10181 Abstracts Collection
Program Development for Extreme-Scale Computing
— Dagstuhl Seminar —

Jesus Labarta[1], Barton P. Miller[2], Bernd Mohr[3] and Martin Schulz[4]

[1] BSC, ES
jesus.labarta@bsc.es
[2] University of Wisconsin - Madison, USA
bart@cs.wisc.edu
[3] Forschungszentrum Jülich, DE
b.mohr@fz-juelich.de
[4] LLNL - Livermore, USA
schulzm@llnl.gov

Abstract. From May 2^{nd} to May 7^{th}, 2010, the Dagstuhl Seminar 10181 "Program Development for Extreme-Scale Computing " was held in Schloss Dagstuhl – Leibniz Center for Informatics. During the seminar, several participants presented their current research, and ongoing work and open problems were discussed. Abstracts of the presentations given during the seminar as well as abstracts of seminar results and ideas are put together in this paper. Links to extended abstracts or full papers are provided, if available.

Keywords. Parallel programming, performance analysis, debugging, scalability

10181 Executive Summary – Program Development for Extreme-Scale Computing

From May 2nd to May 7th, 2010, the Dagstuhl Seminar 10181 "Development for Extreme-Scale Computing " was held in Schloss Dagstuhl Leibniz Center for Informatics. During the seminar, several participants presented their current research, and ongoing work and open problems were discussed. This paper provides an executive summary of the seminar.

Keywords: Parallel programming, performance analysis, debugging, scalability

Joint work of: Labarta, Jesus; Miller, Barton P.; Mohr, Bernd; Schulz, Martin

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2010/2674

Toward (More) Scalable, Autonomous Tool Infrastructure

Dorian C. Arnold (University of New Mexico - Albuquerque, US)

This talk describes current and future research in autonomous overlay networks for communication and data analysis for scalable, robust tools and applications. More specifically, I describe ongoing performance enhancements to improve tree-based overlay network (TBON) start-up and run-time execution. I also describe our current and future work in run-time monitoring and modeling that will lead eventually to a self-adaptive overlay network infrastructure to detect and address performance deficiencies.

Keywords: Parallel programming, performance analysis, debugging, scalability

Scalable Root-Cause Analysis

David Böhme (Forschungszentrum Jülich, DE)

Load- or communication imbalance prevents many codes from taking advantage of the ever-increasing parallelism available in today's large-scale machines. Delays in single processes may spread wait states across the entire machine; moreover, when employing complex point-to-point message-passing patterns, these wait states may propagate along far-reaching cause-effect chains. The large temporal and spatial distances between cause and effect may severely complicate the assessment of the actual cost of an imbalance.

In this talk, I will present a scalable root cause analysis approach which automatically identifies delays (wait-state inducing imbalances) and assesses their total impact on wait-state formation. Using a parallel replay of event traces in backward direction, we attribute the cost of wait states in terms of resource waste back on their original causes. The talk introduces the root cause analysis algorithm, and demonstrates the enhanced insight on wait-state formation we gain for codes running at 10,000 processes and more.

Joint work of: Böhme, David; Geimer, Markus; Wolf, Felix

Scalable Tool and Middleware Development using Group File Operations

Michael J. Brim (University of Wisconsin - Madison, US)

The scale of current and proposed High-End Computing (HEC) systems introduces new challenges for users, administrators, applications, and tools and middleware. One such challenge is performing the same file operations across large groups of files on distributed hosts. To address this challenge, we have defined a new idiom, group file operations, that is a natural, intuitive, and portable

abstraction for performing file operations on groups, and designed TBON-FS, a scalable distributed file system supporting group file operations on thousands of distributed files. This talk will first introduce the group file operation idiom and TBON-FS architecture. I will then discuss how group file operations were used to develop several scalable tools for management and monitoring of distributed systems. Finally, a new specification language for describing custom composition of file system name spaces from thousands of independent file servers will be introduced.

Keywords: Group file operations, distributed file systems, name space composition

Full Paper:
ftp://ftp.cs.wisc.edu/paradyn/papers/Brim09GroupFile.pdf

See also: Michael J. Brim and Barton P. Miller, "Group File Operations for Scalable Tools and Middleware", 16th Annual International Conference on High Performance Computing, Cochin, India. December 2009.

Applications of Spectral Analysis in Data Acquisition, Multiplexing Hardware Counters and Architecture Simulation

Marc Casas (Barcelona Supercomputing Center, ES)

We have typically applied spectral analysis, which works detecting the most important frequencies of the application's execution, to reduce the size of tracefiles. This analysis methodology can be applied to a wide range of problems. In particular, two more applications will be described: First, how we can improve the representativeness of information obtained through sampling and, second, how we can extract accurate information through multiplexed hardware counters.

IBM High Productivity Computing Systems Toolkit

I-Hsin Chung (IBM TJ Watson Research Center - Yorktown Heights, US)

High productivity to the end user is critical in harnessing the power of high performance computing systems to solve science and engineering problems. It is a challenge to bridge the gap between the hardware complexity and the software limitations. Despite significant progress in language, compiler, and performance tools, tuning an application remains largely a manual task, and is done mostly by experts. This presentation describes a holistic approach towards automated performance analysis and tuning that is expected to greatly improve the productivity of performance debugging. The approach seeks to build a framework that facilitates the combination of expert knowledge, compiler techniques, and

performance research for performance diagnosis and solution discovery. With the framework, once a diagnosis and tuning strategy has been developed, it can be stored in an open and extensible database and thus be reused in the future. A demonstration is given to show the effectiveness of the approach through the automated performance analysis and tuning of a scientific application.

Keywords: Performance tuning, performance tool, performance analysis

Cray Debugging Support for Large Scale Systems

Luiz DeRose (Cray Inc. - Saint Paul, US)

Large scale systems present significant challenges for debugging. The traditional way of direct management of each control thread when debugging an application is not appropriate to handle thousands of threads of control. The debugging paradigm on these large scale systems need to change from the traditional control-centric to a scalable data-center approach. In order to address these issues, we designed a new set of debugging supporting tools that consists of innovative techniques for productivity and scalability. In this talk I will present the key aspects of these debugging tools.

Keywords: Scalable tools, debugger, data-centric

Development Environment for X10

Evelyn Duesterwald (IBM TJ Watson Research Center - Hawthorne, US)

X10 is a new programming language that provides a new parallel programming model, a new set of tools integrated into Eclipse and new implementation techniques for delivering optimized scalable parallelism. X10 is a type-safe, modern, parallel, distributed object-oriented language intended to be very easily accessible to Java(TM) programmers. It is a member of the Partitioned Global Address Space (PGAS) family of languages, targeted at future low-end and high-end systems with nodes that are built out of multi-core SMP chips with non-uniform memory hierarchies, and interconnected in scalable cluster configurations. X10 is developed along with an Eclipse-based Integrated Development Environment (IDE) to help further increase programmer productivity. In addition to providing a state-of-the-art development environment, X10DT also serves as the platform for delivering advanced development tools, such as concurrency refactoring tools which interactively guide the programmer in the development of scalable parallel X10 code. This talk with provide an overview of the X10 language and the X10 development environment followed by a detailed look at some of the more advanced tools in the X10 IDE. X10 and X10DT are developed open-source and in collaboration with academic partners and all material covered in this talk is available at x10-lang.org.

Effective Performance Measurement at Petascale using IPM2

Karl Fürlinger (University of California - Berkeley, US)

As supercomputers are being built from an ever increasing number of processing elements, the effort required to achieve a substantial fraction of the system peak performance is continuously growing. Tools are needed that give developers and computing center staff holistic indicators about the resource consumption of applications and potential performance pitfalls at scale. To use the full potential of a supercomputer today, applications have to incorporate multilevel parallelism (threading and message passing) and carefully orchestrate File I/O. As a consequence, performance tools must also be able to monitor these system components in an integrated way and at the full scale of the machine.

We present IPM2, a modularized monitoring approach for MPI, OpenMP, File I/O, and other event sources. We describe its implementation design principles, which are targeted for efficiency and minimal application perturbation, and present application studies of using IPM2 at scale.

Keywords: Application Performance monitoring, workload monitoring

Joint work of: Fürlinger, Karl; Wright, Nick; Skinner, David

Full Paper:
http://ipm-hpc.sourceforge.net/

Analyzing PEPC with Open|SpeedShop with 12000 Cores

Jim Galarowicz (The Krell Institute - Ames, US)

A port of the Open|SpeedShop performance tool to the Cray-XT5 platform is in progress. Open|SpeedShop is an open source multi platform Linux performance tool which is targeted to support performance analysis of applications running on both single node and large scale platforms. Although three of the six base experiments are working, no specific scaling optimizations have been done for the Cray-XT5 at this time. In this talk, which includes a demo on 12000 cores, I will present the Open|SpeedShop tool overhead numbers gathered while profiling the PEPC application on the ORNL jaguar computer system using 600, 2400, 6000, and 12000 cores.

Keywords: Open|SpeedShop performance tools

Full Paper:
http://www.openspeedshop.org/

Demo: Scalable Load-Balance Analysis with Libra

Todd Gamblin (LLNL - Livermore, US)

Libra is a scalable load balance analysis tool for quickly assessing the time spent in different regions of a parallel code over time. It has run on up to 65,536 cores on a BlueGene/P system, as well as smaller sizes on Intel clusters. Libra consists of two parts: 1. scalable backend data collection library and 2. Scalable-front-end GUI visualization. The back-end uses PMPI-layer instrumentation to measure the time a code spends between consecutive synchronous MPI callsites, as well as the time within these calls. This data is collected per-timestep, and as the application runs, Libra uses a parallel wavelet algorithm to compress this data and to store it to scalable visualization files. A user can view Libra performance profiles using the GUI, and data from large runs can be visualized effectively using the multi-scale nature of wavelet -compressed data.

This demo will show Libra applied to PFLOTRAN (and possibly PEPC). We will demonstrate how Libra can be used to quickly measure and visualize the load imbalance of different regions of these codes on 10,000 or more cores.

Keywords: Scalable performance measurement, wavelets, in-situ analysis

Full Paper:
http://www.cs.unc.edu/∼tgamblin/pubs/wavelet-sc08.pdf

See also: Todd Gamblin, Bronis R. de Supinski, Martin Schulz, Robert J. Fowler, and Daniel A. Reed. Scalable load-balance measurement for SPMD codes. In Supercomputing 2008 (SC'08), Austin, Texas, November 15-21 2008.

Scalable in-situ Analysis Techniques

Todd Gamblin (LLNL - Livermore, US)

This talk will describe two scalable in-situ analysis techniques for large-scale performance data. The first is the distributed wavelet compression used by Libra and the model used to collect this data. This part of the talk will be closely related to the demo described above, and we will describe our experiences using Libra with S3D at scale.

The second part of this talk will cover a novel distributed O(log(P)) K-Medoids clustering algorithm we have developed that we have scaled to 131,072-process runs. ăThis algorithm can run in less than a second at this scale, making it suitable for detecting equivalence classes on-line in large-scale runs. ăOur algorithm is generic, requiring only a distance metric defined on the data to be clustered. ăWe believe that this algorithm can serve as a foundation for in-situ analysis in a wide variety of tools, from performance analysis to debugging to fault tolerance.

Keywords: Clustering, in-situ analysis, wavelet, MPI load balance, scalability algorithms

Full Paper:
http://www.cs.unc.edu/~tgamblin/pubs/scalable-cluster-ics10.pdf

See also: Todd Gamblin, Bronis R. de Supinski, Martin Schulz, Robert J. Fowler, and Daniel A. Reed. Clustering performance data efficiently at massive scales. In International Conference on Supercomputing (to appear), Tsukuba, Japan, June 1-4 2010.

Online Performance Analysis

Hans Michael Gerndt (TU München, DE)

This presentation will highlight the potential of online performance analysis for current and future HPC architectures. Based on the initial research done in Paradyn, Periscope was developed at Technische Universität München. It is based on a hierarchy of analysis agents that analyze the performance data obtained from thousands of processes while the application is running. If no further investigation is required by the agents, the application is terminated. However, if further investigation is needed although the application terminated, Periscope automatically restarts the application. The search results in a list of performance properties which can be ranked by their severity and inspected via an Eclipse-based GUI.

Keywords: Performance analysis, parallel programming, high performance computing

Full Paper:
http://www.lrr.in.tum.de/~gerndt/home/Research/PERISCOPE/Periscope.htm

Using CEPBA-Tools to Analyze PEPC and PFLOTRAN at Large Scale

Judit Gimenez (Barcelona Supercomputing Center, ES)

Is it possible to use a trace based approach to analyze large scale runs? Many people consider that tools based on traces are not scalable to hundreds of processors, so what about targeting 10,000 cores as the minimum?

We are going to try to apply the techniques developed at BSC to the proposed cases, experimenting with the scalability limits of the analysis and visualization tools based on traces. The experiment will show how the current tools and methodologies we use give some insight to improve the performance of applications running at large scale.

Joint work of: Gimenez, Judit; Servat, Harald; Huck, Kevin; Llort, German; Casas, Marc; Labarta, Jesus
Full Paper:
http://www.bsc.es/plantillaA.php?cat_id=485

Online Adaptive Code Generation and Tuning

Jeff Hollingsworth (University of Maryland - College Park, US)

This talk summarizes our recent efforts using Active Harmony, a light-weight auto-tuning framework, to permit on-the-fly adaptation of parallel applications during production executions. A unique feature of our search-based auto-tuning system is a powerful parallel search backend that leverages parallelism to effectively navigate high dimensional search spaces. The system provides tuning options not only for symbolic parameters but also for parameters that require dynamic code-generation and compilation. Conceptually, this can be viewed as a merger of the traditional and just-in-time compilation. Initial result indicate that generating new code variants at runtime can improve the performance of programs running on 512 processors by up to 40

Keywords: Auto tuning, runtime code generation, paralllel computing

Joint work of: Hollingsworth, Jeff; Tiwari, Ananta

Full Paper:
http://www.dyninst.org/harmony

Performance Diagnosis through Classification of Computation Bursts to Known Computational Kernel Behavior

Kevin A Huck (Barcelona Supercomputing Center, ES)

Current performance tools do a good job at measuring and identifying performance behavior. However, it takes the trained eye of an experienced performance analyst to understand the behavior. Other times, a hardware expert is required to give insight into the complex combination of available hardware counters and relate the behavior back to programming choices. Classification based on selected hardware counters, combined with a diagnosis of those classifications performed by an expert system, can provide identification of inefficient behavior as well as suggested modifications to improve performance.

In this presentation, we will discuss current work in classifying computation bursts in large scale parallel applications to the well understood performance behavior of benchmark kernels. Our framework is based on collecting hardware counters for a set of benchmark kernels on the test system, which is the training set used to build a classification system. The kernels we use are fine grained, in order to focus on fine grained algorithmic choices. The test application is clustered (at run time or post-mortem) in order to find representative computation bursts. Those bursts are then classified to match the benchmark kernels with respect to memory hierarchy spatial locality, memory hierarchy temporal locality, computation intensity, and control flow. The burst classifications are then

passed through an expert system. If the computation bursts are classified as poorly performing benchmark kernels, the expert system provides performance diagnoses of the bursts, and potential code changes and the predicted change in runtime performance.

Joint work of: Huck, Kevin A; González, Juan; Labarta, Jesús

Scalable Methods for Performance and Power Data Collection and Analysis

Karen L. Karavanic (Portland State University, US)

Challenging science goals continue to push the high end of computing to larger and more complex systems. Coupled with this trend is increasing concern about the power consumption of data centers and computer laboratories, which in some cases matches or exceeds the resources required to power a small city. These trends together drive a need for a new, integrated approach to parallel performance analysis that integrates traditional application-oriented performance data with measurements of the physical runtime environment. We have developed the needed infrastructure for combined evaluation of system, application, and machine room performance in the high end environment. We demonstrate the integration of measured performance data from the application, system, and physical room environment, and discuss the challenges encountered.

Joint work of: Karavanic, Karen L.; Marquez, Andres; Knapp, Rashawn; Adihkari, Agniv; Smith, Mike

Short Presentation of Exatec

Bettina Krammer (University of Versailles, FR)

In 2009, CEA, GENCI, Intel and UVSQ entered into an agreement to create an Exascale Computing Research Center named Exatec, which has been operational since March 2010.

The research agenda of the center will include integrating multi petaFLOPS systems, developing advanced performance optimization techniques, and collaborating with end users to optimize supercomputer performance in areas such as energy, geosciences and lifesciences.

Keywords: Exascale computing

Challenges and Successes in MRNet

Matthew LeGendre (University of Wisconsin - Madison, US)

In this talk we will discuss some of the challenges and successes we have had with achieving MRNet scalability on very large machines. MRNet is a Tree Based Software Overlay Network (TBON) for HPC tools.

By arranging computation and communication into a tree-based structure, MRNet is able to achieve logarithmic performance. This allows tools to operate fully on large systems such as LLNL's BG/L and Jaguar. This talk will discuss our goals in building MRNet (scalability, reliability, flexibility, ...), and the challenges we encountered in achieving these goals. We will focus on the challenges presented by the BlueGene and Cray XT architectures. We will also discuss some tools that have achieved scalability using MRNet, such as STAT, Cray ATP, and TAU. Finally we will demonstrate the STAT tool running on 10,000 cores on Jaguar.

Joint work of: LeGendre, Matthew; Krishnan, Madhavi

Petascale Debugging - and Beyond?

David Lecomber (Allinea Software Ltd - Warwick, GB)

In late 2009, Allinea announced it had debugged 220,000 cores within its graphical debugger, Allinea DDT, on the Jaguar Petaflop system at ORNL.

Having changed the architecture of the debugger to a scalable tree, very responsive performance was recorded at this size - quicker than it would have been at 1,000 cores previously.

Allinea is currently scaling every feature of DDT to this massive scale, and developing new ways to help users at scale. In this open discussion, we would like to discuss the current state of the art in scalable debugging, what debuggers should be doing to simplify bugs at scale, and the challenge of Exascale.

Keywords: Exascale, debugging, tools, usability, bugs, petascale

Maximizing Information/Data Ratio at Run-time

German Llort (Barcelona Supercomputing Center, ES)

One of the approaches we're following to deal with the scalability issues of detailed trace-based analysis of large-scale applications is to automatically determine and focus on the most representative regions of the whole execution.

We've developed an on-line framework that applies automatic analysis techniques (clustering and time analysis) in order to filter performance data as it is being collected to minimize the amount of data emitted to the trace, while maximizing the amount of relevant information presented to the analyst.

The combined use of these analysis tools, a tracing toolkit and a software communication network enables us to produce a detailed yet small trace of a representative time interval, along with additional summary reports for the rest of the execution.

Keywords: On-line analysis, tracing, clustering, scalability

See also: G. Llort, J. Gonzalez, H. Servat, J. Gimenez, J. Labarta. On-line detection of large-scale parallel application's structure. In proceedings of 24th IEEE International Parallel and Distributed Processing Symposium. April 19-23, Atlanta, GA, United States.

PFLOTRAN Meets TAU

Allen D. Malony (University of Oregon, US)

While profile measurements are naturally scalable local to threads/processes, analyzing a large dataset consisting of many per-process profile files presents a number of scalability challenges.

Profile merging and identifier unification can scalably produce a compact single profile file without replicating static per-process event information. Partial event filtering further reduces data volume by keeping the number of significant events in profiles by eliminating small routines with exclusive times below a threshold.

There is compelling movitation for applying these scalable techniques online at the end of the application. While the application is still running, opportunities also exist for scalably generating additional useful information like time-series mean profiles and histograms.

We show TAU deployed for PFLOTRAN on 131k processors. We also demonstrate the use of some of these new scalability-support features of the TAU performance tool to collect PFLOTRAN performance data and visualize at large processor counts through a live demonstration on 12,000 Cray XT5 processors.

Keywords: TAU, scalability, PFLOTRAN

Joint work of: Malony, Allen D.; Lee, Chee Wai; Morris, Alan; Spear, Wyatt; Shende, Sameer

Full Paper:
http://tau.uoregon.edu

Hybrid Parallel Performance Measurement and Analysis

Allen D. Malony (University of Oregon, US)

Modern parallel performance measurement systems collect performance information either through probes inserted in the application code or via statistical sampling. Probe-based techniques measure performance metrics directly using calls to a measurement library that execute as part of the application. In contrast, sampling-based systems interrupt program execution to sample metrics for statistical analysis of performance. Although both measurement approaches are represented by robust tool frameworks in the performance community, each has its strengths and weaknesses. In this talk, we investigate the creation of a hybrid

measurement system, the goal being to exploit the strengths of both systems and mitigate their weaknesses. We show how such a system can be used to provide the application programmer with a more complete analysis of their application.

Simple example and application codes are used to demonstrate its capabilities.

We also show how the hybrid techniques can be combined to provide real cross-language performance evaluation of an uninstrumented run for mixed compiled/interpreted execution environments (e.g., Python and C/C++/Fortran).

Joint work of: Morris, Alan; Malony, Allen D.; Shende, Sameer; Huck, Kevin

Scalable Event Tracing on High-End Parallel Systems

Kathryn Mohror (LLNL - Livermore, US)

Event traces are required to correctly diagnose a number of performance problems that arise on today's highly parallel systems. Unfortunately, the collection of event-based data presents scalability challenges: the large volume of collected data increases tool overhead, and results in data files that are difficult to store and analyze. To address these challenges, we present a new performance measurement method: trace profiling. Trace profiling reduces tracing overhead and volume by identifying repeating trace patterns and retaining only one representative of each pattern. In this talk, we report on our experiences with this approach: we explore the challenges of determining the similarity of sections of traces, i.e., identifying patterns, across time and across processes. We show that we can gather reduced traces that still retain the information needed to correctly diagnose performance problems, with file sizes as small as 4

Keywords: High-performance computing, performance measurement, event tracing

Joint work of: Mohror, Kathryn; Karavanic, Karen L.

Reports from Various Exascale Computing Initiatives

Bernd Mohr (Jülich Supercomputing Centre, DE)

This presentation will report on various Exascale computing initatives the author is involved in. The ExaScale Innovation Center (EIC) is a cooperation between IBM Germany and the Jülich Supercomputing Centre. The European Exascale Software Initiative (EESI) is an European FP7 project.

The EESI main goal is to build a European vision and roadmap to address the challenge of performing scientific computing on the new generation of computers which will provide multi-Petaflop performances in 2010 and Exaflop performances in 2020. The International Exascale Software Program (IESP) is an international effort to define a roadmap for building an open-source software infrastructure for Exascale computing systems.

ScalaTrace and Beyond: Ultra-scalable Tracing, Analysis and Modeling of HPC Codes

Frank Mueller (North Carolina State University, US)

Characterizing the communication and I/O behavior of large-scale applications is a difficult and costly task due to code/system complexity and their long execution times. An alternative to running actual codes is to gather their communication and I/O traces and then replay them, which facilitates application tuning and future procurements. While past approaches lacked lossless scalable trace collection, we contribute an approach that provides orders of magnitude smaller, if not near constant-size, communication and I/O traces regardless of the number of nodes while preserving structural information. We introduce intra- and inter-node compression techniques of MPI and I/O events, we develop a scheme to preserve time and causality of events, and we present results of our implementation for several systems (BlueGene/L+P, Jaguar). Given this capability, the talk emphasizes the impact our approach on communication tuning, multi-level I/O tracing, trace extrapolation and their respective projection onto exascale computing.

Keywords: Parallel computing, high-performance computing, performance analysis, tracing

Scalable Performance Analysis with the Vampir Toolset

Matthias S. Müller (TU Dresden, DE)

The Vampir tool suite for performance analysis provides detailed insight into parallel application behavior. The Vampir tools use a tracing approach to capture application behavior including communication on a fine-grained level of detail. We present performance analysis results for PFLOTRAN and PEPC on up to 16384 processes. Such large scale application runs entail two major challenges for our trace based analysis: First, the total amount of trace data increases with scale. Second, the visualization and analysis of thousands of interacting parallel elements is challenging.

To overcome the first issue the measurement environment of the Vampir suite provides new process specific filtering rules. As a sequential analysis of large scale trace data is infeasible, we use a parallel trace analysis. The visualization challenge of large scale traces is addressed with so called "scalable" displays for performance analysis. These displays use filtering and clustering approaches to reduce the amount of data that is presented.

Keywords: Performance analysis, tracing, scalability

Joint work of: Müller, Matthias S.; Hilbrich, Tobias; Brunst, Holger

Full Paper:
http://www.vampir.eu

PFLOTRAN Performance Analysis Using the Cray Toolset

Heidi Poxon (Cray Inc. - Saint Paul, US)

The Cray performance analysis tools provide an integrated infrastructure for measurement and analysis of application computation and communication. The toolset allows developers to collect information at process and thread levels, and provides analysis of data including load imbalance, derived metrics based on hardware events, and optimal MPI rank placement strategies. The toolset functionality will be demonstrated with the application PFLOTRAN. Performance data will be shown at 16,000 MPI ranks. Analysis of the data will be presented through Cray Apprentice2, showing how users are guided to relevant performance bottlenecks.

Toward Performance Prediction of Tree-Based Overlay Networks on the Cray XT

Philip Roth (Oak Ridge National Lab., US)

Tree-Based Overlay Networks (TBONs) provide a scalable communication and data manipulation infrastructure for tools and applications. There is a great deal of flexibility in mapping TBON processes to a parallel system's resources, but some process mappings will provide better performance than others. In this talk, we present early work on an infrastructure for predicting the performance of software that uses MRNet, the initial TBON implementation, as a communication and data manipulation infrastructure. Because of the difficulty in constructing closed form expressions describing overlay network performance for arbitrary network topologies on some platforms, our approach uses parallel discrete event simulation for performance prediction. The target platform for our work is the Cray XT, such as the Jaguar system deployed in the Leadership Computing Facility at Oak Ridge National Laboratory.

Keywords: Tree-based overlay networks, performance prediction, Cray XT

Benefits of Sampling in Tracefiles

Harald Servat (Barcelona Supercomputing Center, ES)

In order to deal with large runs, our instrumentation package is able to selectively instrument the large computation bursts of an application. Also, to complement that information, it also computes some statistics for the non computation regions. The obtained tracefile results with an important space reduction and provides a general insight of the application behavior.

Sampling, in contrast with instrumentation, can provide application details beyond the instrumented points. Sampling not only provides information on

uninstrumented regions of code, but also it is not tied with the application granularity. In addition, the amount of data generated by the sampling can be easily tuned by simply choosing a different frequency, making it an excellent companion for the instrumentation.

In this session we will show how to tackle the scalability of traces combining both mechanisms.

Keywords: Tracing, sampling, instrumentation, scalability

Combining PMPI Event Profiling and Clock Sampling in Scalasca

Zoltan Szebenyi (Forschungszentrum Jülich, DE)

The challenges encountered while implementing sampling support in the Scalasca toolset are introduced. This new capability is especially valuable for measuring applications where the intrusion of instrumentation in frequently executed short functions is unacceptable. Profile summaries combining exact message-passing event statistics with representative call-path timings, within the highly scalable Scalasca measurement infrastructure, are evaluated with an early prototype.

Joint work of: Szebenyi, Zoltan; Gamblin, Todd; Schulz, Martin; de Supinski, Bronis; Wolf, Felix; Wylie, Brian

Identifying Scalability Bottlenecks In Large-scale Parallel Programs Using HPCToolkit

Nathan Tallent (Rice University, US)

HPCToolkit is an integrated suite of tools that supports measurement, analysis, attribution, and presentation of application performance for both sequential and parallel programs. HPCToolkit can attribute very precise (instruction-level) measurements to source-level static and dynamic contexts in fully optimized applications, all for an average run-time overhead of a few percent. To enable rapid performance analysis, HPCToolkit presents the resulting context-sensitive metrics in three complementary views. In this demonstration, we highlight HPCToolkit's ability to pinpoint and quantify (a) scalability bottlenecks in general and (b) load imbalance in particular within large scale programs.

Keywords: HPCToolkit, performance analysis, performance tools, load imbalance, call path profiling

Joint work of: Tallent, Nathan; Mellor-Crummey, John; Adhianto, Laksono; Fagan, Mike; Krentel, Mark

Full Paper:
http://www.hpctoolkit.org

Towards an Automatically Distributed Evaluation of Event Data

Roland Wismüller (Universität Siegen, DE)

When monitoring large scale parallel programs on-line, a huge amount of event data is produced, which should be evaluated and thus compressed as close to the event source as possible. I.e., event processing must be performed in a distributed way. On the other hand, the monitoring system should be fully programmable, in order not to restrict the tools' functionality. Traditionally, the combination of both requirements means that the tool developer (or even the tool user, if the tool allows a user-defined evaluation) must deal with the complexity of distributing the evaluation process. A second source of complexity results from the experience that in on-line tools, all event processing must occur as asynchronous as possible, i.e., event driven.

The talk presents a concept to overcome this complexity. The idea is to allow the tool programmer (or user) to specify the evaluation via a 'normal', sequential Java program, which makes use of a class library offering completely asynchronous method calls as well as extensions for object sets and stream processing. The basic mechanism is an extension of the concept of 'futures' implemented in ProActive[1]: when a method is called, it immediately returns a future representing the result, while the method is processed 'in background' (usually on a remote computer). While ProActive blocks and waits for the real result to become available whenever the future is used, i.e., a method is called on it, our approach recursively uses the same approach to handle this method call: the call immediately returns another future and executes asynchronously in the background. In this way, a data flow graph develops. This basic concept is extended with automatically generated classes to represent sets of objects and object streams. In a monitoring system, this is used for results of different nodes, and event streams, respectively. A method called on one of these special objects will be executed (asynchronously) on the whole set or stream of objects and again returns (a future representing) a set or stream of results. Like before, the method calls result in nodes being added to the data flow graph. In this way, when the Java code is executed, a data flow graph is built, which can then automatically be analysed and distributed to the nodes of the monitored system for execution. This has already been demonstrated by our work on G-PM in the CrossGrid project[2,3].

References

1. F. Baude, L. Baduel, D. Caromel, A. Contes, F. Huet, M. Morel and R. Quilici. Programming, Composing, Deploying for the Grid. In GRID COMPUTING: Software Environments and Tools, Springer Verlag, January 2006. http://proactive.inria.fr/userfiles/file/papers/ProgrammingComposingDeploying.pdf
2. R. Wismüller, H. Mehammed, M. Gerndt, and A. Bode. "Performance Monitoring and Analysis for the Grid. In B. Di Martino et al., editors, Engi-

neering the Grid, chapter 16. American Scientific Publishers, January 2006. http://www.aspbs.com/grid.html
3. R. Wismüller, M. Bubak, W. Funika, T. Arodz, and M. Kurdziel. Support for User-Defined Metrics in the On-line Performance Analysis Tool G-PM. In Grid Computing – Second European AcrossGrids Conference AxGrids2004, LNCS 3165, pages 159-168, Nicosia, Cyprus, January 2004. Springer-Verlag. http://www.bs.informatik.uni-siegen.de/web/wismueller/pub/axgrids04.ps.gz

Keywords: Online monitoring, distributed system, dataflow, asynchronous evaluation, Java

Experiences with Scalasca at Scale

Brian J. N. Wylie (Jülich Supercomputing Centre, DE)

The open-source Scalasca toolset [www.scalasca.org] supports integrated runtime summarization and automated trace analysis on a diverse range of HPC computer systems. Developed from the outset for scalability, various usability impediments have been identified and re-engineered in successive releases as scalability requirements have grown. Analyses of the Dagstuhl challenge applications PEPC and PFLOTRAN on IBM BlueGene/P and Cray XT systems at scale are demonstrated, followed by an examination of associated scalability issues.

Keywords: Open-source, parallel application execution performance analysis, scalability

Joint work of: Wylie, Brian J. N.; Böhme, D; Geimer, M.; Szebenyi, Z.

Full Paper:
http://www.scalasca.org

Tool Strategies for Sequoia and Beyond

Bronis R. de Supinski (LLNL - Livermore, US)

In FY12, LLNL will field Sequoia, a next generation BlueGene system that will feature over 1.6 million cores and over 6 million threads. This unprecedented level of parallelism will pose new challenges for application developers that will require a scalable and robust set of development environment tools. LLNL will develop this tool set in close collaboration with IBM and independent software vendors as well as the open source community. The wider community will play a particularly important role as the challenges are too large for any one supercomputing site or vendor. In this talk, I will present the Sequoia tool strategy including details of the current and emerging infrastructure, such as the GIG interface, upon which we plan to build.

Keywords: Sequoia, GIG, infrastructure