michael. "Heterogeneous Computing and HPC Accelerators, Disruptive Technologies in the Making". *HPC Wire*. June 18, 2011. $http://www.hpcwire.com/hpcwire/2011-06-18/heterogeneous\_computing\_and\_hpc\_accelerators, \_disruptive\_technologies\_in\_the\_making.html$. Accessed April 26, 2012.

[3] G. Amdahl,Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities,AFIPS Conference Proceedings,30,pp. 483-485, 1967.

[4] Betro, V and Fahey, M. Performance of the fusion code GYRO on three generations of Cray computers. Poster at International Super Computing 2013. Liepzig, Germany. June 17-20, 2013.

[5] Chrysos, G. Senior Principal Engineer, Intel. "Hot Chips" Presentation. August 28, 2012. Accessed April 8, 2013. $http://www.theoverclocker.com/wp-content/uploads/2012/08/Intel\_Xeon\_Phi\_Hotchips\_architecture\_presentation.pdf$.

[6] The Enzo Project Team. "The Enzo Project." Accessed April 9, 2013. $http://enzo-project.org/$.

[7] Jones, G.S. "When worlds collide: Researchers harness ORNL supercomputers to understand solar storm/magnetosphere". February 6, 2012. Accessed April 8, 2013. $http://www.ornl.gov/info/features/get\_feature.cfm?FeatureNumber=f20120206-00$.

[8] Germascshewski, Kai. "Benchmark Analysis of Plasma Simulations on the Intel Xeon Phi". Presentation at SIAM Southeastern Atlantic Section Meeting. Knoxville, TN. March 23, 2012.

[9] Betro, V., Brook, G., and Hulguin, R. "Hybrid Message Passing and Threading for Heterogeneous Use on CPUs and the Intel Many Integrated Core (MIC) Architecture". Proceedings of the XSEDE Extreme Scaling Workshop 2012: Chicago, IL. July 15-16, 2012.

[10] Edwards, J., Thomas, S., Nair, R. "A mass and energy conserving spectral element atmospheric dynamical core on the cubed-sphere grid". Journal of Physics: Conference Series. Volume 78, Issue 1. June 2007.

[11] Lauritzen, P., Nair, R., Ulrich, P. "A conservative semi-Lagrangian multi-tracer transport scheme (CSLAM) on the cubed-sphere grid". Journal of Computational Physics. Volume 229, Issue 5. pp. 1401-1424. March 1, 2010.

[12] High Performance Geocomputing Laboratory. "Anelastic Wave Propagation (AWP-ODC)". Accessed April 8, 2013. $http://hpgeoc.sdsc.edu/AWPODC/$.

[13] Smith, S. "Transimis: An Open Source Transportation Modeling and Simulation Toolbox". June 7, 2010. Accessed April 8, 2013. $http://code.google.com/p/transims/wiki/GettingStarted$.

[14] Parker, M. "Ascape Guide". Accessed April 8, 2013. $http://ascape.sourceforge.net/$.

[15] Zverina, Jan. "SDSC Mourns the Loss of Dr. Robert P. Harkness". UC San Diego News Center. February 1, 2013. Accessed April 9, 2013. $http://ucsdnews.ucsd.edu/pressrelease/sdsc\_mourns\_the\_loss\_of\_dr.\_robert\_p.\_harkness$