Understanding the effects of process placement on application performance on an AMD Interlagos processor

Kalyana Chadalavada, Manisha Gajbe
National Center for Supercomputing Applications
University of Illinois at Urbana-Champaign
Urbana, IL, USA
{kalyana, manisha}@ncsa.illinois.edu

***Abstract*—In this paper, we explore the impact of process placement on application performance when using an AMD Opteron bulldozer architecture CPU on a Cray XE6 node. We conduct a low-level analysis of possible resource contention on the Interlagos core modules using application kernels to exemplify target workloads. We will also characterize the performance of OpenMP threads in dual stream or packed mode and single stream or unpacked configuration. Using CrayPat tools and PAPI counters, we attempt to quantify bottlenecks to efficient utilization of the processors.**

Keywords - XE6, bulldozer, application performance

I. INTRODUCTION

The Blue Waters [1] system is a track-1 National Science Foundation (NSF) super-system being deployed at the National Center for Supercomputing Applications (NCSA), will deliver sustained performance of 1 petaflop on a range of real-world science and engineering applications. This hybrid Cray XE6/XK architecture, in its final configuration will contain over 25000 compute nodes connected in a 3D torus configuration with Cray Gemini network. The Cray XE6 nodes utilize AMD's new 16-core Opteron processor codenamed, the Interlagos processor, providing over 380,000 x86 compute core modules. Each node provides up to 64GB of DDR3 memory. The Cray XK nodes utilize one AMD Opteron processor and one accelerator chip. Each XK node will provide up to 32GB of DDR3 memory. The super-system provides a Lustre file system with over 1TB/s usable throughput. An initial deployment of the Blue Waters super-system called the early science system (ESS), a 1.3+ petaflop system, is complete.

Petascale Computing Resource Allocations (PRAC) [2] from the National Science Foundation allows research teams to work closely with the Blue Waters project team in preparing their codes. Initial set of more than 25 teams selected through the PRAC process cover a wide range of domains including astrophysics, molecular dynamics, weather science, earthquake system science, magnetosphere simulations and computational chemistry. A selected set of the PRAC teams, now called as Science Teams is utilizing the Blue Waters ESS system to achieve breakthroughs in their areas of research.

Initial performance analysis has shown significant variations in application performance based on how the application threads were mapped to the Interlagos processor cores. Understanding the factors driving this behavior will be one of the key aspects in optimizing the wide range of applications chosen for the Blue Waters super-system. For this study, we utilized a smaller single rack XE6 system used as a test and development (TDS) platform.

The remainder of this paper is organized as follows. In section 2, we describe the AMD Interlagos processor (Bulldozer Architecture) and Blue Waters compute node architecture. In Section 3, we describe the methodology adopted for performance analysis. Section 4 describes the kernels and applications used and Section 5 has their results. Finally, Section 6 summarizes this work and comments on remaining issues we plan to address in the future.

II. SYSTEM ARCHITECTURE

In this section, we describe the bulldozer architecture and physical packaging as used on the Blue Waters super-system. Bulldozer is a major redesign of the AMD processor architecture driving more core density and throughput. Contemporary x86 designs share last level of cache, memory controllers and IO interfaces among multiple cores in a single packaging unit. Breaking away from such practices, bulldozer is a hierarchical design to share substantial on-chip resources among multiple cores blurring the traditional notion of an x86 core. As a result, this architecture offers a greater level of flexibility to the users on utilizing the various resources on the processor.

A. AMD 'Bulldozer' Architecture

A bulldozer core module (CM) is the building block of the Interlagos processor. Each core module combines a shared frontend, two integer units, a single shared floating-point unit, level 1 cache, and a shared level 2 cache. Figure 1 provides an overview of the bulldozer core module. A single

bulldozer chip (silicon die) contains four bulldozer core modules. All the core modules on a single bulldozer chip share a level 3 cache, memory controllers, Hypertransport links and other system interfaces. There are four 16-bit Hypertransport links per chip. Each link runs at 6.4 GigaTransfers per second. Figure 2 illustrates the components on a single bulldozer chip.

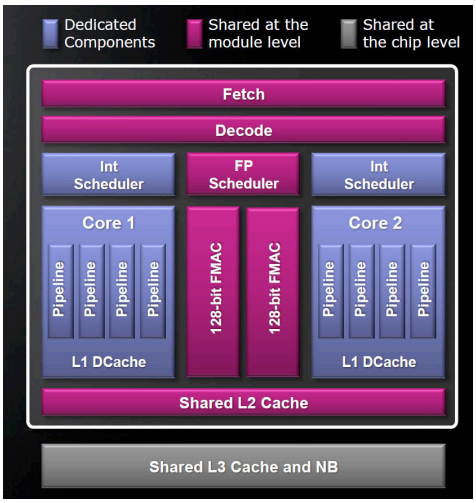

Figure 1. Bulldozer core module

A single AMD Opteron (Interlagos processor) used on the Blue Waters Cray XE6 node is a multi-chip module (MCM) packing two bulldozer chips. The two chips are connected using one full width and one half-width Hypertransport link. A single MCM provides as many as sixteen cores in a single CPU socket. Figure 3 provides an overview of the Interlagos processor.

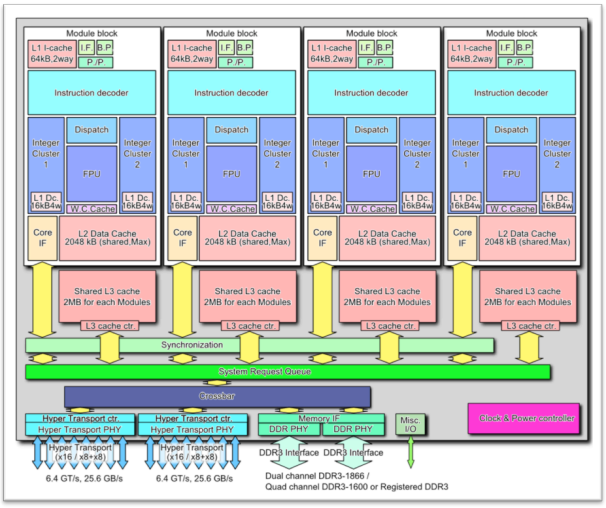

Figure 2. Bulldozer chip architecture

The floating-point unit consists of two 128-bit pipelines and can be fused to function as a single 256-bit pipeline. Each floating-point unit is capable of executing two 128-bit SSE instructions or one 256-bit AVX add or multiply instruction in a single clock cycle. This yields a peak performance of 4 double precision operations per cycle.

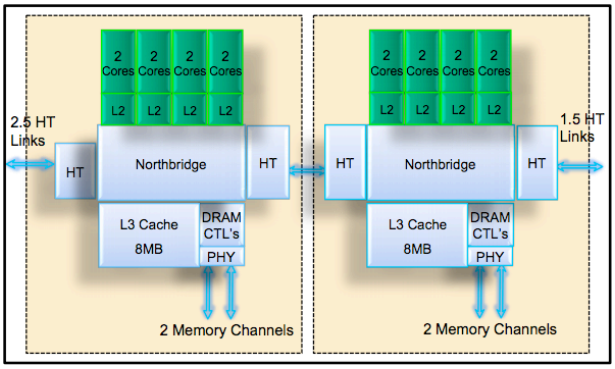

Figure 3. Interlagos processor architecture

Additionally, the floating-point unit also supports a 256-bit fused multiply-add instruction effectively doubling the theoretical performance to 8 double precision operations per cycle and provides improved accuracy.

B. Compute Node Architecture

A Cray XE6 compute node ties together two AMD Opteron processors, their associated memory banks and a Cray Gemini network processor. The Cray Gemini network processor is connected via a full width Hypertransport link to the first bulldozer chip on the first CPU. Each bulldozer chip is connected with every other bulldozer chip on the compute node using the Hypertransport links. However, these connections are not uniform.

- Two chips on an MCM are connected using one full width and one half width link for a total of 24-bit Hypertransport link running at 19.2 GB/s
- Chips 0 and 1 on both MCMs are connected to each other with a half width, 8-bit hyper transport link running at 6.4 GB/s
- Chip 0 on one MCM is connected to chip 1 on the other MCM with a full width, 16-bit hypertransport link running at 12.8 GB/s

A compute node on the Blue Waters system is configured with two 16 core Interlagos processors and 64 GB DDR3 memory. From a memory access standpoint, a compute node is divided in to four non-uniform memory access (NUMA) domains. Each NUMA domain contains four bulldozer core modules and 16 GB DDR3 memory. The operating system, Cray Compute Node Linux (CNL), considers each integer unit a single core. Hence, up to 32 processes or tasks can be placed on a single Blue Waters Cray XE6 compute node. Figure 4 illustrates a single Blue Waters XE6 node.

The impact of NUMA organization on memory bandwidth and application performance has been investigated on several other processors including the previous generations on AMD processors.

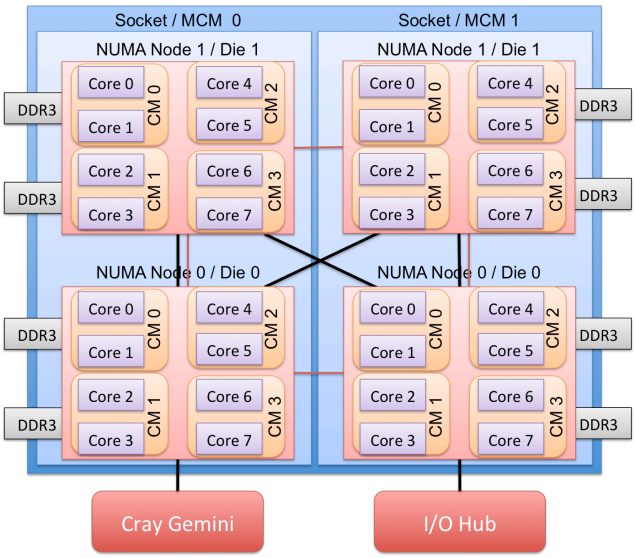

Figure 4. Cray XE6 Compute Node Architecture.
Black line indicates full width Hypertransport link and red indicates half width link

III. TOOLS USED AND DATA COLLECTION

A. *Cray Performance Analysis Tool (CrayPat)*

Cray provides a multitude of performance analysis tools on its systems. CrayPat[6], Cray Performance Analysis Tool, provides a simple interface for program instrumentation, data capture and basic text reporting. It also provides high-level summary or observations of possible performance inhibitors. CrayPat uses PAPI[7], the performance API. This interface is normally transparent to the user.

There are three steps involved in performance analysis of codes using CrayPat.

- *Instrument* your program, to specify what kind of data you want to collect under what conditions.
- *Execute* your instrumented program, to generate and capture the desired data.
- *Analyze* the collected data

CrayPat provides automatic code instrumentation and an API to manually control data collection. CrayPat also provides predefined groups of hardware counters and function groups.

B. *Preparing The Applications For Data Collection*

The Cray Compute Node Linux presents a single bulldozer core module as two distinct cores to the user applications. To understand the performance impacts of simultaneously utilizing both the cores in a single bulldozer core module, we focus on collecting performance counters for the shared components. Performance counters of interest include L2 cache, Translation Look aside Buffer (TLB), floating-point operations etc. We also collect performance counter data for other important events like L1 data cache and instructions per cycle.

All the selected codes were compiled with Cray compilers and libraries. To instrument the target applications, we utilize CrayPat's *pat_build* mechanism. After initial profiling runs, top two code sections with high percentage of run time were wrapped with CrayPat API calls to control performance counter data collection for those specific code segments. Specifically, we used *PAT_record* to toggle data collection.

We use the term dual stream mode or packed mode to refer to the run configuration when two application threads are assigned to a core module, one thread per core. We use the term single stream mode or unpacked mode to refer to the run configuration when only one application thread is assigned to a single core module. We explicitly control thread and process placement using the aprun command line options that enforce process affinity.

IV. KERNELS AND APPLICATIONS

In this paper, we have chosen to use computational kernels from NAS benchmarks, and two scientific application codes, one from the field of cosmology and the other from astrophysics. A brief description of all the selected codes follows :

A. *NAS Benchmarks*

The NAS Parallel Benchmarks (NPB)[3][4] is a small set of programs designed to help evaluate the performance of parallel supercomputers. The benchmarks are derived from computational fluid dynamics (CFD) applications and consist of five kernels and three pseudo-applications in the original "pencil-and-paper" specification. In this paper we have chosen the OpenMP implementation of NAS Benchmarks (NPB3.3-OMP).

- *LU : Lower-Upper Gauss-Seidel solver*

LU is a simulated CFD application that uses symmetric successive over-relaxation (SSOR) method to solve a seven-block-diagonal system resulting from finite-difference discretization of the Navier-Stokes equations in 3-D by splitting it into block Lower and Upper triangular systems.

- *FT : discrete 3D fast Fourier Transform, all-to-all communication*

FT contains the computational kernel of a 3-D fast Fourier Transform (FFT)-based spectral method. FT performs three one-dimensional (1-D) FFT's, one for each dimension.

B. Cosmology Application : Gadget

GADGET[5] is a code for cosmological N-body/SPH simulations on massively parallel computers with distributed memory. It uses an explicit communication model that is implemented with the standardized MPI communication interface. The code can be run on essentially all supercomputer systems presently in use, including clusters of workstations or individual PCs.

GADGET computes gravitational forces with a hierarchical tree algorithm (optionally in combination with a particle-mesh scheme for long-range gravitational forces) and represents fluids by means of smoothed particle hydrodynamics (SPH). The code can be used for studies of isolated systems, or for simulations that include the cosmological expansion of space, both with and without periodic boundary conditions. In all these types of simulations, GADGET follows the evolution of a self-gravitating collision-less N-body system, and allows gas dynamics to be optionally included. Both the force computation and the time stepping of GADGET are fully adaptive, with a dynamic range that is, in principle, unlimited.

This code uses, TreePM method, where the tree is used for short-range gravitational forces only while long-range forces are computed with a FFT-based particle-mesh (PM) scheme. Periodic boundary conditions can be computed, either by means of the Ewald summation technique or based on the FFT algorithm used in the TreePM scheme. Simulations that only follow gas dynamics without self-gravity can be run in periodic boxes with arbitrary aspect ratios, and also in 2D, if desired. In the context of this paper, we consider the performance of dual stream gadget runs relative to the single stream runs.

C. Astrophysics Application : Castro

CASTRO [14], a fully compressible hydrodynamics code to simulate the explosion phase of a Type Ia supernova. CASTRO use structured grids with adaptive mesh refinement (AMR). A time step in CASTRO requires the fully explicit advance of a hyperbolic system of conservation laws, as well as the computation of self-gravity. In addition to simulations of Type Ia supernovae, CASTRO is also being used to study core-collapse and pair-instability supernovae.

CASTRO is implemented using the BoxLib framework developed in the Center for Computational Sciences and Engineering at LBNL. BoxLib is a hybrid C++ / Fortran90 software system that provides support for the development of parallel structured-grid AMR applications. CASTRO uses a hybrid MPI-OpenMP model. CASTRO runs successfully on a wide range of supercomputing systems.

V. RESULTS AND DISCUSSIONS

In this section, we present our experiments and results. We run each application in dual stream and single stream mode for varying process counts. We collect and examine hardware performance counter data and attempt to identify potential resource contention or lack there of. Results from single stream mode are used as a reference for comparison.

In figure 5, we examine performance on LU class C OpenMP benchmark. We present performance data for five key events: Total time, L2 data cache hit ratio (D2 hit Ratio), L1+L2 data cache utilization (D1+D2 Utilization), TLB utilization and L1 data cache misses (D1 misses). The problem size is maintained at class C for all runs.

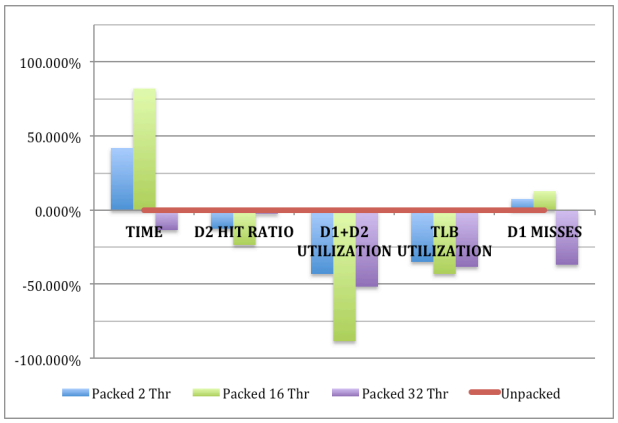

Figure 5. LU Class C benchmark

Performance of dual stream with 2 threads (one core module) and 16 threads (one Interlagos MCM) fares badly compared to single stream with 2 threads (two adjacent core modules) and single stream with 16 threads (two Interlagos MCMs, one thread per core module).

In dual stream mode with 2 and 16 threads, performance metrics of shared components display a marked degradation compared to the single stream mode. We attribute the lower performance in dual stream mode to contention between the two threads for these shared resources on the bulldozer core module.

In the case of dual stream mode with 32 threads, we see an improvement compared to the performance of single stream mode with 16 threads. We attribute this to a marked improvement in D2 hit ratio and D1 misses for the dual stream 32 thread run. Since the problem size is maintained constant between the runs, each thread in the 32-thread run

operates on a smaller data set leading to better cache utilization and improved overall performance.

In figure 6, we present performance data for FT class C benchmark. We observe the same pattern in this case as in LU class C benchmark. The 32 thread dual stream run fares only slightly better than 16 thread single stream run due to smaller operating data set per thread resulting in significant reduction in L1 data cache misses.

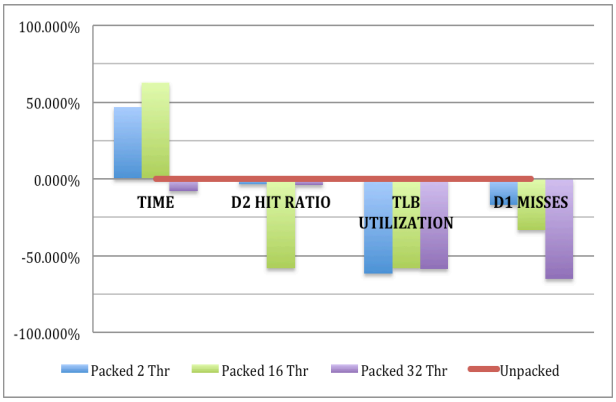

Figure 6. FT Class C benchmark

In figure 7, we examine the PGADGET results for single and dual stream cases with 2 and 16 processes and single stream 32 processes. The input dataset was maintained constant across all the run configurations. The code is run using only MPI programming model. Using we CrayPat we determined the size of operating dataset per thread in each case as shown in table 1. For this paper, we have used a sample data set appropriate for a single Blue Waters XE6 node.

We continue to see contention at level 1 and level 2 data cache in the dual stream 2 and 16 processes run. As the size of the operating dataset diminishes to 167 MB in case of the dual stream 32 processes case, we observe significant improvement in cache utilization, compared to the single stream 16 processes run. CrayPat reports less than desirable TLB utilization in all dual stream experiments.

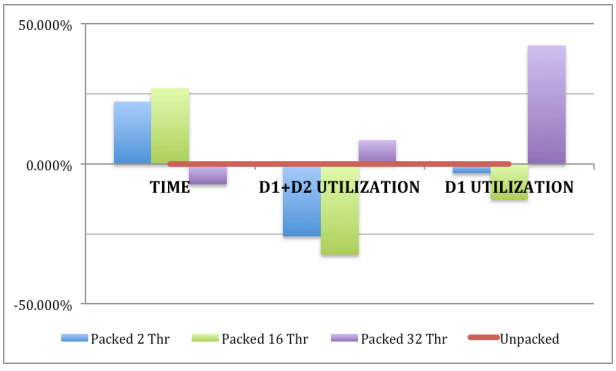

Figure 7. PGADGET Application

Num. MPI Processes	Memory per Process
2	2096 MB
16	295 MB
32	167 MB

Table 1. Memory per MPI process

We now present CASTRO performance data. MPI+OpenMP programming model was utilized in this experiment to spawn one MPI process and either 8, 16 or 32 OpenMP threads. A single data set was used to collect performance data for all the configurations. Figure 8 provides performance of each run relative to 8 threads single stream mode.

In this case, 16-thread dual stream provides 30% performance improvement over 8-thread single stream configuration. In both the cases, the application maps completely to a single Interlagos MCM and has equal access to processor and system resources.

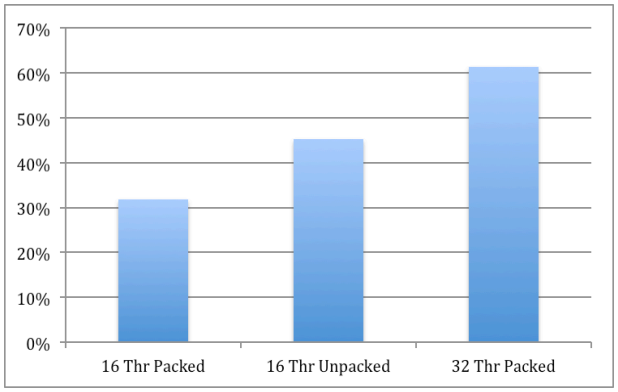

Figure 8. CASTRO Application

And when utilizing both the Interlagos MCM's on a Blue Waters XE6 node, 32-thread dual stream provides a 29% performance improvement over 16-thread single stream configuration. Here as well, in both the cases, the application is mapped completely to both the Interlagos MCM's on a single XE6 node and has access to processor and system resources.

In case of this application, dual stream mode offers better performance than single stream mode.

From the above experiments, we infer both single stream and dual stream modes offer their own unique advantages. Single stream mode offers higher memory bandwidth and larger caches to applications. Dual stream mode offers higher concurrency and possibly better resource utilization with careful cache & TLB management. We have demonstrated applications that perform better in single stream and applications that perform better in dual stream modes. CrayPat and other Cray software offer users with a powerful

set of tools to analyze application codes and identify performance-inhibiting factors.

VI. Summary and Future Work

We have described the Blue Waters super system, its XE6 node architecture and explored in detail the AMD Bulldozer architecture and Interlagos CPU.

In order to explore the dual stream and single stream flexibility of the AMD bulldozer architecture, we configured and conducted performance analysis runs with two benchmark kernels and two full-scale applications chosen for Blue Waters PRAC allocation. We have demonstrated codes that perform better with single stream mode and codes that perform better with dual stream mode. We have successfully employed CrayPat to identify possible resource contention for the shared components of the Bulldozer architecture. Although the selected codes do not represent the breadth of Blue Waters applications, this effort helped us understand the Bulldozer architecture and various usage scenarios.

The Bulldozer architecture also provides greater level of flexibility in the floating-point unit. It supports 128-bit SSE, AVX and 256-bit FMA4 instructions. More studies are needed to specifically target the shared floating-point unit and understand the effects of dual and single stream modes with a mix of SSE, AVX and FMA instructions. Identifying code constructs that cause floating point unit contention will be helpful in preparing petascale applications for Blue Waters.

We will continue to investigate additional features of the AMD Interlagos CPU, the Bulldozer architecture, Blue Waters compute and accelerated node architecture and the Gemini interconnect. In the near future, we plan to experiment with the shared floating-point unit and the Gemini interconnect to explore the performance aspects and optimization techniques.

Acknowledgment

This research is part of the Blue Waters sustained-petascale computing project, which is supported by the National Science Foundation (award number OCI 07-25070) and the state of Illinois. Blue Waters is a joint effort of the University of Illinois at Urbana-Champaign, its National Center for Supercomputing Applications, Cray, and the Great Lakes Consortium for Petascale Computation.

The authors would like to thank the following: Gregory Bauer from NCSA for useful discussions, Kentaro Nagamine, Dept. of Physics & Astronomy, University of Nevada, Las Vegas for providing the PGadget code and members of the Type Ia Supernovae PRAC team, Andy Nonaka at Lawrence Berkeley National Laboratory, Chris Malone at University of California Santa Cruz for providing the CASTRO code, input sets and for working with us in porting the code to the Blue Waters system.

References

[1] Blue Waters : http://www.ncsa.illinois.edu/BlueWaters/

[2] PRAC : http://www.ncsa.illinois.edu/BlueWaters/prac.html

[3] D. Bailey, T. Harris, W. Saphir, R. Van der Wijngaart, A. Woo, and M. Yarrow, "The NAS Parallel Benchmarks 2.0," *NAS Technical Report NAS-95-020*, NASA Ames Research Center, Moffett Field, CA, 1995. http://science.nas.nasa.gov/Software/NPB.

[4] H. Jin, M. Frumkin, J. Yan, "The OpenMP Implementation of NAS Parallel Benchmarks and Its Performance", *NAS Technical Report NAS-99-011*, NASA Ames Research Center, Moffett Field, CA, 1999.

[5] GADGET: A code for cosmological simulations of structure formation, Springel, V. 2005, *MNRAS, 364, 1105*

[6] CrayPat: Cray Performance Analysis Tool, *http://docs.cray.com*

[7] Browne, S., Dongarra, J., Garner, N., Ho, G., and Mucci, P. 2000. A Portable Programming Interface for Performance Evaluation on Modern Processors. *International Journal of High Performance Computing Applications* 14(3):189–204

[8] Vaughan, C., Rajan, M., Barrett, R. F., Doerfler, D., Pedretti, K., (2011). "Investigating the Impact of the Cielo Cray XE6 Architecture on Scientific Application Codes", *Proceedings, IEEE International Parallel and Distributed Processing Symposium, Workshop on Large-Scale Parallel Processing (LSPP)*, Anchorage, Alaska.

[9] How to make best use of the AMD Interlagos processor, *White Paper, Numerical Algorithms Group Ltd*, November 2011.

[10] Ted Barragy, Bulldozer Overview, *Titan Workshop, ORNL*, 2012

[11] Software Optimization Guide for AMD Family 15h Processors, *Publication No. 47414, Rev 3.06*, January 2012

[12] K. Antypas, Y. He, "Transitioning Users from the Franklin XT4 System to the Hopper XE6 System", *CUG Procceedings*, Fairbanks, Alaska, May 23, 2011

[13] Vaughan, C. T. (2011). "Application Characteristics and Performance on a Cray XE6," *Proceedings, Cray User Group Meeting,* Fairbanks, Alaska.

[14] A. Almgren, J. Bell, D. Kasen, M. Lijewski, A. Nonaka, P. Nugent, C. Rendleman, R. Thomas, and M. Zingale. MAESTRO, CASTRO, and SEDONA – *Petascale codes for astrophysical applications. Proceedings of SciDAC* 2010, July 2010.

[15] GADGET : A code for Collisionless and gasdynamical cosmological simulations, Springel V., Yoshida N., White S. D. M., 2001, *New Astronomy, 6, 51*