Deliverable D2.1

Intermediate Tool Set

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

Due date of deliverable: February 1st, 2012

Actual submission date: February 24th, 2012

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

Name of lead contractor for this deliverable: BSC

Abstract: This report describes the 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

1. EXECUTIVE SUMMARY 4
1.1 THE BROADER CONTEXT: THE HOPSA PROJECT 4
1.2 WORK PACKAGE 2: HPC APPLICATION-LEVEL PERFORMANCE ANALYSIS 5
2. DIMEMAS, EXTRAE, PARAVER (BSC) 6
2.1 BASIC DESCRIPTION 6
2.2 MAIN ACHIEVEMENTS IN 2011 7
3. SCALASCA AND CUBE (GRS, JSC) 8
3.1 BASIC DESCRIPTION 8
3.2 MAIN ACHIEVEMENTS IN 2011 9
4. THREADSPOTTER (RW) 10
4.1 BASIC DESCRIPTION 10
4.2 MAIN ACHIEVEMENTS IN 2011 11
5. VAMPIR (TUD) 12
5.1 BASIC DESCRIPTION 12
5.2 MAIN ACHIEVEMENTS IN 2011 13
6. SCORE-P (GRS, JSC, TUD) 14
6.1 BASIC DESCRIPTION 14
6.2 MAIN ACHIEVEMENTS IN 2011 15
7. CONCLUSIONS 16
8. BIBLIOGRAPHY 17

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 an 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 will be 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.

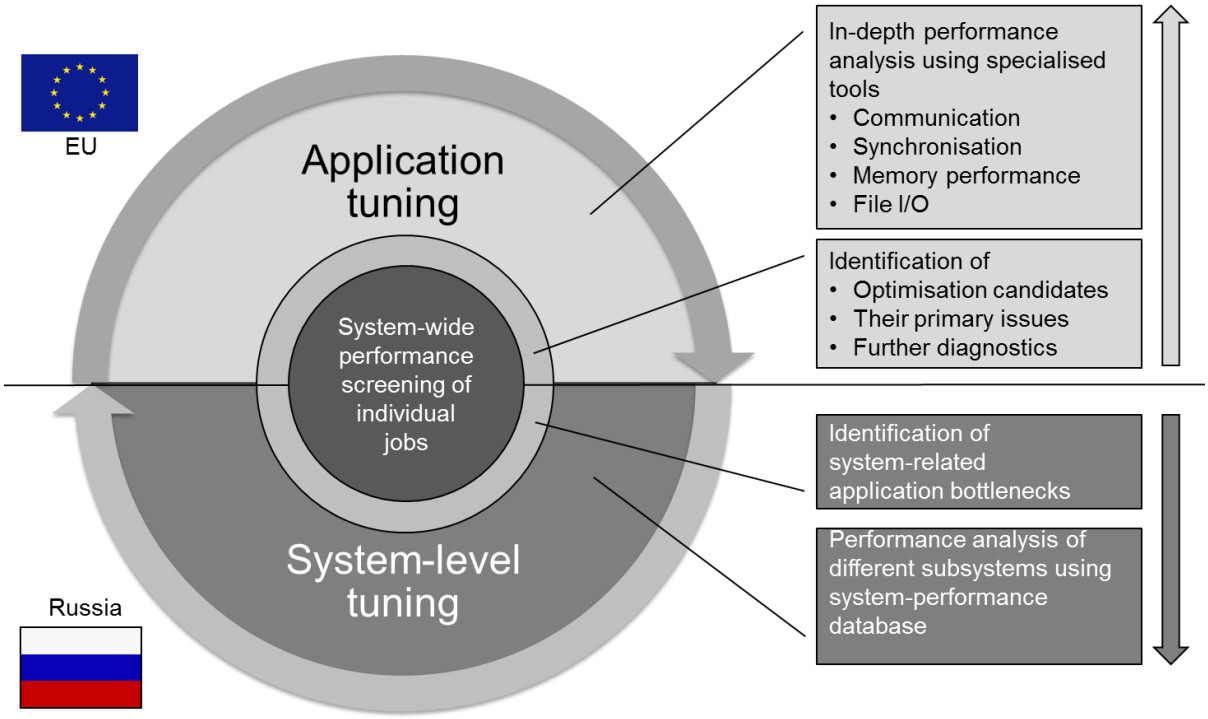

Figure 1: System-level tuning (bottom), application-level tuning (top), and system-wide performance screening (centre) use common interfaces for exchanging performance properties.

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 will be 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 is 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 is 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 following, the status of the HOPSA application-level tools is described and the work done as part of work package 2 in the first year of the HOPSA-EU project is briefly summarized. A more complete and detailed description of this work is provided by deliverable D1.1 (“Intermediate Progress Report”).

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.

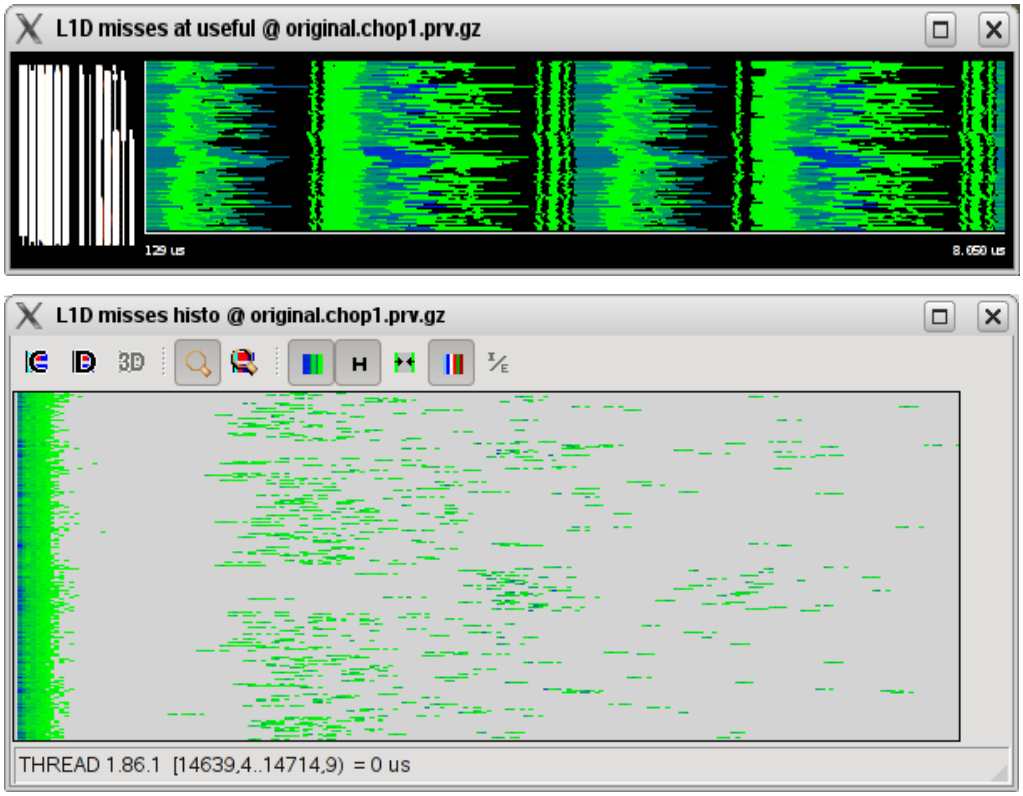

Figure 2. Paraver's two main types of views: timelines and histograms.

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.

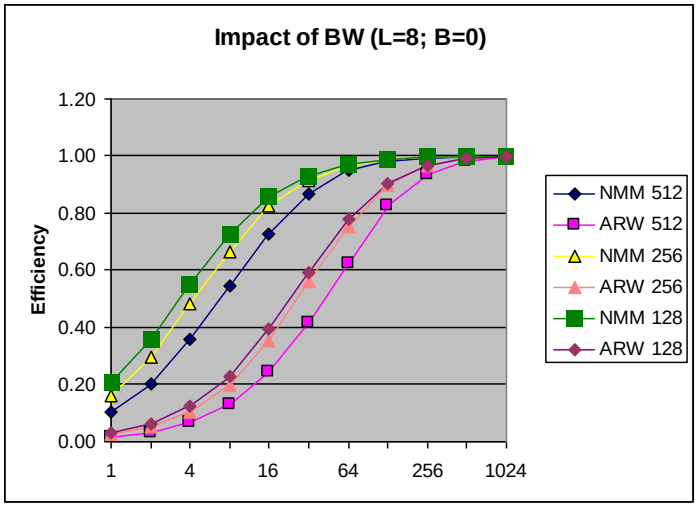

Figure 3. Dimemas parametric study example.

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 2011

Most of the BSC efforts during this first year of HOPSA concentrated on enhancing the scalability of the tools:

- Extrae has been extended to complement a very preliminary prototype that determines which information is dumped into the trace file based on an on-line automatic analysis of the collected data. The current prototype integrates both the clustering and the time analysis modules.
- Dimemas has been modified to allow simulations of tens of thousands of processes and threads requiring modifications on the tool structure, re-implementing the input trace module and Paraver trace generation module. The first prototype is ready and is currently being tested.
- Paraver has been enhanced by parallelizing its computation kernel using SMPSs. A first prototype deployed includes just a single parallel computation which is currently being tested.

Furthermore, to facilitate the usage of Paraver, a new mode for configuration files was implemented that abstracts from the Paraver details to offer an interface easier to use. We have also started to work on on-line tutorials to allow users to follow the tutorial steps within the Paraver graphical interface. Extrae has been extended to support CUDA instrumentation and the OpenMP runtime for Intel compilers.

Finally with respect to the tools integration, BSC has developed a prototype to integrate the sampled call-stack analysis with the result browser of Scalasca, CUBE. This prototype generates CUBE input from Paraver trace files, correlating the sources with the folded hardware counter metrics.

The functionality enhancements are already included in the latest releases available on the BSC website. The scalability improvements have not been publicly released yet but are available to HOPSA project partners.

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.

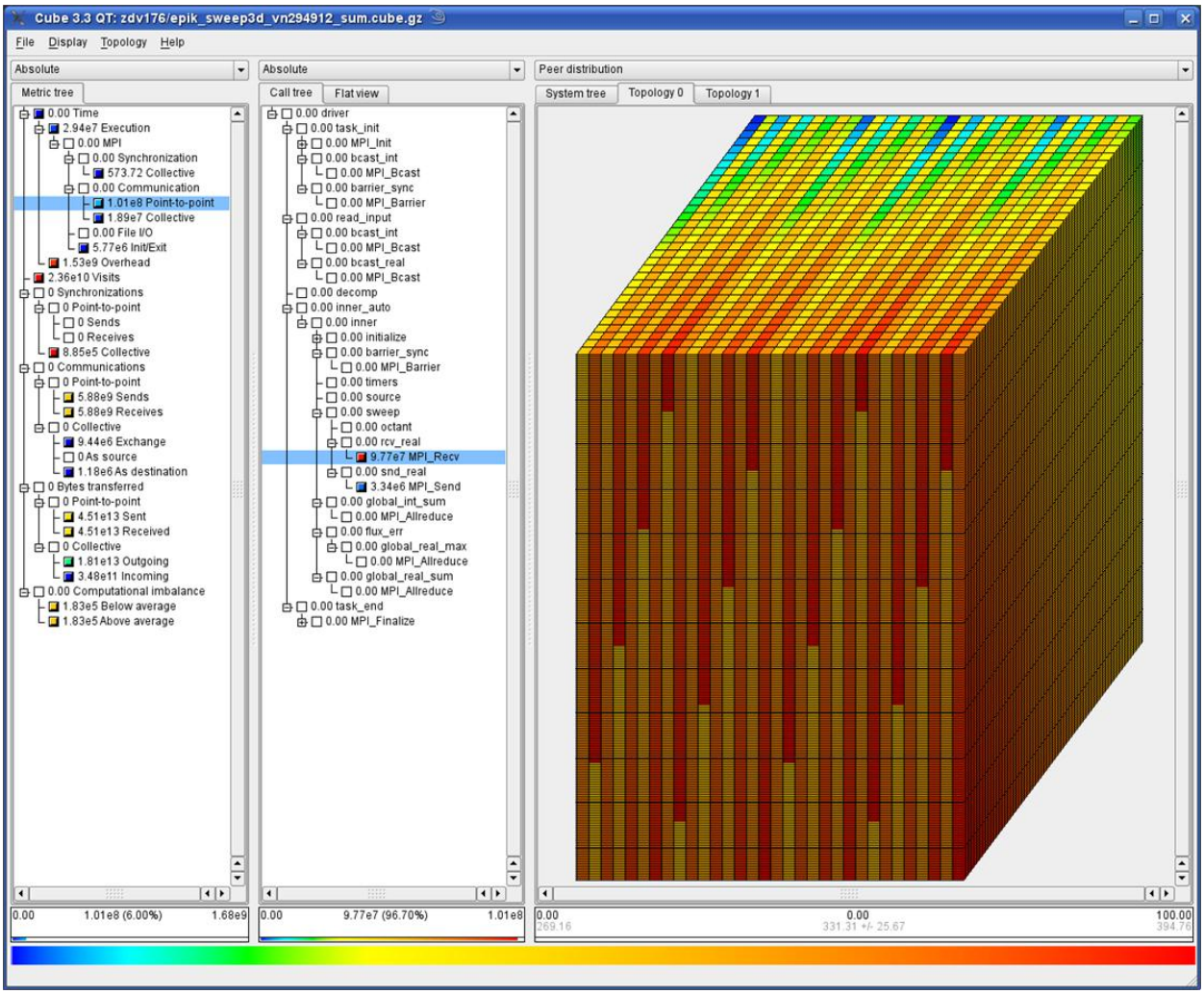

Figure 4. Scalasca result browser display CUBE. The left panel shows the hierarchy of measured metrics. The middle panel shows the distribution of the selected metric over the call tree of the program. Finally, the right panel shows the distribution of the selected metric at the selected call path over 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 4).

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. Future versions (Scalasca 2.x) will be based on the new community-developed open-source measurement system Score-P (see Section 6).

3.2 Main achievements in 2011

Several enhancements have been made in 2011 and are already available in the last public release (Version 1.4.1 of February 6, 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 prototype for the so-called root-cause analysis for Scalasca. It supports MPI, OpenMP and hybrid MPI/OpenMP measurements, facilitating the analysis of wait-state propagation from MPI to OpenMP synchronization points and vice versa.

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.

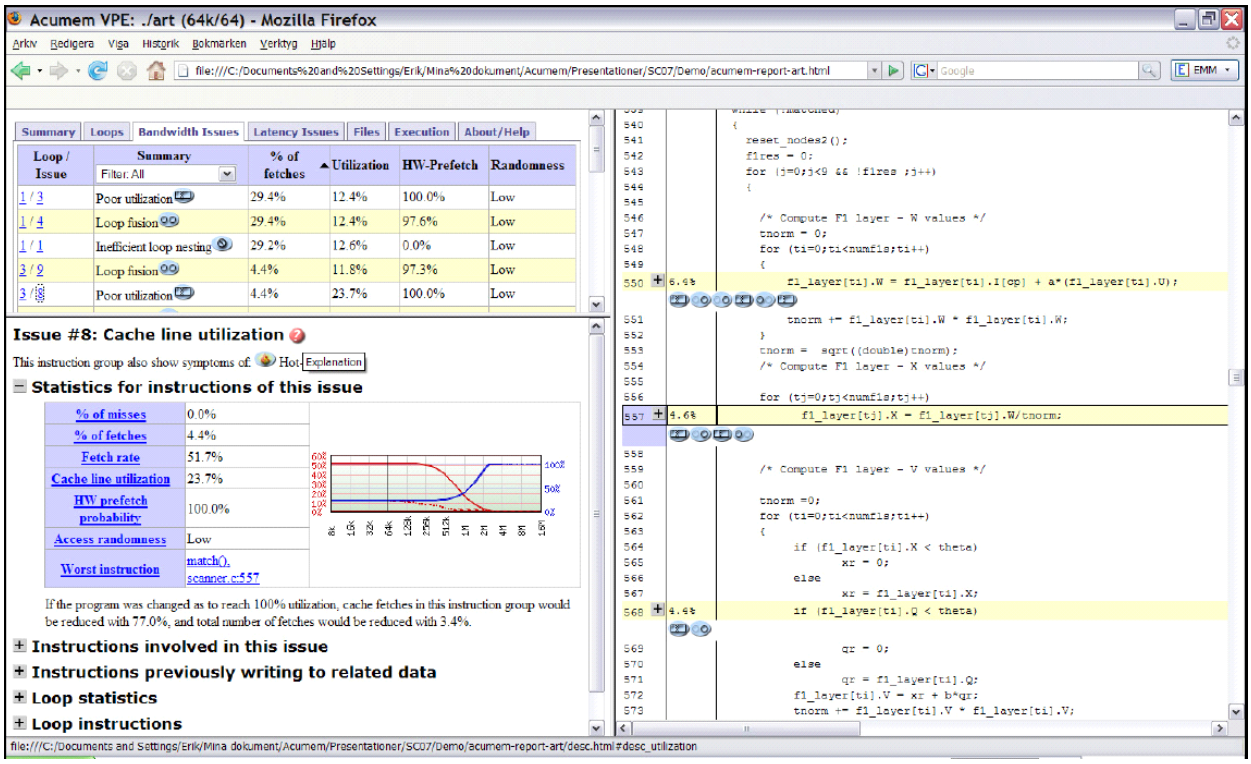

Figure 5. ThreadSpotter result display showing various detected performance issues (top left), the affected code regions (right), and further explanations and instructions on how to resolve the performance issues (bottom left).

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 in 2011

Several new enhancements have been explored and prototyped during 2011:

- Different time-sampling schemes have been evaluated and partly prototyped.
- Several partial stack implementation options have been evaluated and partly prototyped.
- A prototype of the ThreadSpotter tool integrating time sampling and partial stacks has been created.
- Analytics experts within the company have been engaged in exploring algorithms to determine performance similarities between MPI ranks.
- Design options for tool integrations within HOPSA-EU have been discussed and explored.

The new functionality developed in HOPSA is planned to be included in future releases of the ThreadSpotter product (http:://www.roguewave.com/products/threadspotter.aspx).

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).

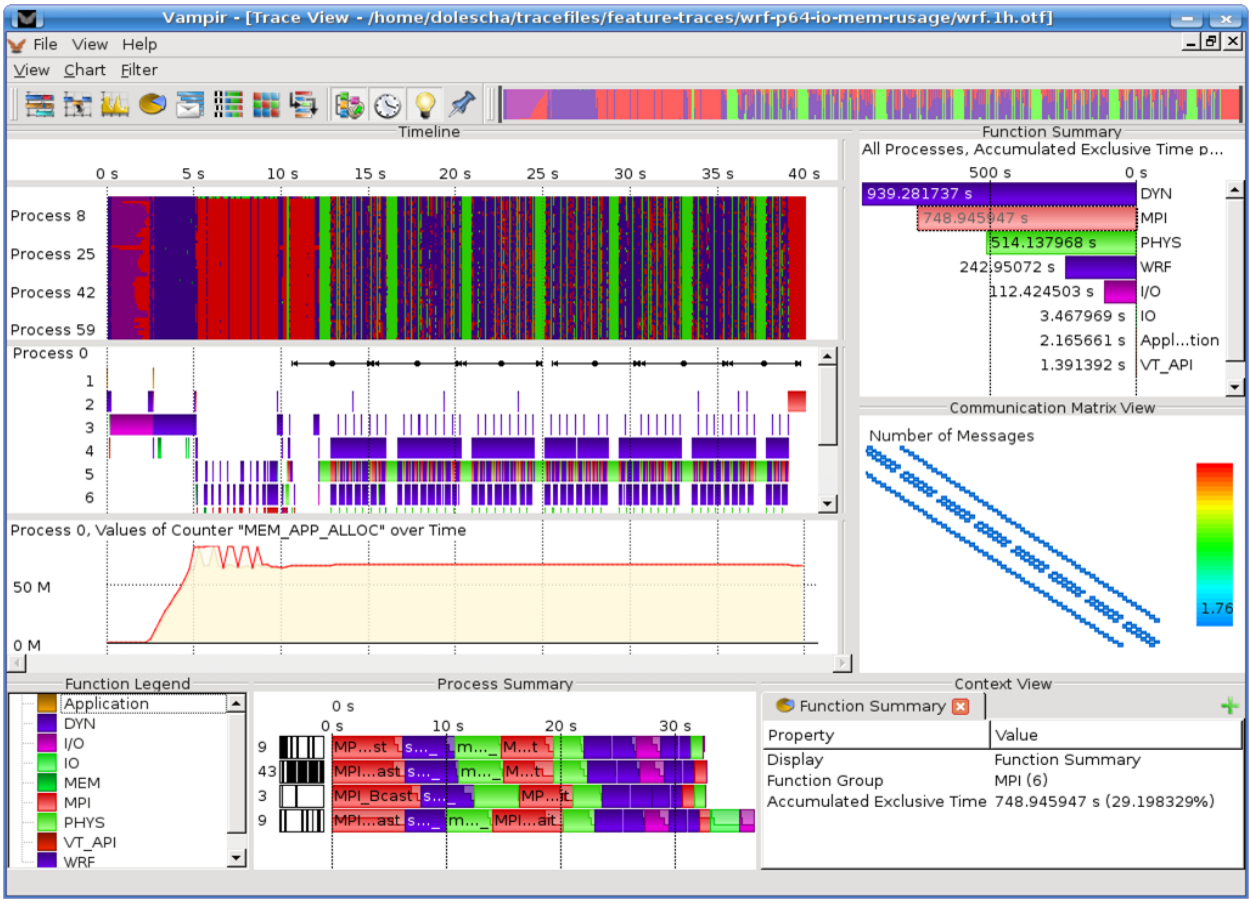

Figure 6. Vampir screen dump showing various displays: timeline view, function summary view, communication matrix view and process summary view.

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 in 2011

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). One enhancement of the Vampir tool was done:

- Partial loading in Vampir allows the handling of traces larger than the available memory. This feature is based on snapshot information and is currently limited to OTF1.

This enhancement is available as a prototype and will be released together with OTF2 support.

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 in 2011

A first public release (Score-P Version 1.0 from January 22, 2012) is available at http://www.score-p.org as open source under a BSD license which was the final deliverable of the SILC project.

Score-P enhancements as part of the HOPSA project made in 2011 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 statistics and snapshots. The profile statistics allows a coarse summary about the performance behaviour of predetermined time intervals. The snapshots provide all status information required to start reading trace events from this point without the preceding events. The additional data is generated by a post-processing tool.
- 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.
- The Score-P and OTF2 software infrastructure has been improved in preparation for the integration of system-level performance data.
- Improved scalability of MPI communicator and group handling.

The enhancements are available to HOPSA project partners and will later be included in a public release.

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.

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, the following H4H tool enhancements were made available to HOPSA:

- Improved OpenMP instrumentation for the Scalasca measurement system and Score-P
- A new comparative analysis mode for the comparison of multiple trace files for Vampir
- A new filtering mechanism to facilitate the analysis process for the user for Vampir
- Also the scalability of Vampir analysis has been further improved

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.

