Next Generation Science Applications for the Next Generation of Supercomputing

C.T. Vaughan, S.D. Hammond, D.C. Dinge, P.T. Lin, D.M. Pase,
C.R. Trott, J. Cook, C. Hughes and R.J. Hoesktra

Sandia National Laboratories, Albuquerque, New Mexico, USA

Abstract

The Trinity supercomputer deployment by Los Alamos and Sandia National Laboratories represents the first Advanced Technology System (ATS) deployment for the United States National Nuclear Security Administration (NNSA). The platform will be one of the largest XC40 deployments in the world when final integration of its nearly ten-thousand nodes of dual-socket Intel Xeon Haswell and ten-thousand nodes of Intel Knights Landing processors is completed during 2017. In the deployment of the Knights Landing partition, we have recognized a number of novel hardware technologies that are in many ways sentinels for the next generation of supercomputers. These include wide vector processing units (VPUs), greater use of symmetric multi-threading (SMT) as well as multiple pools of memory, each with differing and complex performance characteristics. Trinity, therefore, represents the start of significant change in the production computing landscape for the NNSA. Recognizing that these new hardware features may pose unique challenges for legacy codes has created an effort associated analyzing the performance of our application portfolio on the novel partitions of Trinity. We summarized our early findings in this context at CUG 2016 covering early evaluations of Sandia's SIERRA engineering suite on Haswell and Knights Landing systems. In this paper however, we turn our attention to the performance of a suite of new applications that have been written from the ground up to be portable across computing architectures, parallel in terms of multi-node and on-node threading and to feature more flexible component- based code design. These applications strongly leverage technologies such as Kokkos, Sandias C++ Performance Portability programming mode, the Trilinos linear solver library and our broader performance analysis capabilities in a close knit codesign program. Driven by the NNSAs Advanced Technology Development and Mitigation (ATDM) program, the new codes represent prototypes of fully-capable production science codes that will execute with high levels of efficiency on the next-generation of supercomputing platforms including Trinity and beyond. In our talk, we will specifically study the new SPARC computational fluid dynamics code developed under the ATDM program, reporting on its performance on both the Haswell and Knights Landing partitions of the Trinity supercomputer. We show a range of performance observations but importantly, that through very careful design, we can provide strong MPI and threading performance on these machines countering some of the concern that the use of OpenMP threading will always lead to poorer performance than pure MPI instantiations. Such conclusions are important within the NNSA because they speak to the ability to maintain scientific delivery on future machines while more novel threading algorithms are added to our production code suites. While considerably more work is required in this area, and this continues to be done under the ATDM program, early studies such as this are helping to characterize what performance levels we might expect to find on systems along the path to Exascale. By engaging with the broader Cray User Group community, we expect to be able to share our observations and invite feedback for the approaches we are pursuing so that the HPC community at large can benefit from the next generation of supercomputing hardware designs.

1 Introduction

The United States Department of Energy (DOE) is embarking on a fundamentally new course for the high-performance supercomputing-based science which sits at the heart of its mission. Since the global ban on nuclear testing during the 1990s, the DOE's National Nuclear Security Administration (NNSA) has invested heavily in the development of computer-based simulation to support its qualification of the United States nuclear stockpile. As the nuclear devices on which the nation's deterrent are built continue to grow older, the computational demands associated with ensuring their performance and safety have continued to rise. To this end, the NNSA is looking to a new generation of "Advanced Technology Systems" – supercomputers with novel or advanced computing hardware – to provide efficient scaling and high computational performance.

Trinity, a Cray XC40, is the first system of this class to be procured and installed within the NNSA complex. It marks a new chapter in the development and evolution of complex science and engineering-based scientific applications. The traditional MPI-only based execution paradigms of the past are pushed to their limits on this platform which, at full scale will exceed half a million processor cores. Instead, advanced application development activities that will extend these codes to include extensive levels of vectorization, threading and multiple memory pool management are underway.

In this paper we report on our early experiences with the Intel Knights Landing partition of the Trinity supercomputer with respect to a next-generation of applications being developed at Sandia National Laboratories. We benchmark these applications on the traditional production domain of Xeon processors and the next-generation of computing hardware in the form of Intel's Knights Landing, self-hosted, many-core processor. Our results show strong levels of performance and good scaling can be demonstrated both across multiple-nodes and with on-node threading. While much work remains to be done in this area, performance studies such as those described in this paper, provide a steer for our development activities and our planning for workloads/application development that will be needed in the continued push towards Exascale-class computing in the next decade.

The contributions of this paper are to provide some of the earliest analysis of production-class engineering applications operating at scale on Cray XC40 Knights Landing-based supercomputers. The complexity of the applications described in this work is high and porting a non-trivial exercise. We provide insight into the scaling characteristics of the platform and the performance associated with the use of MPI/OpenMP hybrid applications. In so doing, we document our first attempts at utilizing the novel features of this production-grade Cray supercomputer, describing our experiences for the broader HPC community and our own internal development teams. Trinity represents a unique opportunity to utilize the dual-socket Xeon processor partition for legacy production workloads, while simulatenously, optimizing for the a future generation of systems using the Knights Landing nodes. The platform is an ideal step in the migration of our complex stockpile workloads from our historical machine architectures to the computing of our future.

2 NNSA/ASC Trinity ATS Platform

The *Trinity* supercomputer project was launched by Los Alamos and Sandia National Laboratories in 2011 and was the first Advanced Technology System (ATS) [1, 3, 7] project within the Department of Energy's National Nuclear Security Administration (NNSA). ATS platforms represent a new strategy for the NNSA which seek to blend production high-performance computing cycles with novel additions to the computing hardware, allowing production code groups to gradually port their applications over to next-generation processors and accelerators.

The first partition ("Phase-I") of Trinity was installed during 2015 and featured roughly 9,500 nodes of dual-socket Haswell E5 Xeon processors interconnected with Cray's high-performance Aries DragonFly interconnect. In November 2016, the machine entered the Top500 HPL list in 10^{th} position, providing the NNSA programs with more than 8PF/s of computing power.

Phase-II of the Triniy deployment, which at the time of writing is currently in the middle stages of Early Science testing, will comprise almost 10,000 Intel Knights Landing Xeon Phi ("KNL") 7250 processors. Each node is a single socket self-booting KNL that provides 34, dual-core compute tiles that share a 1MB L2 cache. The tiles are arranged on a two-dimensional mesh network-on-chip that reduces latency of previous Intel designs while providing the extreme inter-core and core-to-memory bandwidths required for high-performance scientific workloads. A novel two-level memory system is offered on each processor, with a 16GB MCDRAM (high-bandwidth memory) that can either be used as an addressable region or a large, direct-mapped cache for the 96GB of DDR4 capacity system memory. Knights Landing pushes hard into the ‘advanced‘ definition of our ATS systems for the broad range of production computing codes that the three NNSA nuclear stockpile laboratories hope to port and run on the final Trinity system whrn both phases are merged during the summer of 2016. KNL provides dual wide-vector processing units, complex memory and caching options and quad-hyperthreading on each core – all features that are first of a generation technologies along the path to Exascale computing early in the next decade.

3 Porting Codes to the NNSA/ASC Advanced Technology Systems

Recognizing that the advanced and novel features of a machine like Trinity pose both challenges for the portfolio of legacy stockpile stweardship codebase, but also, significant opportunities to enhance the scientific delivery for the stockpile program, the NNSA/ASC program launched the Advanced Technology Development and Mitigation program (ATDM) during 2013.

ATDM is a project started to identify future computing technologies that will be required to support advanced stockpile workloads in the coming decades and then to develop a suite of scientific and engineering capabilities that are designed, from the ground up, to take advantage of the performance that such technologies can offer. To this end, the program encompasses a broad range of foundational technologies such as advanced linear solvers, mesh definition and handling frameworks, as well as state-of-the-art software engineering practices and skill development. Each NNSA/ASC laboratory is using ATDM to develop next-generation applications written to take advantage of mutli-threaded execution, vectorization, GPUs (where appropriate) and multiple memory spaces, with the understanding, that the skills and approaches that are pioneered in the new generation of codes can be used as path-finding activites for the broader legacy code base. Trinity is therefore well matched as an ATS platform to provide computing cycles to the ATDM projects and applications.

3.1 Sandia’s ATDM-SPARC Application

Under the ATDM project, Sandia is developing several next-generation engineering capabilities. One such application is Sandia’s Parallel Aerosciences Research Code (SPARC) which provides advanced simulation of computational fluid dynamics (CFD) calculations. The code is mutli-node parallel, three-dimensional, using traditional MPI but features on-node parallelism provided by OpenMP directives and, for some of the significant computational kernels, implementations using Sandia’s Kokkos C++ parallel pattern abstractions [2]. SPARC utilizes cell-centered finite volume methods for CFD calculations and Galerkin finite-element methods for ablation and thermal analysis [5]. Several packages from the Trilinos framework [4] are also used during execution to provide solutions to complex systems of equations.

4 Benchmarking SPARC on the Trinity Platform

In Figures 1-3 we present benchmarked performance on the most recent update to the Trinity Phase-II KNL partition (currently undergoing Early Science shake down). For these benchmark runs the Intel 2017 Update 2 compiler was used via Cray’s platform optimization wrappers which allow the compiler to target the KNL AVX512 vector instruction set. Each run is timed for 200 timesteps executing either Line or Point integration schemes (both results are shown) using the KNL Quad-Flat memory mode (MCDRAM available as an allocatable space). Our initial experimentation has shown little performance change when forcing SPARC to use either the capacity DDR4 memory or the MCDRAM via standard Linux NUMA tools. Unless otherwise noted, the runs utilize 64 MPI ranks per node with affinitization of each rank set to the core on which it is allocated.

4.1 Multinode Scaling of the ATDM-SPARC Application

Figures 1 and 2 present weak and strong scaling of SPARC solving a reactive gas problem of interest when utilizing its native and Trilinos solvers respectively. We note that the weak scaling results show a signifcant increase in execution time for point integration schemes beyond 4,096 ranks – the first time the job spans

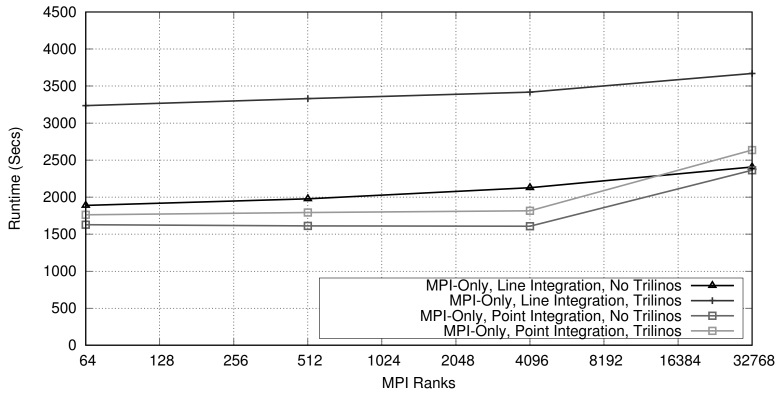

Figure 1: SPARC Weak Scaling Performance on Trinity KNL for MPI-Only Execution

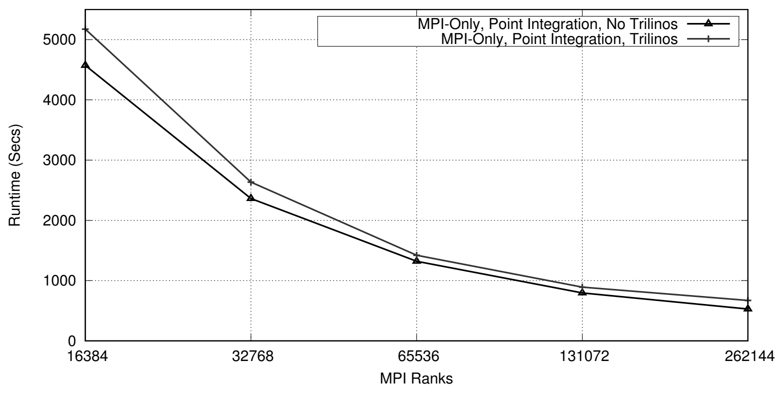

Figure 2: SPARC Strong Scaling Performance on Trinity KNL for MPI-Only Execution

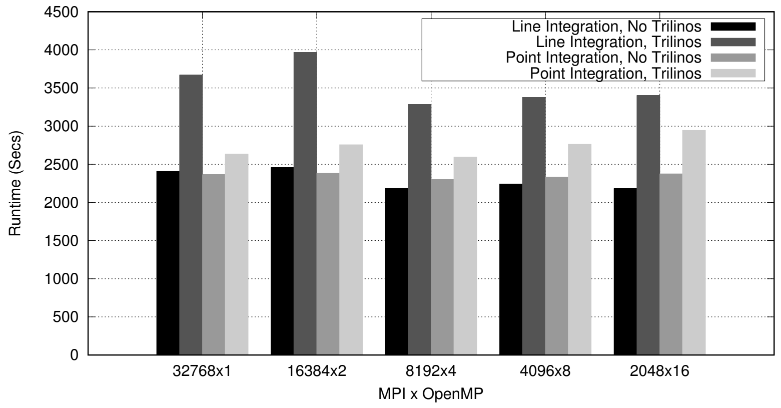

Figure 3: SPARC Performance on Trinity KNL for MPI/OpenMP Hybrid Execution

multiple Aries DragonFly groups. Weak scaling for the line integration scheme is smoother but it is clear that the implementation of the solvers within Trilinos performs very poorly versus the native SPARC solver with performance more than 50% slower. Improving the performance of equivalent solvers in Trilinos remains an on-going task within the ATDM project and these studies will serve as input to optimization objectives in subsequent development years.

Strong scaling of the SPARC benchmark problem is shown in Figure 2. The application achieves very good scaling behavior right the way out to 4,096 nodes (262,144 MPI ranks) of the Trinity partition which is the current maximum permitted due to static splitting of the machine into two halves, one providing quad-cache memory mode KNLs and the other providing quad-flat which is where these jobs have been run. As with the point integration results in Figure 1, we see close performance correlation between the native solver and the packages provided by Trilinos with a gap of between 7 and 27%.

4.2 Hybrid MPI and Thread Scaling of the ATDM-SPARC Application

In Figure 3 shows benchmarked hybrid MPI/OpenMP execution on the KNL systems of Trinity. For these runs Intel's OpenMP runtime is used with a single hyperthread enabled per core and affinity now set to the depth (size) of the OpenMP domain. Our benchmarking has shown that the use of a single hyperthread per core yields the best results for SPARC. We present results using a total of 32,768 cores arranged in different combinations of MPI ranks and OpenMP threads. The native solvers of SPARC show consistent performance across all of the combinations run and demonstrate their best runtimes when using either 4 or 8 threads. When using the Trilinos solvers however, the picture is a little more mixed with some improvement at 4 and 8 threads for line integration schemes but consistently increasing runtime when using point integration. These results give us increased confidence in our ability to find high performance threaded solutions for complex solver schemes. The native results from SPARC indicate very strong performance where threading can provide performance advantages over MPI-only execution (a configuration which performs very well on KNL). As we continue to develop and extend Trilinos in the coming years, the approaches pioneered in the native schemes of SPARC will be gradually added to Trilinos and then made available to the broader HPC community as a set of high-performance, open source packages within the framework.

4.3 Performance Analysis - Threaded Solve Times for ATDM-SPARC

Tables 1 and 2 show the breakdown of a complete application run of SPARC running an ideal gas problem using its native and Trilinos solvers respectively. We show benchmarked times for 64 nodes of KNL and Haswell as a baseline comparison of a per-node evaluation. In each of the cases we have used the same MPI rank and OpenMP placement options and varied only the internal solvers used. Note that the times shown here for a complete run of a SPARC input and accordingly differ from those from the larger scaling runs shown earlier.

Table 1: Timing Breakdown of SPARC Compute and Communication Regions for MPI/OpenMP Hybrid Threaded Execution (Native SPARC Solver, Full Problem)

CPU	MPI/OMP	Compute	Send	Reductions	Wait	Wait Any	Wait All
KNL	4096/1	2893.0	246.6	108.9	32.7	50.5	25.6
KNL	2048/2	2921.4	200.2	97.7	26.2	37.5	18.4
KNL	1024/4	2915.1	97.0	87.5	18.8	28.3	13.4
KNL	512/8	2960.9	77.9	93.9	8.3	27.0	11.2
KNL	256/16	3047.6	95.7	128.5	16.0	24.1	4.2
KNL	128/32	3139.8	90.6	174.2	16.6	30.0	3.2
KNL	64/64	3410.9	115.6	242.2	25.6	75.6	0.4
HSW	2048/1	1956.1	147.5	64.5	10.4	36.7	7.5
HSW	1024/2	2041.1	92.3	67.2	11.4	31.3	6.2
HSW	512/4	2053.9	72.1	53.9	8.4	26.4	4.6
HSW	256/8	2091.8	53.5	66.0	7.0	25.0	3.2
HSW	128/16	2144.5	46.8	79.7	9.9	19.5	1.4

Table 2: Timing Breakdown of SPARC Compute and Communication Regions for MPI/OpenMP Hybrid Threaded Execution (Trilinos Solvers, Full Problem)

CPU	MPI/OMP	Compute	Send	Reductions	Wait	Wait Any	Wait All
KNL	4096/1	3933.1	222.2	219.5	0.008	55.7	105.5
KNL	2048/2	4030.8	158.2	172.4	0.003	42.1	80.1
KNL	1024/4	4178.7	106.4	117.2	0.005	30.4	46.6
KNL	512/8	4517.6	91.8	92.0	0.004	26.8	33.8
KNL	256/16	5293.0	96.9	98.0	0.004	24.8	30.4
KNL	128/32	6533.6	76.6	103.9	0.01	29.9	23.5
KNL	64/64	9140.6	111.4	178.1	1.8	63.6	45.8
HSW	2048/1	2210.0	142.1	128.9	0.0002	34.4	47.1
HSW	1024/2	2353.5	94.9	101.6	0.006	28.2	28.1
HSW	512/4	2451.2	74.4	82.8	0.001	24.8	21.3
HSW	256/8	2700.8	58.2	74.6	0.003	24.8	15.2
HSW	128/16	3123.4	44.6	56.5	0.003	19.6	11.6

Table 3: Performance of SPARC on Trinity Haswell (XC40) vs. Commodity Sky Bridge Cray CCS System

MPI	Sky Bridge (CCS)				Trinity-Haswell (XC40/50)			
Ranks	Nodes	Runtime (S)	MPI Time (S)	MPI Time (%)	Nodes	Runtime (S)	MPI Time (S)	MPI Time (%)
32	2	8917	108	1.21	1	12949	104	0.80
64	4	4521	105	2.31	2	6446	56	0.87
128	8	2364	97	4.10	4	3288	54	1.63
256	16	1250	72	5.77	8	1670	46	2.77
512	32	729	115	15.78	16	927	55	5.96
1024	64	386	88	22.82	32	486	93	19.24
2048	128	227	71	31.39	64	270	73	27.17

The tables show that the communication strategies employed by the two runs are considerably different – Trilinos employs Waitall-based communication routines over the SPARC-native calls which are based on individual calls to Wait (note that on aggregate the amount of time spent in these routines is quite similar between the runs but is distributed between the operations differently). Send communication times are similar with variation of approximately 10-15% between the two sets. Where there is a significant difference between the two solve schemes is in the compute times recorded. Even without threaded execution (the 4,096 MPI ranks with a single OpenMP thread for KNL runs and 2048 MPI ranks with a single OpenMP thread for Haswell), execution time when using Trilinos can be from 10-35% higher, growing to nearly 3X the compute time when 64 MPI ranks are used with 64 threads per rank for KNL. The native solvers increase slightly (10-15%) but remain fairly consistent when using OpenMP threads runs indicating that it is possible to utilize high levels of threading and maintain performance on this specific problem configuration. These results also reflect our andecdotal evidence from our Trinity code porting activities – that for complete runs of applications (not just selected kernels), KNL is roughly competitive but slightly slower than Haswell once representative, complex input decks are benchmarked on the system. This reflects the multi-decade optimization for codes on multi-core processors and the limited optimization which has been applied for wide-vector many-core systems – something that the ATDM project at Sandia is seeking to address during the coming years.

4.4 Performance Analysis - Commodity Servers and Capability Machines for ATDM-SPARC

The NNSA purchases a broad range of computing machinery for its mission. Machines like Trinity, which represent the Advanced Technology Systems, represent our capability supercomputing machinery which is optimized to achieve large scale and consistent high performance. The smaller machine installations are used for capacity computing – often to run many smaller jobs. In Table 3 we benchmark the recently installed Sky Bridge, Cray CCS (commodity) cluster against Haswell compute nodes from Trinity. Sky Bridge is one of the last TriLab Commodity Cluster (TLCC-2) installations of the program (see [6] for additional machine specifications), which recently moved onto a new suite of machines to be installed under the new NNSA "CTS" project. The machine provides dual-socket, eight-core Sandy Bridge Xeon E5 processors interconnected with QLogic QDR InfiniBand. The Sandy Bridge processors feature half of the cores available on their Haswell equivalents but provide a 300MHz higher clock rate. Memory on the Sandy Bridge systems is 1600MHz DDR3 versus 2133MHz DDR4 on the Trinity compute blades. The results from Table 3 show that the Trinity compute nodes provide approximately 40% performance improvement over the those used in Sky Bridge which closely resembles the cycles available from a doubling of the processor cores but lowering of the processor clock. The MPI time is approximately consistent between the two systems from a node-equivalent basis which reflects the ability of the Trinity (XC) systems to double the number of MPI ranks per node but maintain network performance characteristics. It remains an important aspect of our capability machines to demonstrate strong performance when compared to capacity equivalents, the results shown earlier in this paper show good levels of scaling as we utilize the Aries interconnect to run progres-

Table 4: Performance Metrics from Counter Analysis on Trinity Haswell Nodes (Column indicates the number of MPI ranks used)

Metric	32	64	128	256	512	1024	2048
L1 Load Hit Rate	92.177	93.660	92.824	93.426	94.554	95.046	96.062
L 1 Load/Store Ratio	3.112	3.103	3.049	3.020	3.037	2.967	3.005
LLC Load/Store Ratio	41.128	40.620	40.784	39.332	24.779	25.491	19.360
Cycles/L1 Access	2.809	2.251	2.583	2.837	2.201	2.539	1.689
Cycles/LLC Access	90.989	91.012	94.015	115.596	107.626	141.914	124.917
Instructions Per Cycle	0.791	0.800	0.845	1.014	0.992	1.136	1.271
Pages Faults per Access	0.000747	0.000742	0.000725	0.000001	0.000001	0.000001	0.000002

sively larger and larger jobs across the machine, the results shown here demonstrate excellent performance when compared to commodity equivalents that are usually cheaper to procure and install but often do not scale to the large installation required for our production capability workloads.

4.5 Haswell Performance Analysis

Table 4.5 shows performance metrics observed when strong scaling MPI only runs of SPARC using native solves on the Trinity Haswell partition. Note the results show the variation in metrics when the number of MPI ranks is increased. The metrics shown help to correlate with our anecdotal evidence that the code performs well on the hardware when compared with similar metrics obtained for our broader portfolio. In particular, the issuing of more than an instruction per cycle for any production code is quite rare even on well optimized hardware such as the Haswell E5 Xeon processor. Cache performance is strong with a heavy read dominance of 3 reads per write in the L1, and between 20 and 40 reads per write at the Last-Level Cache (LLC). The high number of cycles per LLC access (more than 100 for some cases) reflects the design of the code to utilize the smaller caches closer to the core wherever possible. TLB behavior is very good for the problem size shown with very few faults per access. We have been unable to obtain all the metrics on the Knights Landing partition of Trinity so far because the porting of our tooling infrastructure is only partially complete however, the strong L1 cache performance and the relatively relaxed use of the LLC shown in this profile have been metrics which have indicated the code will likely perform reasonably on KNL – an observation which correlates with our time based benchmark shown earlier.

5 Conclusions and Discussion

The constant pressure on the computational scientists in the NNSA is seeing the program requirements for large-scale, capability-grade production computing exceed demand. In this context there are two paths forward to the program: (1) optimize the application portfolio to execute in fewer resources (either time or compute hardware), or, (2), grow the compute hardware to faster processors and larger scale than ever before. Exascale will push these constraints even further.

In this paper we have described the deployment of Trinity Phase-II, a capability-grade system which will enter service to support stockpile stewardship during 2017. We examined the performance of SPARC, a ground up rewrite of an important class of CFD application, demonstrating excellent scalability and performance for its native solve using MPI-only and hybrid MPI/OpenMPI applications. Compared to previous commodity systems, SPARC is both faster and more scalable on Trinity. The use of the Trilinos solver framework for SPARC remains a work in progress, although performance for MPI-only codes is similar, the use of threads shows performance degredation that needs additional development. SPARC's native solvers are a design vehicle and sentinel for the future direction of Trilinos and the performance reported in this paper is just one step along a path to further extensions that will see multiple packages optimized for Exascale and beyond.

The Knights Landing partition of Trinity contains several first-of-kind hardware features – wide, but incredibly capable, vector units, many-core hyperthreading and complex memory hierarchies. While these pose unique challenges to our legacy applications they also offer us the opportunity for significant increases in scientific delivery for the future. Studies such as this paper outline our first attempts and impressions on this unique computing platform. We look forward to future works which detail our progress in further optimizing our applications and documenting our continued development in preparation for a future which builds on these first steps.

Acknowledgments

Installing, configuring and testing a new machine deployment the size and complexity of Trinity is a difficult task. We are grateful for the support of our colleagues both at Sandia and Los Alamos who perform these tasks with incredible diligence and professionalism, particularly to Ann Gentile who leads the environment and test system bring up for Sandia.

The benchmarking efforts described in this paper have been enabled and signifciantly enhanced by our conversations with Micah Howard.

Trinity is the first advanced technology computing system ('ATS') operated by the National Nuclear Security Administration (NNSA) for the purposes of ensuring the safety and performance of the United States nuclear deterrent. Access to Trinity and the associated test systems is provided by the NNSA's ASC program to Lawrence Livermore, Los Alamos and Sandia National Laboratories.

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

References

[1] J.A. Ang, T.T. Hoang, S.M. Kelly, A. McPherson, and R. Neely. Advanced Simulation and Computing Co-design Strategy. *Publications of the Department of Energy, National Nuclear Security Administration*, February 2016.

[2] H Carter Edwards, Daniel Sunderland, Vicki Porter, Chris Amsler, and Sam Mish. Manycore Performance-Portability: Kokkos Multidimensional Array Library. *Scientific Programming*, 20(2):89–114, 2012.

[3] K.S. Hemmert, M.W. Glass, S.D. Hammond, R.J. Hoekstra, M. Rajan, S. Dawson, M. Vigil, D. Grunau, J. Lujan, D. Mortan, H.A. Nam, P. Peltz, A. Torrez, and C. Wright. Trinity: Architecture and Early Experience. In *Proc. of the Cray User Group (CUG) 2016*, 2016.

[4] Michael A Heroux, Roscoe A Bartlett, Vicki E Howle, Robert J Hoekstra, Jonathan J Hu, Tamara G Kolda, Richard B Lehoucq, Kevin R Long, Roger P Pawlowski, Eric T Phipps, et al. An Overview of the Trilinos Project. *ACM Transactions on Mathematical Software (TOMS)*, 31(3):397–423, 2005.

[5] M.A. Howard. A Multi-Dimensional Finite Element Based Solver for Decomposing and Non-decomposing Thermal Protection Systems. *Proc. of the AIAA Conference on Aviation 2015*, June 2015.

[6] Mahesh Rajan, DW Doerfler, Paul T Lin, Simon D Hammond, Richard F Barrett, and Courtenay T Vaughan. Unprecedented Scalability and Performance of the new NNSA Tri-Lab Linux Capacity Cluster 2. In *High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion*, pages 417–425. IEEE, 2012.

[7] C.T. Vaughan, D.C. Dinge, P.T. Lin, S.D. Hammond, J. Cook, C.R. Trott, A. Agelastos, D. Pase, R.E. Benner, M. Rajan, and R.J. Hoekstra. Early Experiences with Trinity - The First Advanced Technology Platform for the ASC Program. In *Proc. of the Cray User Group (CUG) 2016*, 2016.