A Holistic Approach for Performance Measurement and Analysis for Petascale Applications

Heike Jagode[1,2], Jack Dongarra[1,2]
Sadaf Alam[2], Jeffrey Vetter[2]
Wyatt Spear[3], Allen D. Malony[3]

[1] The University of Tennessee
[2] Oak Ridge National Laboratory
[3] University of Oregon
{jagode,dongarra}@eecs.utk.edu
{alamsr,vetter}@ornl.gov
{wspear,malony}@cs.uoregon.edu

Abstract. Contemporary high-end Terascale and Petascale systems are composed of hundreds of thousands of commodity multi-core processors interconnected with high-speed custom networks. Performance characteristics of applications executing on these systems are a function of system hardware and software as well as workload parameters. Therefore, it has become increasingly challenging to measure, analyze and project performance using a single tool on these systems. In order to address these issues, we propose a methodology for performance measurement and analysis that is aware of applications and the underlying system hierarchies. On the application level, we measure cost distribution and runtime dependent values for different components of the underlying programming model. On the system front, we measure and analyze information gathered for unique system features, particularly shared components in the multi-core processors. We demonstrate our approach using a Petascale combustion application called S3D on two high-end Teraflops systems, Cray XT4 and IBM Blue Gene/P, using a combination of hardware performance monitoring, profiling and tracing tools.

Key words: Performance Analysis, Performance Tools, Profiling, Tracing, Trace files, Petascale Applications, Petascale Systems

1 Introduction

Estimating achievable performance and scaling efficiencies in modern Terascale and Petascale systems is a complex task. A diverse set of tools and techniques are used to identify and resolve their performance and scaling problems. In most cases, it is a challenging combination of tasks which include porting the various software components to the target systems, source code instrumentation usually associated with performance tests and management of huge amounts of performance measurement data. The complexity of analysis concepts is extensively described in [1]. In order to address these issues, we propose a methodology for

performance measurement and analysis that is aware of applications and the underlying system hierarchies. On the application level, we measure cost distribution and runtime dependent values for different components of the underlying programming model. On the system front, we measure and analyze information gathered for unique system features, particularly shared components in the multi-core processors.

A detailed analysis of the massivly parallel DNS solver - S3D - on the latest generation of large scale parallel systems allows a better understanding of the complex performance characteristics of these machines. As part of the Petascale measurement effort, we target a problem configuration capable of scaling to a Petaflops-scale system. A complete performance analysis approach requires studying the parallel application in multiple levels such as the computation level, communication level, as well as cache level. This multilayered approach makes it a challenging exercise to monitor, analyze and project performance using a single tool. Hence, a combination of existing performance tools and techniques have been applied to identify and resolve performance and scaling issues. In this context, we are evaluating the collected performance data of S3D using the PAPI library [2] as well as TAU [3], VampirTrace [4], and Vampir [5] toolsets for scalable performance analysis of Petascale parallel applications. Such tools provide detailed analyses that offer important insight into performance and scalability problems in the form of summarized measurements of program execution behavior.

The report is organized as follows: first we provide a brief description of the target systems' features. This is followed by a summary of the applied performance analysis tools. A brief outline of the high-end scientific application that is targeted for this study is provided at the end of section 2. In section 3, we provide extensive performance measurement and analysis results that are collected on the Cray XT4 as well as IBM BlueGene/P system using a set of scalable performance tools. We then provide an insight into the factors resulting in the load imbalance among MPI tasks that cause signifcant idle time for the S3D application runs on a large number of cores. Finally, we outline conclusions and a list of future plans.

2 Background

2.1 Computer Systems

We start with a short description of the key features, most relevant for this study, of the super computer systems that had the following characteristics in December 2008. The Jaguar system at Oak Ridge National Laboratory (ORNL) is based on Cray XT4 hardware. It utilizes 7,832 quad-core AMD Opteron processors with a clock frequency of 2.1 GHz and 8 GBytes of memory (maintaining the per core memory at 2 GBytes). Jaguar offers a theoretical peak performance of 260.2 Tflops/s and a sustained performance of 205 Tflops/s on Linpack [6]. In the current configuration, there is a combination of processing nodes with DDR-667 and DDR-800. Peak bandwidth of DDR-800 is 12.8 GBytes/s. If needed, especially for benchmarking purposes, a user has an opportunity to specify memory bandwidth requirements at the job submission. In this paper we merely use nodes

with 800 MHz memory. The nodes are arranged in a 3-dimensional torus topology of dimension $21 \times 16 \times 24$ with full SeaStar2 router through HyperTransport. The network offers toroidal connections in all three dimensions.

The Argonne National Laboratory (ANL) Intrepid system is based on IBM BlueGene/P technology. The system offers 163,840 IMB PowerPC 450 quad-core processors with a clock frequency of 850 MHz and 2 GBytes of memory (maintaining the per core memory at 500 MBytes). The peak performance of the system is 557 Tflops/s and the sustained Linpack performance is 450.3 Tflops/s [6]. The processors are arranged on a torus network of maximum dimension $40 \times 32 \times 32$, which can be divided into sub-tori - as one example, the smallest torus is of dimension $8 \times 8 \times 8$ - or even smaller partitions with open meshes. An individual partition can not be shared between several users. In contrast to the Cray XT architecture, the BlueGene/P offers the user full control over how the tasks are placed onto the mesh network. At the time of the experiments, hardware counter metrics, such as cache misses and floating point operations were not immediately available on BG/P because of an incomplete PAPI implementation on the platform.

2.2 Performance Analysis Tools

Before we show detailed performance analysis results, we will briefly introduce the main features of the used profiling and tracing tools that are relevant for this paper.

The TAU Performance System is a portable profiling and tracing toolkit for performance analysis of parallel programs [3]. Although it comes with a wide selection of features, for this paper it is mainly used to collect performance profile information through function and loop level instrumentation. TAU profiling collects several metrics for each instrumented function or section of code. Here we focus on the exclusive values. These provide the time or counter value accrued in the body of a function over the program's execution and exclude the time or value accrued in the function's soubroutines. TAU also provides a number of utilities for analysis and visualization of performance profiles.

To analyze details of the S3D application, event-based program traces have been recorded using VampirTrace [4]. The generated *Open Trace Format* (OTF) [7] trace files have then been analyzed using the visualization tool Vampir [5]. Vampir is a tool for performance and optimization that enables application developers to visualize program behavior at any level of detail. For this paper, a considerable amount of performance measurement data has been produced. For that reason, the huge data volume has been analyzed with the parallel version of Vampir [5].

2.3 Application Case Study: Turbulent Combustion (S3D)

The application test case is drawn from the workload configurations that are expected to scale to large number of cores and that are representative of Petascale problem configurations. S3D is a massively parallel DNS solver developed at Sandia National Laboratories. Direct numerical simulation (DNS) of turbulent combustion provides fundamental insight into the coupling between fluid

dynamics, chemistry, and molecular transport in reacting flows. S3D solves the full compressible Navier-Stokes, total energy, species, and mass continuity equations coupled with detailed chemistry. The governing equations are solved on a conventional three-dimensional structured Cartesian mesh. The code is parallelized using a three-dimensional domain decomposition and MPI communication. Ghost zones are constructed at the task boundaries by non-blocking MPI communication among nearest neighbors in the three-dimensional decomposition. Time advance is achieved through a six-stage, fourth-order explicit Runge-Kutta method [8]. S3D is typically run in weak-scaling mode where the sub-grid size remains constant per computational thread.

3 Measurements and Analysis Methodology

As indicated earlier, one of the key goals of this study is to understand and categorize the application behavior on Teraflops-scale leadership computing platforms. In order to achieve this goal, we propose a methodology for performance measurement and analysis that is aware of applications and the underlying system hierarchies as well as the underlying programming model adopted by the application and implemented by the system. We systemetically gathered and analyzed data that enabled us to understand the control flow of critical computation and communication phases, utilization of system features in terms of hardware counter data and behavior of MPI communication operations. The next subsections summarize results collected from a set of profiling, tracing and analysis tools.

3.1 Hardware event data profile

To generally measure the computational efficiency of the S3D application, the ratio of instructions executed per CPU cycle (IPC) has been computed using the following PAPI native events [2]: `IPC = RETIRED INSTRUCTIONS / CPU CLOCKS NOT HALTED`

All the performance analysis utilities summarized in section 2.2 support PAPI counter data. The TAU performance system was used to collect these events. For the hardware performance event measurements, S3D was run on 64 processors on Jaguar. IPC indicates the degree to which the hardware is able to exploit instruction level parallelism in the S3D program [9]. Figure 1 presents the effective instructions per CPU cycle, computed for the 13 most time consuming functions in S3D. Higher is better and low IPC may indicate the presence of performance issues. The AMD Opteron processor can issue up to three AMD64 instructions per cycle and per core [10]. A comparison of the measured IPC with the maximum number reveals the low processor efficiency for most of the time consuming functions.

In order to investigate causes of low IPC, we quantitatively evaluate the performance of the memory sub-system. For that purpose the application was run in SMP (1 core per node) as well as VN mode (4 cores per node). We see a significant slowdown of 25% in VN mode as compared to single-core mode on Jaguar only. Table 1 shows the total execution time for runs on Jaguar and BG/P using the two different modes. We do not see such a serious performance

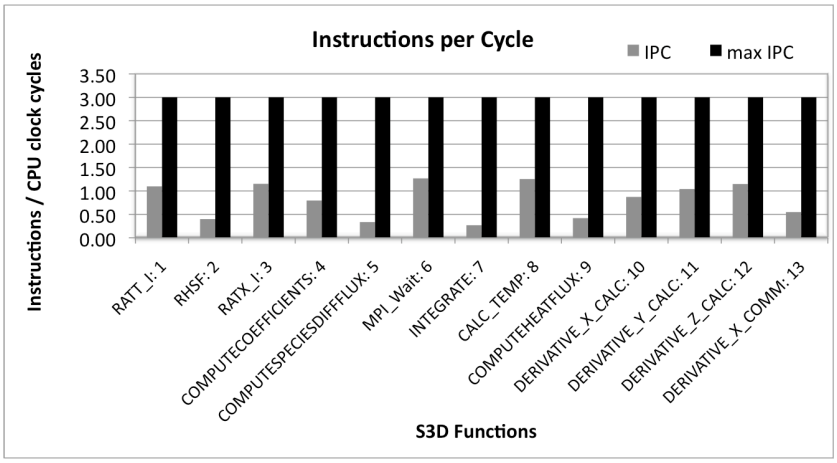

Fig. 1. Instructions per cycle (IPC) for the 13 most time consuming S3D functions (mean)

discrepancy on BG/P. Note, that BG/P also utilizes quad-core nodes but with PowerPC 450 processors that offer a much lower clock frequency (850 MHz) which results in a longer execution time. On both systems - Cray XT4 and BG/P - the L3 cache is shared between all four cores. We collected hardware performance events using the PAPI library [2] that confirms our findings. L3 cache requests are measured and computed using the following PAPI native events: `L3 REQUESTS = READ REQUESTS TO L3 + L3 FILLS CAUSED BY L2 EVICTION`

Note: In VNM all L3 cache measurements have been divided by 4 (4 cores per node on Jaguar)

Figure 2 (a) depicts the number of L3 cache misses and requests when using 4 cores versus 1 core per node for the 13 most expensive functions of the S3D application. It appears that the performance degradation in VN mode is due to the L3 cache behavior. In VN mode we see roughly twice as many L3 cache requests and misses compared to SMP mode. It is not surprising that L3 cache misses increase with VN mode since if every thread is operating on different data, then one thread could easily evict the data for another thread if the sum of the 4 working threads is greater than the size of the L3 cache. However, the increase of L3 requests is rather questionable. The L3 cache serves as a victim cache for L2. In other words, if data is not in L2 cache then L2 TLB checks the L3 cache which results in a L3 request. As mentioned earlier the L3 cache is shared between all four cores while the L2 cache is private. Based on this workflow, it is not clear why the number of L3 requests increases so dramatically when using all 4 cores per node. As verification we measure the L2 cache misses in SMP and VN mode and Fig. 2 (b) presents the comparison. It clearly shows that the number of L2 cache misses does not increase when all four cores are used compared to SMP mode. All the more it is a moot point where the double L3 cache requests come from when VN mode is used. Note that Fig. 2 (a) and (b) use S3D function numbering only while in Fig. 1 the numbers are associated with the corresponding name of the S3D functions.

Table 1. S3D total execution time (s)

Architecture	VNM	SMP
Jaguar	813 s	613.4 s
BG/P	2920.41 s	2918.99 s

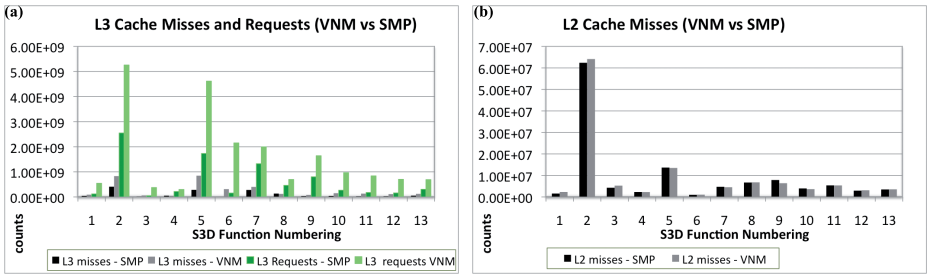

Fig. 2. (a) L3 cache misses and requests (mean); (b) L2 cache misses (mean)

Recent discussions with the research lead [4] for the PAPI project have led us to wonder if this is an artifact of the measurement process. The L3 events in AMD Opteron quad-core processors are not monitored in four independent sets of hardware performance registers but in a single set of registers not associated with a specific core (often referred to as "shadow" registers). There are independent counter registers on each core for most performance events. When an L3 event is programmed into one of these counters on one of these cores, it gets copied by hardware to the shadow register. Thus, only the last event to be programmed into any core is the one actually measured by all cores. When several cores try to share a shadow register, the results are not clearly defined. Performance counter measurement at the process or thread level relies on the assumption that counter resources can be isolated to a single thread of execution. That assumption is generally no longer true for resources shared between cores - like the L3 cache in AMD quad-core nodes. New methods need to be developed to appropriately collect and interpret hardware performance counter information collected from such multi-core systems with interesting shared resources. This is an open area of on-going research.

3.2 Tracing Analysis using Vampir

The results that have been presented so far show aggregate behavior or profile during the appliction run and does not provide an information about the time line of these operations. Unlike profiling, the tracing approach records function calls, messages, etc. as timed events; that is as a combination of timestamp, event type, and even specific data [1]. Tracing experiments allow users detailed observations of their parallel application to understand a time dependent behavior of the

[4] Dan Terpstra is the research lead for the PAPI project: terpstra@eecs.utk.edu

application run. However, the tracing approach also indicates the production of a very large protocol data volume. This is a good time to mention once more the advantage of using a mixture of profiling and tracing tools to overcome those obstacles. Using a profiler prior to the application of a trace tool, for example, can facilitate the exclusion of functions which have a high call frequency [1] and low inclusive time per call. Otherwise, those trivial routines take up most of the time for accessing system clock and maintaining callstack information which makes its timing less accurate. On this account, the TAU profiling tool has been used to create a list of S3D functions that can be excluded from tracing. Turning off those frequently called low level functions results in a S3D trace of a reasonable size while the data accuracy is maintained.

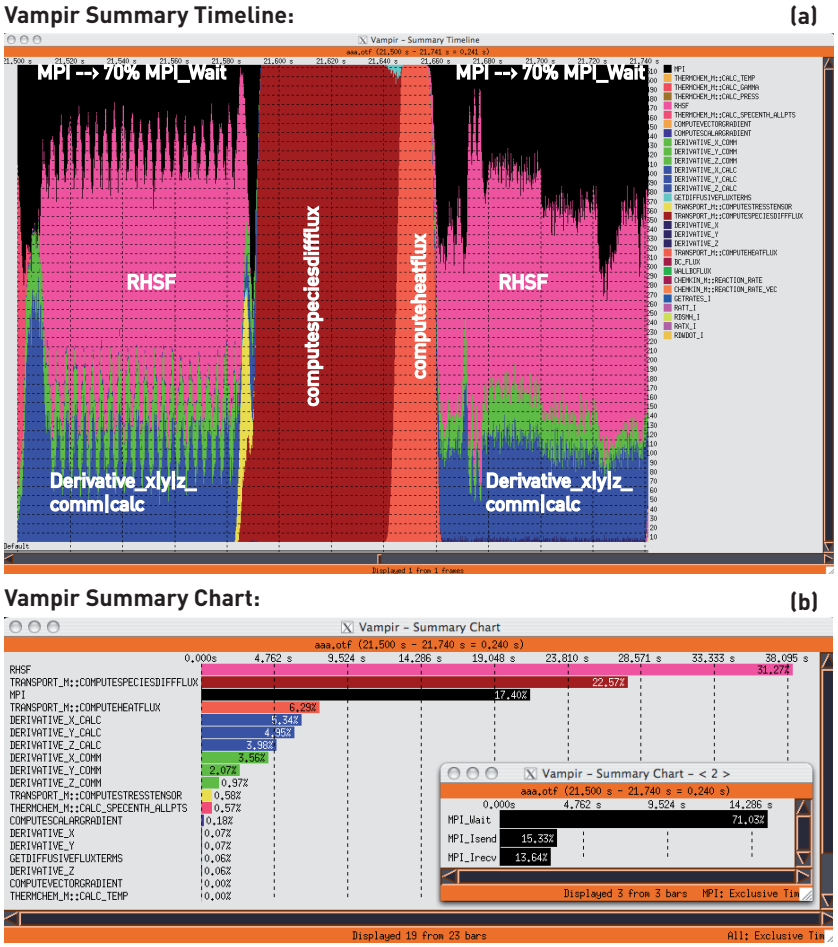

Fig. 3. S3D performance analysis via Vampir

For recording and analyzing event-based program traces, the Vampir suite - a widely recognized tracing facility - has been used. The suite itself is briefly summarized in section 2.2. For the performance analysis via VampirTrace and Vampir, the S3D application has been run on 512 processors in VN mode on Jaguar. Figure 3 (a) presents the number of processes that are actively involved in a given activity at a certain point in time. This information is shown as a vertical histogram. It can be identified how the computation and communication is distributed for a sequence of operations that are repeated in x-, y-, and z-dimensions in one logical time step of the S3D application. Even with visualization tools that graphically present performance data - like TAU and Vampir - for the manual analysis approach some expertise in the field of parallel computing is required to identify possible performance problems [1]. From the high level perspective shown in Fig. 3 (a) together with Fig. 3 (b), it appears that computational work in S3D is not well-balanced in its distribution between the computing resourses. The Summary Chart (Fig. 3 (b)) gives an overview of the accumulated time consumption across all processes and activities. We split MPI times up so that we can see that more than 70% of the entire MPI time is spent in `MPI_Wait`. The rest of the MPI time is spent in non-blocking send and receive operations. Our present main intent is to identify the root causes of load imbalance on large-scale runs and reduction in performance efficiencies on multi-core processors.

3.3 Weak-Scaling Results and Topology Effects on BlueGene/P

Using TAU we collected data from S3D on BlueGene/P for jobs ranging from 1 to 30,000 cores. From this weak-scaling study it was apparent that time spent in communication routines began to dominate as the number of cores increased. A runtime breakdown over trials with increasing numbers of cores, shown in Fig. 4, illustrates this phenomenon. In the 30,000 core case the time spent in routines `MPI_Barrier`, `MPI_Wait` and `MPI_Isend` rose as high as 20.9, 12.7 and 8.7 percent respectively while the time spent in compute routines was not significantly different from lower processor counts.

We further observed deviation between individual threads in time spent in communication routines. The pattern of deviation suggested a load imbalance impacted by node topology. We tested this hypothesis by running an 8000 core test with a random node mapping replacing the default. The default trial had a runtime of 60 minutes and 26.4 seconds. The random node map decreased the runtime to 56 minutes and 47.4 seconds, a speedup of approximately 6%. The change in runtime was almost entirely the result of MPI behavior. The random map saw an average per-thread increase in the `MPI_Wait` routine from 202.3 to 297.4 seconds. However time in `MPI_Barrier` dropped from 349.2 to 48.5 seconds.

The results from the random mapping test indicate that it is possible to improve significantly over BlueGene/P's default mapping for the S3D application. We are presently investigating alternative mapping schemes to find an optimal topology for S3D on the platform.

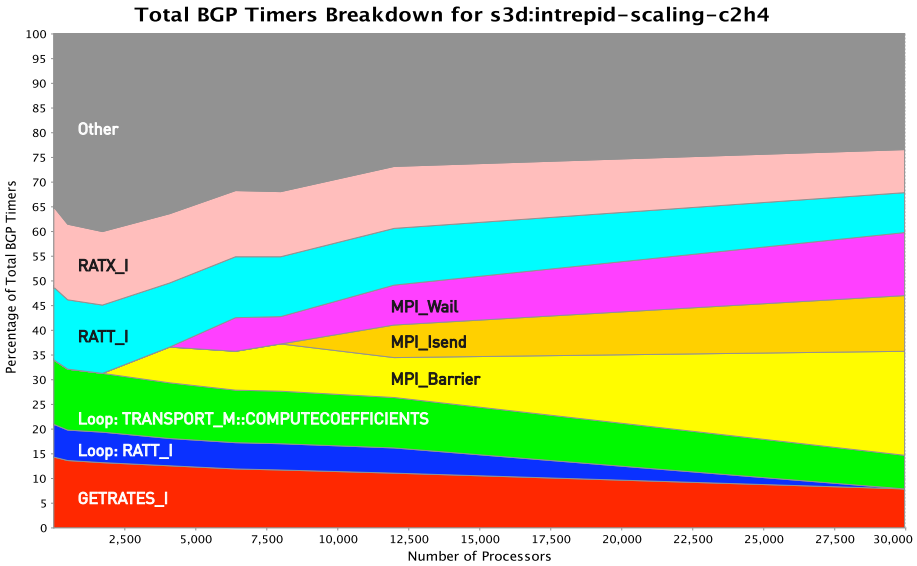

Fig. 4. S3D Scaling Study on BlueGene/P

4 Conclusion and Future Plans

This paper proposes a methodology for performance measurement and analysis that is aware of applications as well as high-end Terascale and Petascale system hierarchies. This approach is demonstrated using a Petascale combustion application called S3D on two high-end systems, Cray XT4 and IBM Blue Gene/P. Our way shows that using a mixture of profiling and tracing tools is highly advisable. It provides important insight into performance and scalability problems by aggregating behavior indicated in profiles with temporal information from the execution time line. This performance analysis approach allowed us to identify a major load imbalance issue when the number of cores are increased. More than 70% of the communication time is spent in `MPI_Wait` on large-scale runs. Our present main intent is to identify the root causes of this behavior.

A deeper investigation of the derivation of time spent in communication routines for individual processes on the Blue Gene architecture shows that the load imbalance is impacted by the node mapping pattern. Using a random instead of the default node mapping confirms the finding and yields a significant performance improvement of about 6% for the entire S3D application. Beside random mapping patterns, we are currently investigating alternative mapping schemes to find an optimal topology for S3D on the Blue Gene/P platform.

Our data collection of hardware performance events shows questionable findings for L3 cache behavior on AMD Opteron processors if all 4 cores on a node are actively used. While only one core can monitor L3 events, it is not clear what happens when several cores try to share the single set of hardware performance registers that is provided to monitor L3 events. Conflicts in measuring those

events related to resources that are shared between cores indicate the need for further research on a portable hardware counter interface for multi-core systems. It is a focus of on-going research in the Innovative Computing Laboratory of the University of Tennessee to develop methods to address this issue.

Acknowledgements

The authors would like to thank the PAPI and Vampir team for their great support. Furthermore, Philip Roth (ORNL) is greatly acknowledged for providing a working S3D version for the BG/P architecture. This research was sponsored by the Office of Mathematical, Information, and Computational Sciences of the Office of Science (OoS), U.S. Department of Energy (DoE), under Contract No. DE-AC05-00OR22725 with UT-Battelle, LLC as well as by the University of Oregon DoE grant from the OoS under Contract No. DE-FG02-07ER25826. This work used resources of the National Center for Computational Sciences at Oak Ridge National Laboratory, which is supported by the Office of Science of the Department of Energy under Contract DE-AC05-00OR22725 and of the Argonne Leadership Computing Facility at Argonne National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract AC02-06CH11357. These resources were made available via the Performance Evaluation and Analysis Consortium End Station, a Department of Energy INCITE project.

References

1. Brunst, H.: Integrative Concepts for Scalable Distributed Performance Analysis and Visualization of Parallel Programs, PhD Dissertation, Shaker Verlag (2008)
2. PAPI Documentation: `http://icl.cs.utk.edu/papi`
3. TAU User Guide: `www.cs.uoregon.edu/research/tau/docs/newguide/index.html`
4. Jurenz, M.: VampirTrace Software and Documentation, ZIH, TU Dresden: `http://www.tu-dresden.de/zih/vampirtrace`
5. VampirServer User Guide: `http://www.vampir.eu`
6. Top500 list: `http://www.top500.org`
7. Knüpfer, A., Brendel, R., Brunst, H., Mix, H., Nagel, W. E.: Introducing the Open Trace Format (OTF), Proceedings of the ICCS 2006, part II. pp. 526-533 (2006)
8. Kennedy, C. A., Carpenter, M. H., Lewis, R. M.: Low-storage explicit Runge-Kutta schemes for the compressible Navier-Stokes equations, Applied numerical mathematics 35(3):177-264 (2000)
9. Drongowski, P.: Basic Performance measurements for AMD Athlon 64 and AMD Opteron Processors, (2006)
10. Software Optimization Guide for AMD Family 10h Processors, Pub. no. 40546 (2008)