Deliverable D2.2

Final Tool Set

CONTRACT NO	HOPSA-EU 277463
INSTRUMENT	CP (Collaborative project)
CALL	FP7-ICT-2011-EU-Russia

Due date of deliverable: November 1st, 2012

Actual submission date: January 15th, 2013

Start date of project: 1 FEBRUARY 2011 Duration: 24 months

Name of lead contractor for this deliverable: TUD

Abstract: This report describes the final status of the HOPSA integrated tool set for the instrumentation, measurement and analysis of parallel programs consisting of the memory and thread analyzer ThreadSpotter (RW), the trace visualizer Paraver including its measurement system Extrae (BSC), the performance prediction tool Dimemas (BSC), the trace visualizer Vampir (TUD), the performance measurement and analysis tool Scalasca (GRS, JSC), and the instrumentation and measurement system Score-P (GRS, JSC, TUD).

Project co-funded by the European Commission within the Seventh Framework Programme (FP7/2007-2013)		
Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Table of Contents

Glossary

Abbreviation / acronym	Description
API	Application Programming Interface
BSC	Barcelona Supercomputing Center, Spain
CEPBA	European Center for Parallelism of Barcelona (UPC, BSC)
CUBE	Performance report explorer for Scalasca (JSC)
CUDA	Compute Unified Device Architecture (Programming Interface for Nvidia GPGPUs)
Dimemas	Message passing performance analysis and prediction tool (BSC)
Extrae	Instrumentation and measurement component for Paraver visualizer (BSC)
GRS	German Research School for Simulation Sciences GmbH, Aachen, Germany
GPGPU	General Purpose Graphical Processing Unit
GUI	Graphical User Interface
HMPP	Hybrid Multicore Parallel Programming (Programming Model for Heterogeneous Architectures)
HOPSA	HOlistic Performance System Analysis. EU FP7 project
HPC	High Performance Computing
H4H	Hybrid Programming For Heterogeneous Architectures. EU ITEA2 project
I/O	Input/Output
JSC	Jülich Supercomputing Centre (of Forschungszentrum Jülich GmbH), Germany
MPI	Message Passing Interface (Programming Model for Distributed Memory Systems)
OpenCL	Open Computing Language (Programming interface for heterogeneous platforms consisting of CPUs and other execution units like GPUs)
OpenMP	Open Multi-Processing (Programming Model for Shared Memory Systems)
OTF2	Open Trace Format Version 2
Paraver	Event trace analysis and visualization tool (BSC)
RW	Rogue Wave Software AB, Sollentuna, Sweden
Scalasca	SCalable Analysis of LArge SCale Applications (Performance instrumentation, measurement and analysis tool from JSC/GRS)

Score-P	Scalable Performance Measurement Infrastructure for Parallel Codes (Community open-source project of GRS, JSC, TUD and others)
SMPSs	Pragma-based programming model for parallel task (Ss = Superscalar) for shared memory parallel computers (SMP) from BSC
UPC	Universitat Politècnica de Catalunya, Barcelona
ThreadSpotter	Commercial memory and multi-threading performance analysis tool (RW)
TUD	Technische Universität Dresden, Germany
Vampir	Visualization and Analysis of MPI Resources (Commercial event trace analysis and visualization tool from ZIH/TUD)
VampirTrace	Instrumentation and measurement component for Vampir visualizer (ZIH/TUD)
ZIH	Zentrum für Informationsdienste und Hochleistungsrechnen. (Center for information services and HPC of TUD).

1. Executive summary

This report describes the status of the integrated tool set for the instrumentation, measurement and analysis of parallel application programs developed in work package 2 of the EU FP7 project HOPSA.

1.1 The broader context: The HOPSA project

To maximize the scientific and commercial output of a high-performance computing system, different stakeholders pursue different strategies. While individual application developers are trying to shorten the time to solution by optimizing their codes, system administrators are tuning the configuration of the overall system to increase its throughput. Yet, the complexity of today's machines with their strong interrelationship between application and system performance demands for an integration of application and system programming.

The HOPSA project (HOlistic Performance System Analysis) therefore sets out for the first time for combined application and system tuning developing an integrated diagnostic infrastructure. Using more powerful diagnostic tools application developers and system administrators will easier identify the root causes of their respective bottlenecks. With the HOPSA infrastructure, it is more effective to optimize codes running on HPC systems. More efficient codes mean either getting results faster or being able to get higher quality or more results in the same time.

The work in HOPSA is carried out by two coordinated projects funded by the EU under call FP7-ICT-2011-EU-Russia and the Russian Ministry of Education and Science. Its objective is the new innovative integration of application tuning with overall system diagnosis and tuning to maximize the scientific output of our HPC infrastructures. While the Russian consortium will focus on the system aspect, the EU consortium will focus on the application aspect.

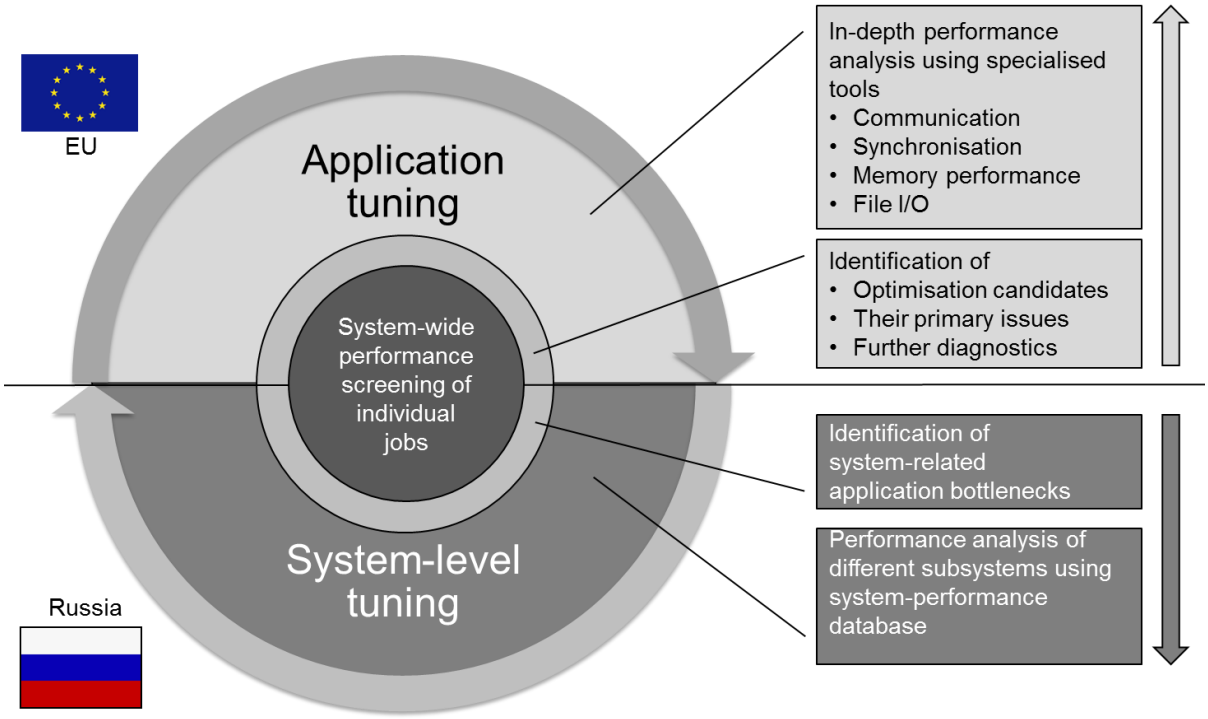

Figure 1: System-level tuning (bottom), application-level tuning (top), and system-wide performance screening (centre) use common interfaces for exchanging performance properties.

At the interface between these two facets of our holistic approach, which is illustrated in the Figure 1, will be the system-wide performance screening of individual jobs, pointing at both inefficiencies of individual applications and system-related performance issues.

1.2 Work package 2: HPC application-level performance analysis

For HPC application tuning, developers can choose from a variety of mature performance-analysis tools developed by the HOPSA-EU consortium: the memory and thread analyzer ThreadSpotter (RW), the trace visualizer Paraver including its measurement system Extrae (BSC), the performance prediction tool Dimemas (BSC), the trace visualizer Vampir (TUD), the performance measurement and analysis tool Scalasca (GRS, JSC), and the instrumentation and measurement system Score-P (GRS, JSC, TUD).

Within work package 2 of the HOPSA project, the tools were further integrated and enhanced with respect to scalability, depth of analysis, and support for asynchronous tasking, a node-level paradigm playing an increasingly important role in hybrid programs on emerging hierarchical and heterogeneous systems. The overall objective of work package 2 was to enhance and extend the already existing individual performance measurement and analysis tools of the project partners to make them fit for the analysis of petascale computations and beyond as well as integrating them with each other where useful. The idea here was not to start new research directions but rather to finalize (i.e., “productize”) current research ideas and make them part of the regular tool products. The tools are available as a combination of open-source offerings (Extrae, Paraver, Dimemas, Scalasca, CUBE, Score-P) and commercial products (Vampir, ThreadSpotter). At the end of the project (January 2013), a single unified installation package for all tools will be provided.

Work package 2 contains all research and technical development which only involves EU partners. Integration with the system-level analysis of the Russian HOPSA partners is the subject of work package 3 and therefore not covered in this deliverable. In the next chapter, the final status of the HOPSA application-level tools is described and the work done as part of work package 2 of the HOPSA-EU project is briefly summarized. A more complete and detailed description of this work is provided by deliverable D1.2 (“Final Progress Report”).

1.2.1 The HOPSA workflow

The indented usage and application of the HOPSA performance tools is specified by the HOPSA performance-analysis workflow (Figure 2). It consists of three basic steps. During the first step (“Performance Screening”), we identify all those applications running on the system that may suffer from inefficiencies. This is done via system-wide job screening supported by a lightweight measurement module (LWM^2) dynamically linked to every executable. The screening output identifies potential problem areas such as communication, memory, or file I/O, and issues recommendations on which diagnostic tools can be used to explore the issue further in a second step (“Performance Diagnosis”). If a more simple, profile-oriented static performance overview is not enough to pin-point the problem, a more detailed, trace-based, dynamic performance analysis can be performed in a third step (“In-depth analysis”). Available application performance analysis tools include Paraver/Dimemas [7, 8, 9], Scalasca [1, 2, 3], ThreadSpotter [4], and Vampir [5]. The data collected by LWM^2 is also fed into the Clustrx.Watch hierarchical cluster monitoring system [10] which combines it with system and hardware data and forwards it to the LAPTA cluster monitoring and analysis system [11] for further analysis by system administrators.

In general, the workflow successively narrows the analysis focus and increases the level of detail at which performance data are collected. At the same time, the measurement configuration is optimised to keep intrusion low and limit the amount of data that needs to be stored. To distinguish between system and application-related performance problems, Paraver and Vampir allow also system-level data to be retrieved and displayed. The system administrator, in contrast, has access to global performance data. He can use this data to identify potential system performance bottlenecks and to optimise the system configuration based on current workload needs. In addition, the administrator can

identify applications that continuously underperform and proactively offer performance-consulting services to the effected users. In this way, it facilitates reducing the unnecessary waste of expensive system resources.

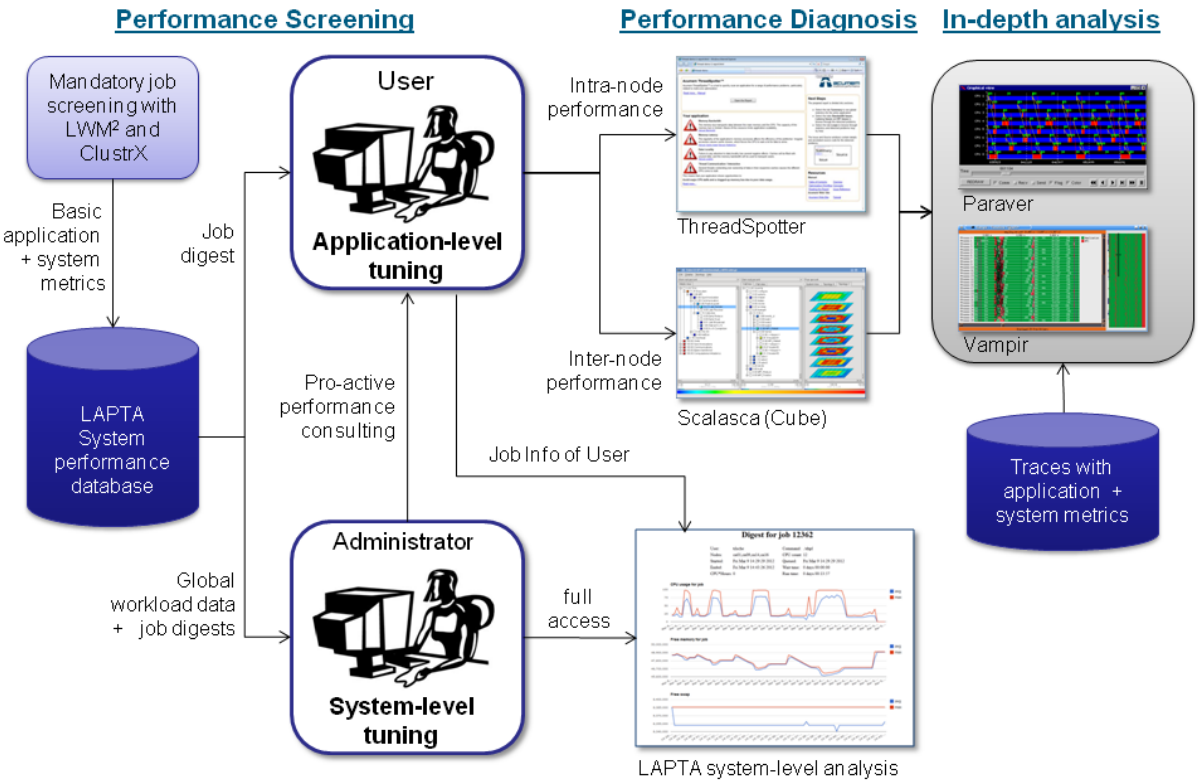

Figure 2: The HOPSA Performance-Analysis Workflow

More details about the HOPSA performance-analysis workflow can be found in the Deliverable D3.2 ("Workflow Report") as well in [12]. The lightweight measurement module (LWM^2) is described in more detail in Deliverable D3.3 ("Light-weight Monitoring Module").

1.2.2 HOPSA tool integration

Sharing the common measurement infrastructure Score-P [6] and its data formats and providing conversion utilities if direct sharing is not possible, the performance tools in the HOPSA environment and workflow already make it easier to switch from higher-level analyses provided by tools like Sclasca to more in-depth analyses provided by tools like Paraver or Vampir. To simplify this transition even further, the HOPSA tools are integrated in various ways (Figure 3). With its automatic trace analysis, Scalasca locates call paths affected by wait states caused by load or communication imbalance. However, to find and fix these problems in a user application, it is in some cases necessary to understand the spatial and temporal context leading to the inefficiency, a step naturally supported by trace visualizers like Paraver or Vampir. To make this step easier, the Scalasca analysis remembers the worst instance for each of the performance problems it recognizes. Then, the Cube result browser can launch a trace browser and zoom the timeline into the interval of the trace that corresponds to the worst instance of the recognized performance problems. In order to allow the use of Paraver for this analysis, the BSC team implemented an OTF2 to Paraver trace format conversion.

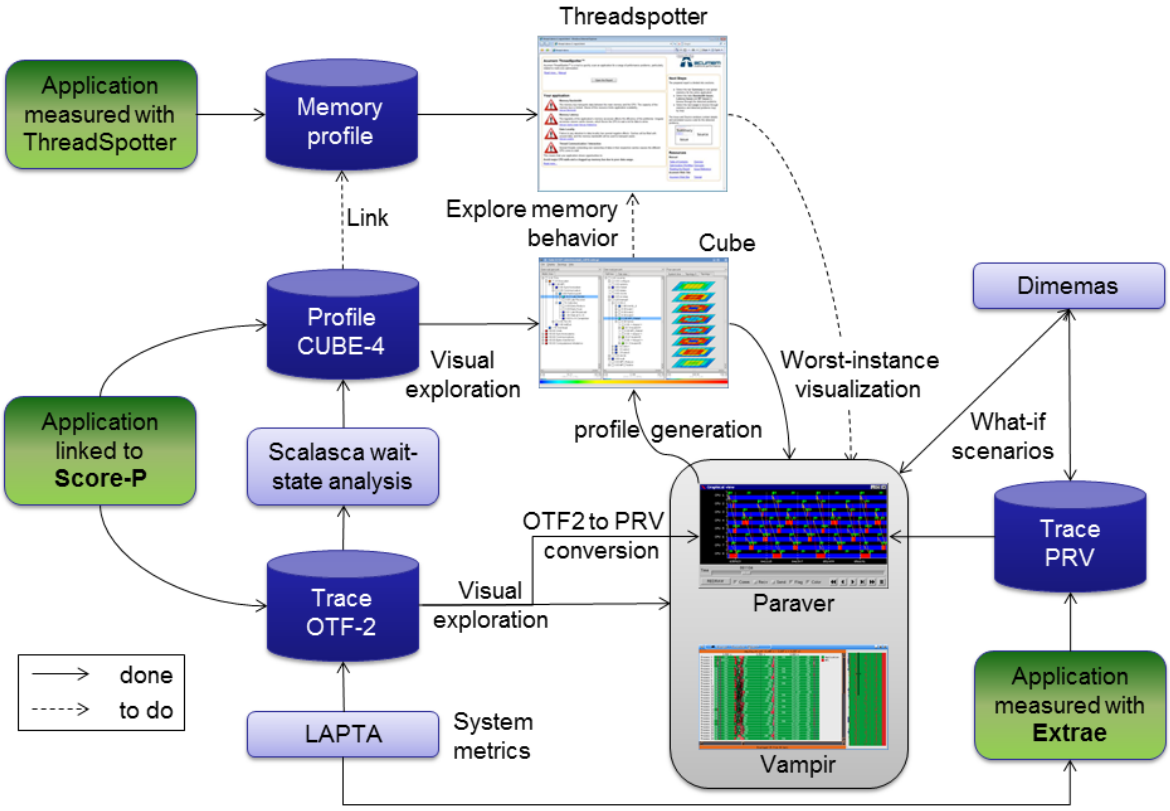

Figure 3: HOPSA Performance Tool Integration

In the future, it is planned to use the same mechanisms for a more detailed visual exploration of the results of Scalasca's root cause analysis as well as for further analyzing call paths involving user functions that take too much execution time. For the latter, ThreadSpotter will be available to investigate their memory, cache and multi-threading behaviour. If a ThreadSpotter report is available for the same executable and dataset, Cube will allow launching detailed ThreadSpotter views for each call path where data from both tools is available. The necessary interfaces have been designed and prototypically implemented during the HOPSA project.

Finally, a tight integration of Dimemas and Paraver allows users to investigate various “what-if scenarios” to further analyze performance properties of their applications.

2. Dimemas, Extrae, Paraver (BSC)

2.1 Basic description

Paraver (http://www.bsc.es/paraver) is a very flexible data browser that is part of the CEPBA-Tools toolkit [7, 8]. Its analysis power is based on two main pillars. First, its trace format has no semantics; extending the tool to support new performance data or new programming models requires no changes to the visualizer, just to capture such data in a Paraver trace. The second pillar is that the metrics are not hardwired into the tool but programmed. To compute them, the tool offers a large set of time functions, a filter module, and a mechanism to combine two timelines. This approach allows displaying a huge number of metrics with the available data. To capture the experts knowledge, any view or set of views can be saved as a Paraver configuration file. After that, re-computing the view with new data is as simple as loading the saved file. The tool has been demonstrated to be very useful for performance analysis studies, giving much more details about the applications behaviour than most performance tools.

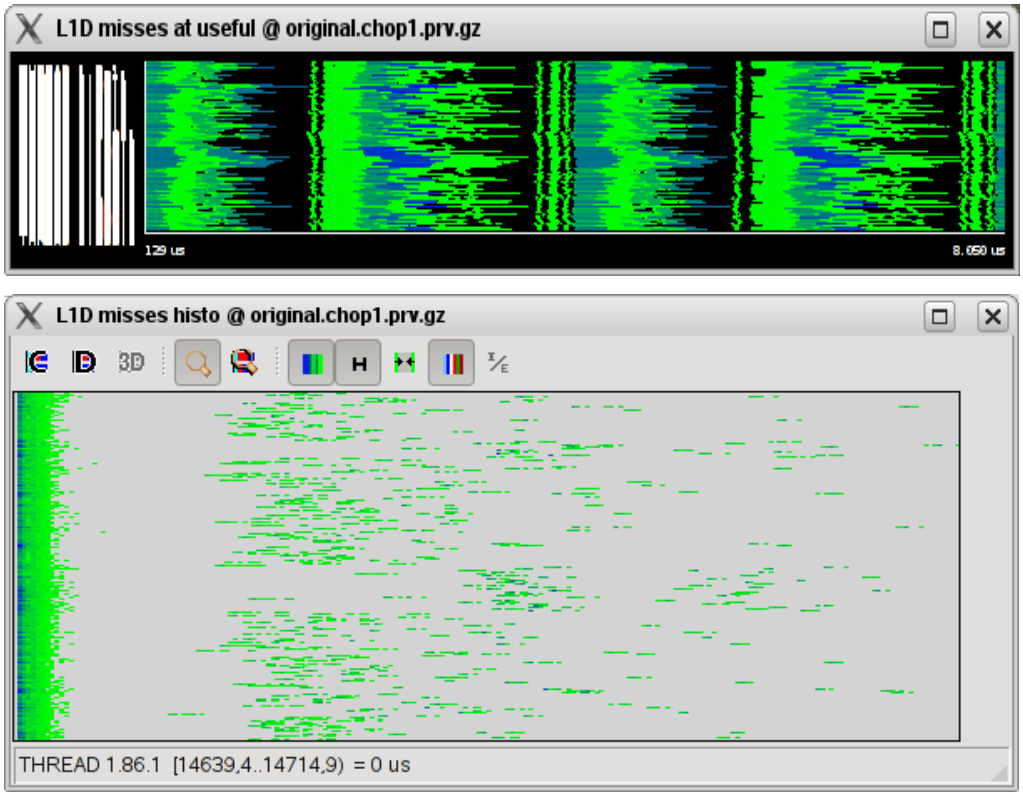

Figure 4. Paraver's two main types of views: timelines and histograms.

Dimemas (http://www.bsc.es/dimemas) is a performance analysis tool for message-passing programs. The Dimemas simulator reconstructs the time behaviour of a parallel application on a machine modelled by the key factors influencing the performance [9]. With a simple model Dimemas allows to simulate complete parametric studies in a very short time frame. The supported target architecture is a cloud of parallel machines, each one with multiple nodes and multiples CPUs per node allowing evaluating a very high range of alternatives. Dimemas generates as part of its output a Paraver trace file, enabling the user to conveniently examine any performance problems indicated by a simulator run.

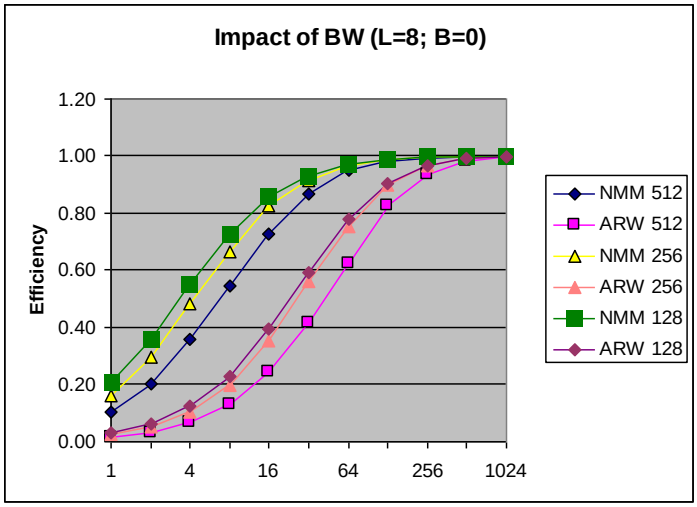

Figure 5. Dimemas parametric study example.

Extrae is the CEPBA-Tools instrumentation package and is composed of a set of programs and libraries to generate or translate Paraver and Dimemas traces. Extrae can instrument different programming models (MPI, OpenMP, POSIX threads, StarSs, CUDA, Cell and their combinations).

CEPBA-Tools have been successfully used for the analysis of large-scale runs in the range of ten thousand processes. Today CEPBA-Tools are developed by BSC and are available for download under an LGPL open-source license.

2.2 Main achievements

In the framework of WP2, BSC efforts during the second year of HOPSA with respect to the tools integration has been focused on:

- A new functionality has been added to Paraver to launch an external analysis tool. The first target has been to run Dimemas simulations driven by Paraver GUI. The user selects an area of analysis and a target configuration and Paraver cuts the tracefile and executes the Dimemas simulation being able to load the output tracefile generated.
- The generation of CUBE files from Extrae folded samples has been extended to include the information required to launch Gnuplot or Paraver tools from CUBE using the interface implemented by JSC and testing this new functionality provided by CUBE.
- A translator from the Score-P traces to Paraver format has been developed so Paraver can be used for a very detailed analysis of the data instrumented by Score-P.

With respect to the tools scalability enhancement, we continued on the working lines initiated during the first year:

- The parallelization of Paraver engine has been extended to the statistics module. With this parallelisation we are targeting local multi core platforms using the BSC programming model OmpSs. The parallelization of the statistics module required some changes on the intermediate structures to eliminate false dependencies between the table rows.
- The improvements on the scalability of Dimemas have been evaluated and validated. The new version improves the performance between 15 and 20 times on traces in the scale of ten thousand processes.

Finally with respect to the tools functionality enhancement the main achievements are:

- Integrate the modeling of StarSs programming model in the Dimemas distribution. The Dimemas distribution now simulates the tasks scheduling and their synchronization enabling to predict the performance when varying the number of cores used by StarSs.
- Translate all the Paraver tutorial guidelines available to on-line format so they can be easily used.
- Extrae allows to automatically initialize and finalize the instrumentation library without requiring to have MPI or to add the corresponding API calls and link with the instrumentation library.
- The instrumentation of Extrae has been extended to include the OpenMP tasks and to support the new Intel MIC (Xeon Phi) accelerators.
- The Extrae support to the OmpSs programming model has been complemented and a publication was presented at [13].

3. Scalasca and CUBE (GRS, JSC)

3.1 Basic description

Scalasca (“SCalable Analysis of LArge SCale Applications”) is a free software tool that supports the performance optimization of parallel programs by instrumenting, measuring and analysing their runtime behaviour [1, 2, 3]. The tool has been specifically designed for use on large-scale systems including IBM Blue Gene and Cray XT, but is also well suited for small- and medium-scale HPC platforms. The analysis identifies potential performance bottlenecks – in particular those concerning communication and synchronization – and offers guidance in exploring their causes. Scalasca mainly targets scientific and engineering applications based on the programming interfaces MPI and OpenMP, including hybrid applications based on a combination of the two.

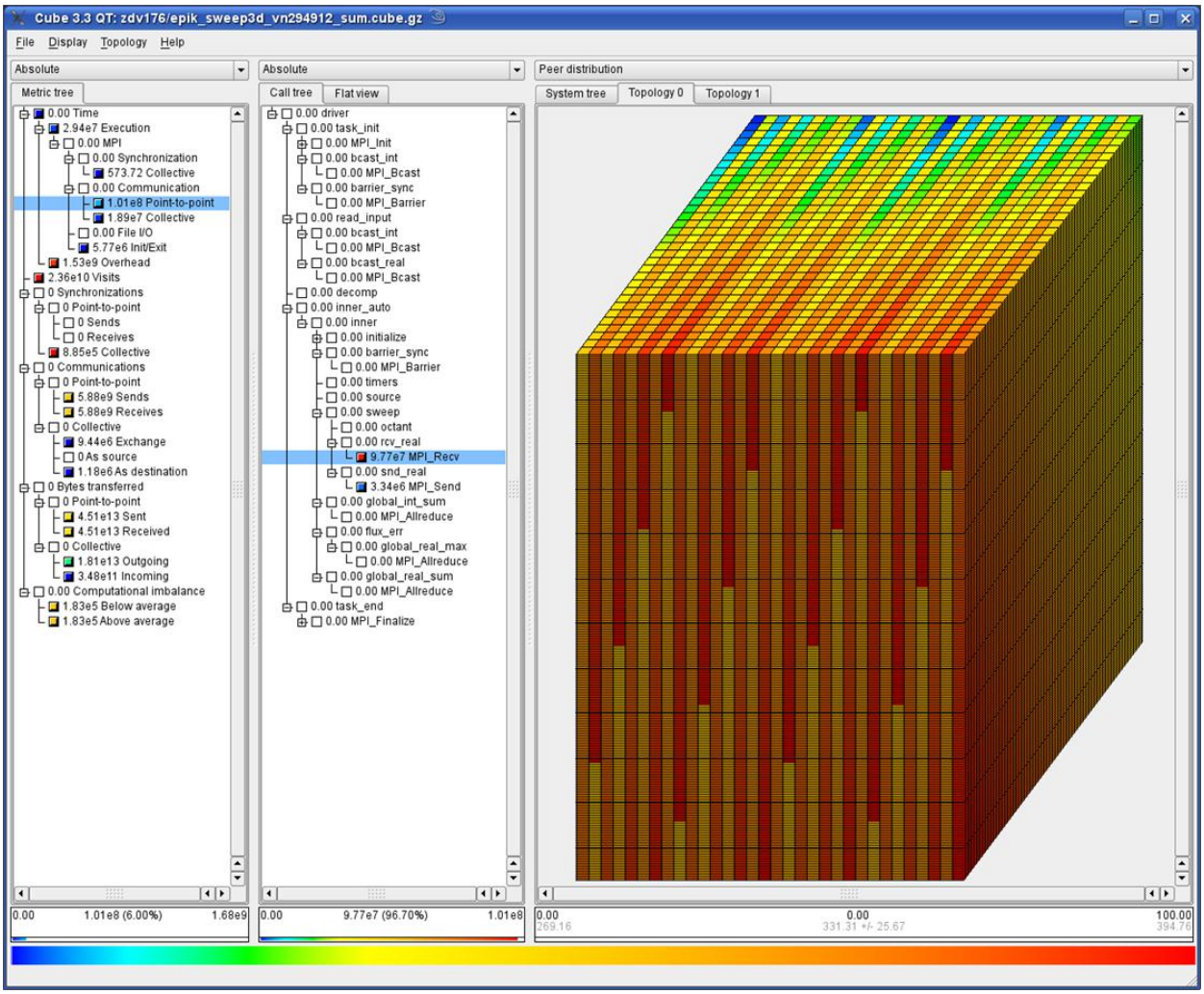

Figure 6. Scalasca result browser display CUBE. The left panel shows the hierarchy of measured metrics. The middle panel shows the distribution of the selected metric over the call tree of the program. Finally, the right panel shows the distribution of the selected metric at the selected call path over the machine topology.

The user of Scalasca can choose between two different analysis modes: (i) performance overview on the call-path level via runtime summarization (aka profiling) and (ii) in-depth study of application behaviour via event tracing. A distinctive feature of Scalasca is its ability to identify wait states that occur, for example, as a result of load imbalance – even at very large scales. Analysis results of Scalasca can be investigated with the result browser CUBE (see Figure 6).

The software is installed at numerous sites in several countries and has been successfully used to optimize academic and industrial simulation codes. Scalasca, which is jointly developed by JSC and GRS, is available for download under the New BSD open-source license at http://www.scalasca.org.

The currently available public version (Scalasca 1.4) is based on its own internal very scalable instrumentation and measurement system. Starting 2013, Scalasca 2.x is based on the new community-developed open-source measurement system Score-P (see Section 6).

3.2 Main achievements

Several enhancements have been made in 2011 and are already available in the last public release of Scalasca (Version 1.4.2 of July 11, 2012) and CUBE (Version 3.4.2 of July 11, 2012):

- Integration of Scalasca with the Paraver and Vampir timeline visualizers ("worst-instance tracking")
- Improved scalability of MPI communicator and group handling for Scalasca's native measurement system as well as for Score-P

The following enhancements are not part of a public release yet but are available to HOPSA project partners:

- Design and implementation of a prototypical generic tool launch feature for CUBE. It allows launching arbitrary commands from the context menu of metric and callpath nodes to display further information connected with metrics or callpaths. A simple cookie-based communication protocol between CUBE and the launched processes was also defined.
- The prototype of Scalasca's root-cause analysis was optimized and extended to support hybrid codes. The current version supports MPI, OpenMP and hybrid MPI/OpenMP measurements, facilitating the analysis of wait-state propagation from MPI to OpenMP synchronization points and vice versa. The prototype is currently being integrated into the release brach of Scalasca.
- A prototype of Score-P with functionality for the dynamic compression of time-series profiling data was created. The prototype supports hybrid applications (i.e., MPI combined with OpenMP).

4. ThreadSpotter (RW)

4.1 Basic description

The ThreadSpotter performance optimization technology [4] was originally developed in the startup Acumem AB – a spin-out from research at Uppsala University in Sweden. Acumem was acquired in 2010 and ThreadSpotter, as well as the original team, is now part of Rogue Wave Software AB. Since the start, the focus has been on performance debugging tools that explains to a programmer what actions need to be taken to achieve optimal performance. While an ordinary binary is running in a production environment, this new performance debugger collects sparse information about its execution behaviour into a ”fingerprint” file. Typically, only a couple of megabytes are needed to store the fingerprint data from several hours of real execution. It should be noted that the information collected in the fingerprint file is architecturally independent, i.e., it correctly represents the access locality of the application at an abstract level. Based on this information, the cache performance of any size cache, any size cache line and several replacement policies can be estimated off-line. Actually, the curve showing the miss-rate as a function of cache size for the entire application, as well as per-loop and per-instruction is generated at a fraction of a second off-line based on this data.

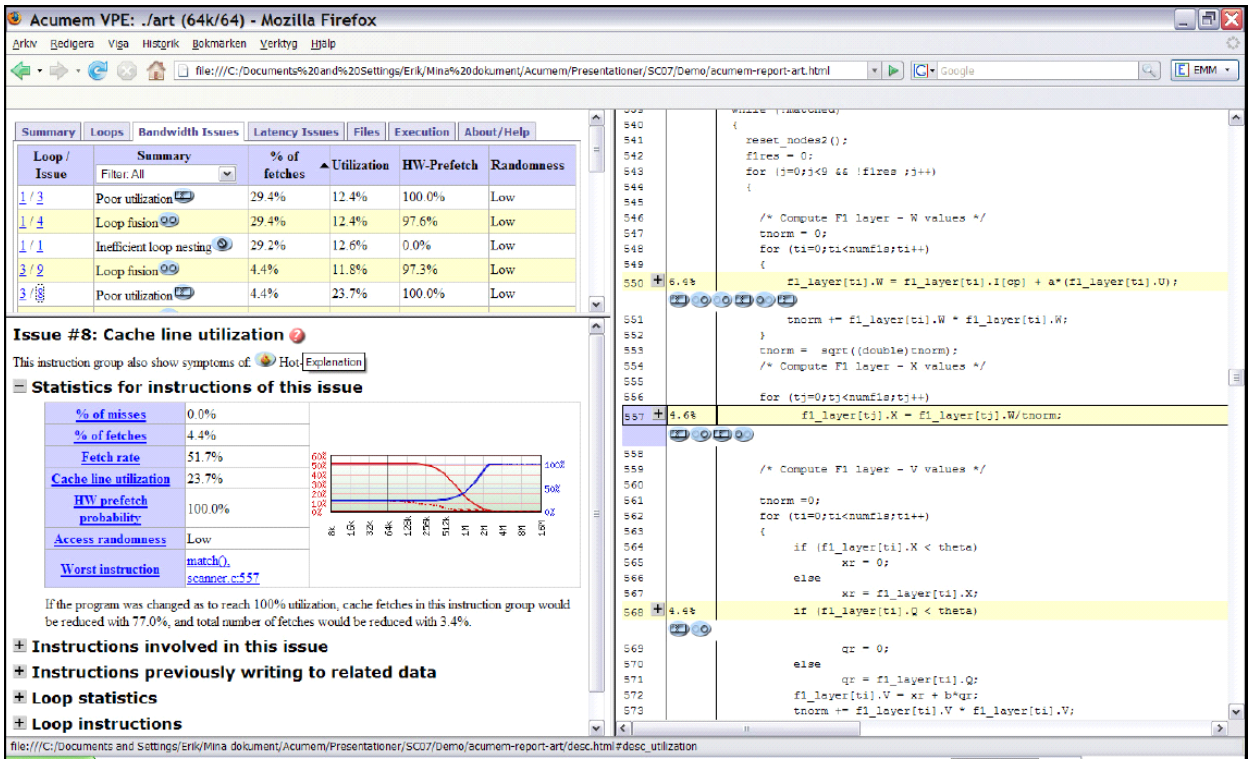

Figure 7. ThreadSpotter result display showing various detected performance issues (top left), the affected code regions (right), and further explanations and instructions on how to resolve the performance issues (bottom left).

While such curves could prove themselves useful for performance experts, the biggest strength of this technology goes far beyond that. ThreadSpotter’s analysis technology also detects performance bugs in the applications, i.e., certain access patterns that result in a sub-optimal performance. ThreadSpotter organizes such performance bugs into four issue groups: bandwidth issues, latency issues, thread interaction issues and cache pollution issues. For each issue group, the individual performance bugs are sorted in a worst-first order and presented in a table form together with an ample of statistics. Clicking on one such issue takes you to the source code where the performance

bug has been committed and opens up a window with more information guiding the programmer towards a more efficient alternative. This enables even non-experts to tune their code towards optimal performance.

4.2 Main achievements

The ThreadSpotter technology has been improved in several dimensions in this project.

In the task focusing on collection and analyse of profile-based performance data for a shared-memory process based on timer interrupts, all the goals are met. The goals include the design of a new entry point into ThreadSpotter based on this analysis and integrate with the existing issues-based (bandwidth, latency, inter-thread and pollution issues) and loop-based views. Within this framework the following steps have been implemented:

- A new profile-based sampling scheme has been implemented to synchronously sample information from all threads.
- The profile-based sampling is done in an interleaved fashion with the existing reuse-distance sampling.
- A time-based profiler GUI has been integrated in the ThreadSpotter GUI.
- This will get integrated in future releases of ThreadSpotter.

Also in the task of designing scalable method for collecting ThreadSpotter's performance fingerprint data from each of the MPI ranks running in a scalable system the goals have been met. This involves designing a filtering function that identifies MPI ranks with similar performance characteristics, based on the fingerprint and the new profile-based data. That way, a user will not have to wade through all performance data collected from 1000s of ranks and can concentrate on the unique behaviour. The following steps have been implemented:

- A scalable MPI launch mechanisms has been implemented
- Each MPI rank writes its own ”fingerprint file”
- ”Clustering” of fingerprint files based on k-mediods has been implemented to find MPI ranks with similar behaviour.
- The effectiveness of this has been demonstrated to find ”similarities” between ranks automatically for many applications.
- Example: Jacobi's execution results in five ”similarity groups”: Inner, top, bottom, left, right.
- This will get integrated in future releases of ThreadSpotter.

Finally, the goal of exploring the possibilities of integration between timeline-based visualizers and ThreadSpotter has been med. The following items have been achieved.

- We have come up with a seamless way to navigate between tools, e.g., between the Scalasca Cube and ThreadSpotter.
- This is based on the least common denominator: Call-stack-based views -- allowing the migration from one tool to the other while focusing on the same call-stack frame.
- A common representation of call stacks has been defined.

5. Vampir (TUD)

5.1 Basic description

Vampir (”Visualization and Analysis of MPI Resources”) is a very well-known event trace visualization software which is available since 1996 as a commercial product. It offers intuitive parallel event trace visualization with many displays showing different aspects of the parallel performance behaviour [5]. It provides interactive zooming and browsing to show either a broad overview or very small details.

The public version supports analysis of traces in OTF (Open Trace Format). These can be generated with the open-source VampirTrace instrumentation and run-time measurement package which supports not only MPI parallel programs but also OpenMP threads, POSIX threads, the IBM Cell architecture, GPGPU computing with CUDA or OpenCL, and combinations of them.

The HOPSA work is based on an upcoming new version based on traces in the new OTF2 format and the new Score-P run-time measurement package (see Section 6).

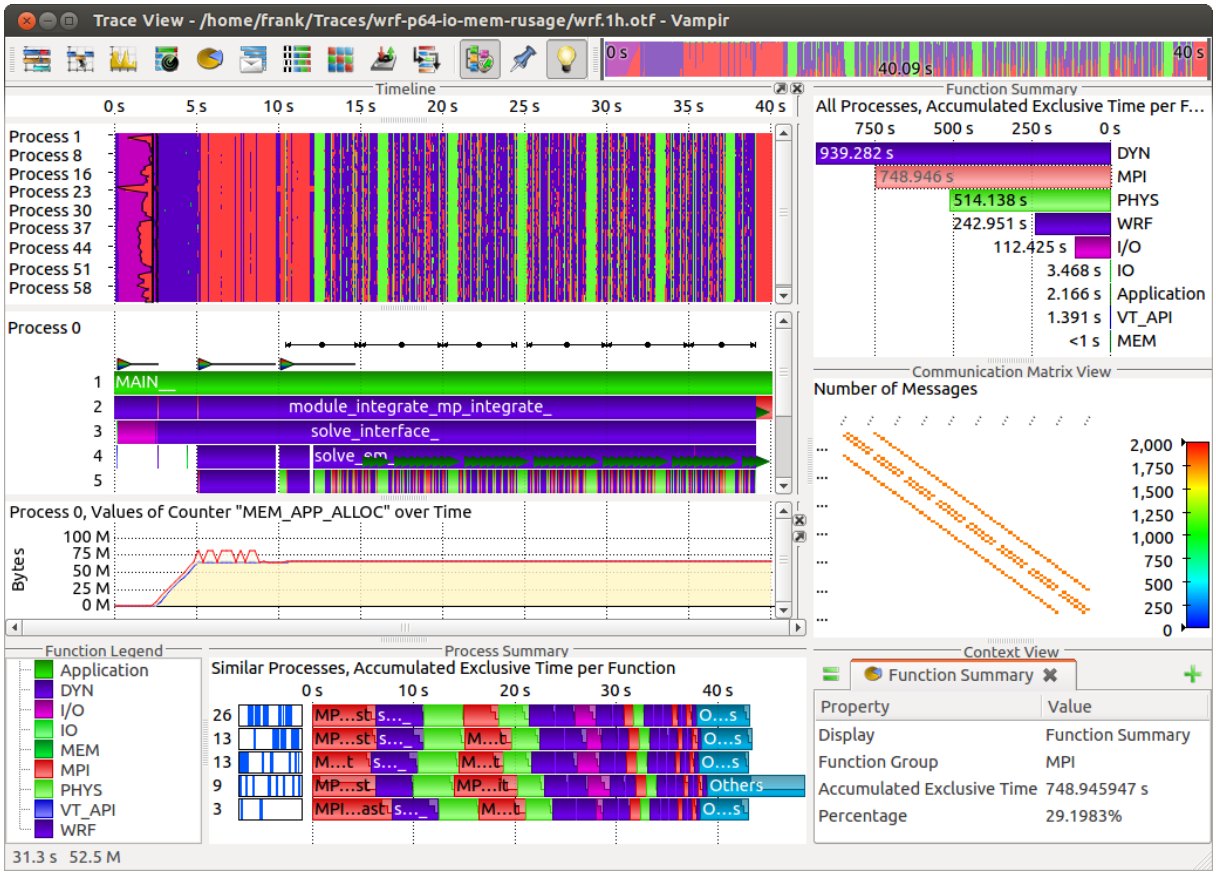

Figure 8. Vampir screen dump showing various displays: timeline views, function summary, communication matrix view and process summary.

All recent versions of Vampir support parallel trace data processing; furthermore a special analysis server allows to cope with very large traces: While the display component runs on the local desktop or laptop machine, the server component processes extensive event trace data sets remotely and in parallel on a part of an HPC system. By this means, Vampir is able to visualize traces with several hundred thousand processes/threads and terabytes in size while still providing an interactive working experience. Vampir is available for all major HPC platforms, including common Linux/Unix systems as well as Windows HPC Server. Today, Vampir is developed by ZIH, TU Dresden and is commercially

distributed by the university-owned company GWT TU Dresden GmbH. Detailed information about Vampir is available at http://www.vampir.eu. VampirTrace is distributed as open source under a BSD license at http://www.tu-dresden.de/zih/vampirtrace/.

5.2 Main achievements

The work of TUD as part of HOPSA focused around tracing of large scale, long running applications as well as the integration and analysis of system level performance data in application traces (see Section 6). Several enhancements during the project have been made and are already available in the latest release of Vampir 8.0 (November 2012):

- Vampir has been extended to visualize system background activities influencing all/several processes/threads in a node/partition/system. This new feature has been integrated into the "Counter Data Timeline" and "Performance Radar" and offers the possibility to visualize metrics belonging to a group of processes such as a node of a cluster system. The assignment of processes to a group is stored in the trace format accordingly.
- The implementation of the D-Bus remote interface of Vampir has been revised and extended and offers now a richer set of remote control capabilities. New features include remote control of display-specific properties such as the selection of the displayed performance data counter.

These enhancements are only available as a prototype to HOPSA project partners but will be part of the next Vampir release in June 2013:

- The latest developer version of Vampir introduces partial loading of large trace files in time and space dimensions and hence allows to visualize a specific segment of a trace without loading the complete trace. The time option allows users to select and load a relevant data section by means of a thumbnail view. This feature is only available if snapshots have been created for the trace. The space option offers the opportunity to select particular processes/threads to reduce the amount of data to be loaded. These two options can be combined. While the partial loading with Vampir works well for OTF traces, it is not completely enabled for OTF2 traces yet since the work on implementing the writing of snapshots for OTF2 traces is still in progress.

6. Score-P (GRS, JSC, TUD)

6.1 Basic description

The Score-P measurement infrastructure[1] is a highly scalable and easy-to-use tool suite for profiling, event trace recording, and online analysis of HPC applications. As a community open-source project, it forms the basic infrastructure for collaborative work on parallel performance tools. It was created in the German BMBF project SILC (2009 to 2011) and the US DOE project PRIMA (mid 2009 to mid 2012), and will be maintained and enhanced in a number of follow-up projects (e.g., BMBF LMAC, EU ITEA2 H4H, and also including EU FP7 HOPSA).

Score-P offers the user a maximum of convenience by supporting a number of analysis tools. Currently, it works with Periscope from the Technical University Munich, TAU from the University of Oregon, the HOPSA tools Scalasca and Vampir, and is open for other tools [6]. Score-P consists of several reusable components:

- The Score-P instrumentation and run-time measurement is the central component and incorporates all other components. It contains code instrumentation functionality using various methods and performs the run-time data collection in the parallel environment.
- OTF2 (Open Trace Format 2) is a highly scalable, memory efficient event trace data format plus support library. It will become the new standard trace format for Scalasca, Vampir, and TAU and is open for other tools.

  OTF2 is the common successor format for OTF (from Vampir) and the EPILOG trace format (from Scalasca). It preserves the essential features as well as most record types of both and introduces new features such as support for multiple read/write substrates, in-place time stamp manipulation, and on-the-fly token translation. In particular, it will avoid copying trace data during unification of parallel event streams.
- CUBE4 is a highly scalable, memory efficient, flexible profile format with support libraries, a set of tools, and a GUI. It will become the new standard profile format for Scalasca and Score-P and is open for other tools.

  CUBE4 is the successor profile format for the CUBE3 profile format from Scalasca. It preserves the CUBE3 data model and extends its internal mechanisms for saving the profile data. In particular, it is able to deal with large amounts of data, by dynamically loading and incrementally writing data. In contrast to CUBE3, CUBE4 is a hybrid format, saving an XML anchor file and set of binary files storing the profile data in a single binary archive.

  For backward compatibility, CUBE4 will provide reading support for the CUBE3 profile format (former Scalasca default).
- The Online Access Interface enables performance analysis tools to employ the Score-P monitoring infrastructure at runtime remotely over TCP/IP. It is currently used by Periscope. Highlights of the online performance analysis are more fine-grained measurement configuration, the support for multiple performance experiments within one run, and remote analysis with data acquisition over the network.
- OPARI2 is a source-to-source instrumentation tool for OpenMP and hybrid codes. It surrounds OpenMP directives and runtime library calls with calls to the POMP2 measurement interface.

[1] The Score-P measurement infrastructure is called “SILC measurement system” in the HOPSA proposal and DoW.

6.2 Main achievements

A new public release (Score-P Version 1.1 from November 03, 2012) is available at http://www.score-p.org as open source under a BSD license. Score-P enhancements as part of the HOPSA project made in 2012 were focused around tracing of large-scale, long-running applications as well as integration and analysis of system-level performance data in application traces:

- The Open Trace Format 2 has been extended with profile snapshots. The snapshots provide all status information required to start reading trace events from predetermined breakpoints without reading the preceding events. The additional data is generated by a post-processing tool and will be used by Vampir for partial loading of OTF2 traces.
- A long-term event-trace recording mode was introduced to the Score-P monitoring component. It allows to discard the preceding section of the event trace at certain rewind control points or phase markers. The runtime decision whether to keep or discard a section can depend on the presence or absence of certain behaviour patterns as well as on similarity or difference with other sections. This mode is part of the latest public release.
- Improved scalability of MPI communicator and group handling.

The following enhancements are available to HOPSA project partners for now and will later be included in a public release:

- The Score-P and OTF2 software infrastructure has been improved to enable the integration of system-level performance data.

7. Conclusions

This report summarized the status of the HOPSA integrated tool set for the instrumentation, measurement and analysis of parallel programs developed as part of work package 2 of the EU FP7 HOPSA project. It consists of the memory and thread analyzer ThreadSpotter (RW), the trace visualizer Paraver including its measurement system Extrae (BSC), the performance prediction tool Dimemas (BSC), the trace visualizer Vampir (TUD), the performance measurement and analysis tool Scalasca (GRS, JSC), and the instrumentation and measurement system Score-P (GRS, JSC, TUD).

The tools are already available as a combination of open-source offerings (Extrae, Paraver, Dimemas, Scalasca, CUBE, Score-P) and commercial products (Vampir, ThreadSpotter). At the end of the project (January 2013), a single unified installation package for all tools will be provided. This is detailed in Deliverable D3.4 ("UNITE Package").

Rogue Wave Software AB, Technische Universität Dresden, and the Jülich Supercomputing Centre are also project partners in the EU ITEA2 project H4H (Hybrid Programming For Heterogeneous Architectures) running in parallel with HOPSA. In H4H, the tools Scalasca, ThreadSpotter, Score-P, and Vampir are enhanced to better support the performance analysis of parallel programs for heterogeneous architectures based on the programming paradigms CUDA, OpenCL, and HMPP. As tool support for heterogeneous architectures was also a subject of the underlying EU FP7 call leading to the HOPSA project, there are synergies with the H4H project relevant to the HOPSA project. In 2011 and 2012, the following H4H tool enhancements were made available to HOPSA:

- Improved OpenMP instrumentation and measurement including support for OpenMP 3.0 tasks for the Scalasca measurement system and Score-P
- A new comparative analysis mode for the comparison of multiple trace files for Vampir
- Improvement of the function filtering mechanism based on filter rules that affects all charts for Vampir
- New customizable performance metrics to Vampir's existing Counter Data Timeline and Performance Radar
- Overview of performance metrics to Vampir’s Master Timline via an overlay functionality to get a relation between performance metrics and the corresponding functions at a glance
- Also the scalability of Vampir analysis has been further improved
- The Score-P instrumentation and measurement support has been extended with CUDA support based on the CUPTI callback API
- A prototype adaptor for HMPP from CAPS was developed which supports instrumentation and measurement of HMPP constructs on the host side
- A prototype adaptor for OmpSs from BSC was developed which supports instrumentation and measurement of OmpSs constructs

These enhancements are already included in the latest released versions of Scalasca and Vampir. Using the HOPSA tool infrastructure, the scientific output rate of a system will be increased in three ways: First, the enhanced tool suite will lead to better optimization results, expanding the potential of the codes to which they are applied. Second, integrating the tools into an automated diagnostic workflow will ensure that they are used both (i) more frequently and (ii) more effectively, further multiplying their benefit. Finally, the HOPSA holistic approach will lead to a more targeted optimization of the interactions between application and system.

8. Bibliography

[1] M. Geimer, F. Wolf, B. J. N. Wylie, E. Abraham, D. Becker, and B. Mohr. The Scalasca performance toolset architecture. *Concurrency and Computation: Practice and Experience*, 22(6):702–719, April 2010.

[2] M. Geimer, F. Wolf, B. J. N. Wylie, B. Mohr: A scalable tool architecture for diagnosing wait states in massively parallel applications. *Parallel Computing*, 35(7):375-388, July 2009.

[3] B. J. N. Wylie, M. Geimer, F. Wolf: Performance measurement and analysis of large-scale parallel applications on leadership computing systems. *Scientific Programming*, 16(2-3):167-181, 2008, Special Issue Large-Scale Programming Tools and Environments.

[4] E. Hagersten, M. Nilsson and M. Vesterlund, Improving Cache Utilization Using Acumem VPE, *TOOLS FOR HIGHPERFORMANCE COMPUTING 2008*, III, 115-135, DOI: 10.1007/978-3-540-68564-7_8

[5] A. Knüpfer, H. Brunst, J. Doleschal, M. Jurenz, M. Lieber, H. Mickler, M. Müller and W.E. Nagel, “The Vampir Performance Analysis Tool-Set”, *Tools for High Performance Computing*, pp 139-155, Springer Verlag, 2008.

[6] A. Knüpfer, C. Rössel, D. an Mey, S. Biersdorf, K. Diethelm, D. Eschweiler, M. Gerndt, D. Lorenz, A. D. Malony, W. E. Nagel, Y. Oleynik, P. Saviankou, D. Schmidl, S. Shende, R. Tschüter, M. Wagner, B. Wesarg, F. Wolf: *Score-P - A Joint Performance Measurement Run-Time Infrastructure for Periscope, Scalasca, TAU, and Vampir,* Proceedings of 5th Parallel Tools Workshop, 2011, (to appear).

[7] J. Labarta, Trace-Based Tools, *Performance Tuning of Scientific Applications*, Edited by D. H. Bailey, R. F. Lucas and S. W. Williams, pages 87–122, 2011.

[8] V. Pillet et al.: PARAVER: A Tool to Visualize and Analyze Parallel Code, in: 18th World OCCAM and Transputer User Group Technical Meeting, April 1995.

[9] A. Snavely, X. Gao, C. Lee, L. Carrington, N. Wolter, J. Labarta, J. Giménez, P. Jones, Performance Modeling of HPC Applications, Proceedings ParCo 2003.

[10] T-Platforms, Moscow, Russia. Clustrx HPC Software. http://www.t-platforms.com/products/software/clustrxproductfamily.html, last accessed September 2012.

[11] A.V. Adinets, P.A. Bryzgalov, Vad.V. Voevodin, S.A. Zhumatiy, D.A. Nikitenko. *About an approach to monitoring, analysis and visualization of jobs on cluster system* (In Russian). In: Numerical Methods and Programming, 2011, vol. 12, Pp. 90–93

[12] Bernd Mohr, Vladimir Voevodin, Judit Giménez, Erik Hagersten, Andreas Knüpfer, Dmitry A. Nikitenko, Mats Nilsson, Harald Servat, Aamer Shah, Frank Winkler, Felix Wolf, and Ilya Zhujov: *The HOPSA Workflow and Tools.* In: Proceedings of the 6th International Parallel Tools Workshop, Stuttgart, September 2012, Springer. To appear.

[13] H. Servat, X. Teruel, G. Llort, A. Duran, J. Giménez, X. Martorell, E. Ayguadé, J. Labarta: *On the Instrumentation of OpenMP and OmpSs Tasking Constructs* In: Proceedings of the PROPER2012 Workshop, Springer. To appear.

