Deliverable D1.2

Final Progress Report

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

Due date of deliverable: February 1st, 2013

Actual submission date: April 5th, 2012

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

Name of lead contractor for this deliverable: JUELICH

Abstract: This report summarizes the work done in the second year (February 1st, 2012 to January 31st, 2013) of the EU FP7 project HUPSA-EU.

Project co-funded by the European Commission within the Seventh Framework Programme (FP7/2007-2013)		
Dissemination Level		
PU	Public	
PP	Restricted to other programme participants (including the Commission Services)	X
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. PUBLISHABLE SUMMARY 5

2. COMMENTS ON PROJECT REVIEWER RECOMMENDATIONS FROM FIRST YEAR PROJECT REVIEW 7

3. CORE OF THE REPORT FOR THE PERIOD: PROJECT OBJECTIVES, WORK PROGRESS AND ACHIEVEMENTS, PROJECT MANAGEMENT 10

3.1 PROJECT OBJECTIVES FOR THE PERIOD 10
3.1.1 WP1: Project management 10
3.1.2 WP2: HPC application-level performance analysis 11
3.1.3 WP3: Integration of system and application performance analysis 11
3.2 WORK PROGRESS AND ACHIEVEMENTS DURING THE PERIOD 12
3.2.1 WP2: HPC application-level performance analysis 12
3.2.2 WP3: Integration of system and application performance analysis 26
3.3 PROJECT MANAGEMENT DURING THE PERIOD 30
3.3.1 WP1: Project management 30
3.3.2 Project Dissemination 30
General dissemination actions 30
Dissemination at international HPC conferences and workshops 31
HPC training events 33
Refereed publications 34
3.4 EXPLANATION OF THE USE OF THE RESOURCES 36

4. DELIVERABLES AND MILESTONES TABLES 39

5. ANNEX: USE OF RESOURCES TABLES FROM NEF 42

Declaration by the scientific representative of the project coordinator

I, as scientific representative of the coordinator of this project and in line with the obligations as stated in Article II.2.3 of the Grant Agreement declare that:

- The attached periodic report represents an accurate description of the work carried out in this project for this reporting period;
- The project (tick as appropriate) [1]:
  - X has fully achieved its objectives and technical goals for the period;
  - ☐ has achieved most of its objectives and technical goals for the period with relatively minor deviations.
  - ☐ has failed to achieve critical objectives and/or is not at all on schedule.
- The public website, if applicable
  - X is up to date
  - ☐ is not up to date
- To my best knowledge, the financial statements which are being submitted as part of this report are in line with the actual work carried out and are consistent with the report on the resources used for the project (section 3.4) and if applicable with the certificate on financial statement.
- All beneficiaries, in particular non-profit public bodies, secondary and higher education establishments, research organisations and SMEs, have declared to have verified their legal status. Any changes have been reported under section 3.2.3 (Project Management) in accordance with Article II.3.f of the Grant Agreement.

Name of scientific representative of the Coordinator: ....Dr.-Ing. Bernd Mohr.....

Date: April / 01 / 2013

For most of the projects, the signature of this declaration could be done directly via the IT reporting tool through an adapted IT mechanism.

[1] If either of these boxes below is ticked, the report should reflect these and any remedial actions taken.

Glossary

Abbreviation / acronym	Description
API	Application Programming Interface
BSC	Barcelona Supercomputing Center, Spain
CEPBA	European Center for Parallelism of Barcelona (UPC, BSC)
ClustrX	Cluster monitoring system (T-Platform)
CUBE	Performance report explorer for Scalasca (JSC) and Score-P (GRS, JSC, TUD)
CUDA	Compute Unified Device Architecture (Proprietary 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 (Proprietary 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
LAPTA	Database and analysis system for cluster monitoring data (MSU)
LWM2	Light Weight Monitoring Module (GRS) (Used for system-wide application performance screening)
MPI	Message Passing Interface (Programming Model for Distributed Memory Systems)
MSU	Moscow State University
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
PAPI	Performance Application Programming Interface (Library for portable access to hardware performance counter)
Paraver	Event trace analysis and visualization tool (BSC)
PMPI	Standard monitoring API for MPI
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
T-Platforms	Russian HPC cluster vendor
ThreadSpotter	Commercial memory and multi-threading performance analysis tool (RW)
TUD	Technische Universität Dresden, Germany
UNITE	UNiform Integrated Tool Environment (Unified documentation and installation procedures for HPC tools)
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. Publishable summary

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 in the HPC context developing an integrated diagnostic infrastructure. Using more powerful diagnostic tools, application developers and system administrators can 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 was 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 was 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 focused on the system aspect, the EU consortium focused on the application aspect.

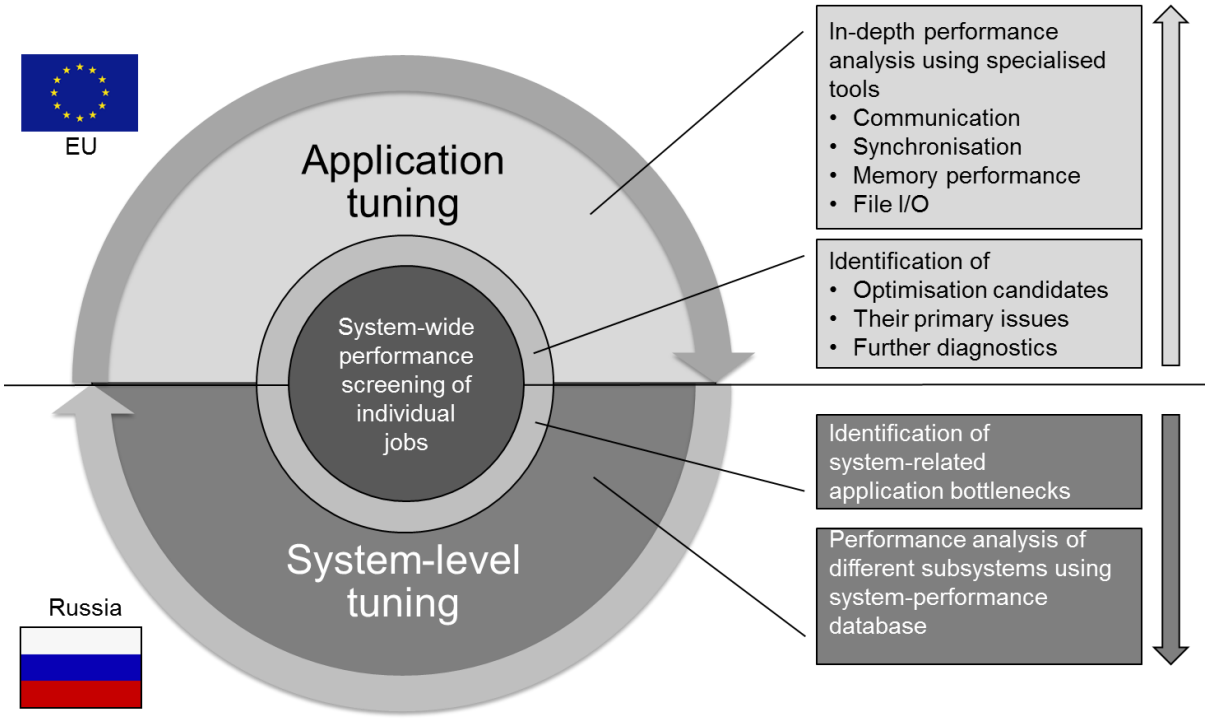


Figure 1: System-level tuning (bottom), application-level tuning (top), and system-wide performance screening (centre) use common interfaces for exchanging performance properties

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

For HPC application tuning, developers can choose from a variety of mature performance-analysis tools developed by our consortium. Within this project, the tools are 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 tools are available as a combination of open-source offerings (Extrae, Paraver, Dimemas, Scalasca, CUBE,

Score-P) and commercial products (Vampir, ThreadSpotter). Also, with the end of the project, a single unified installation package for all tools was provided.

The HOPSA project has fully achieved its objectives and technical goals for this period (February 1st, 2012 to January 31st, 2013). The main results for the second and final year are:

- ***WP1: Project management***

  The project was managed and coordinated as planned. Quarterly project meetings have been organized including our Russian project partners every other meeting. All deliverables were submitted and all project milestones have been reached. The project website has been greatly improved and enhanced (***http://www.hopsa-project.eu***) and has been linked with the website of our Russian partners (***http://hopsa.parallel.ru***). Project dissemination overall was extremely successful: The project was presented at seven events (including ISC and SC), often by multiple partners (1st year: four), 13 training events involving HOPSA tools have been organized (1st year: 12), and 11 project-related publications have been published (1st year: 2) with four completed additional publications waiting for publication.

- ***WP2: HPC application-level performance analysis***

  The individual tools of the HOPSA-EU tool set (Extrae, Paraver, Dimemas, Scalasca, CUBE, Score-P, Vampir, ThreadSpotter) have been considerably enhanced in their functionality and regarding scalability, enabling them to analyze parallel real-world applications executed with very large numbers (ten to hundred thousands) of processes and threads. Integration between the separate tool sets of the project partners also has been considerably improved. All enhancements are either already part of the latest public releases of the software packages, or at least are scheduled to be included in the next public. Milestone 2 (“Final Tool Set”) also has been reached. For a more detailed description of the HOPSA-EU toolset please see the deliverable D2.2 (“Final Tool Set”). The achievements in this work package have been tested, validated and documented in deliverable D2.3 (“Tool Validation Report”).

- ***WP3: Integration of system and application performance analysis***

  We integrated the software for system-level monitoring and analysis from our Russian HOPSA-RU partners with the HOPSA-EU tool set from WP2. After the definition of the interfaces between the system- and application-level components in the first year, which have been documented in deliverable D3.1 (“Requirements for the Interface between System-level and Application-level Performance Analysis”), work concentrated on the refinement and specification of the overall performance-analysis workflow which was documented in deliverable D3.2. The implementation of the central HOPSA component, the so-called light-weight monitoring module (LWM^2), which measures and extracts an application-level performance signature and delivers it to the system-level cluster monitoring layer, was finalized and documented in deliverable D3.3. In addition, both the Extrae/Paraver and Score-P/Vampir tool sets have been enhanced to allow the enrichment of performance analysis results and visualizations with system-level sensor data. Finally, an integrated package of all HOPSA-EU tools with consistent documentation and one unified installation mechanism was provided and documented in deliverable D3.4.

All deliverables were submitted and all milestones have been reached.

The HOPSA project delivered an innovative holistic and integrated tool suite for the optimization of HPC applications integrated with system-level monitoring. The tools are used by the HPC support teams of project partners in their daily work.

Taking an integrated approach for the first time in an HPC context worldwide, the involved 7 universities and research institutions considerably strengthened their scientific position as competence centres in HPC. Dresden University and Rogue Wave Software enriched their commercial software with unprecedented features and T-Platforms are to ship their HPC computer systems with the most advanced software offering, enabling all three of them to increase their respective market shares. Using the HOPSA tool infrastructure, the scientific output rate of a HPC cluster system can be increased in three ways: First, the enhanced tool suite leads to better optimization results, expanding the potential of the codes to which they are applied. Second, integrating the tools into an automated diagnostic workflow ensures that they are used both (i) more frequently and (ii) more effectively, further multiplying their benefit. European citizens will ultimately benefit from higher HPC application performance by for example more accurate climate simulations or a faster market release of medication. Finally, the HOPSA holistic approach leads to a more targeted optimization of the interactions between application and system. In addition, the project resulted in a much tighter collaboration of HPC researchers from the EU and Russia.

2. Comments on project reviewer recommendations from first year project review

In this section we report on how the HOPSA partners followed and implemented the reviewer recommendations from the first year project review report.

[1] While the project website documents the project and consortium as well as progress and results, it is somewhat drab and does not do the project justice. It has not been utilized in the most effective manner to document and more broadly disseminate new knowledge and ideas about performance analysis as well as system and application tuning in HPC environments; to point to materials prepared for training courses; and to attract more attention to from the broader HPC community. Given that the project generated plenty of compelling content and that most partners would benefit from recognition and publication of their achievements, there is an opportunity to rectify that through a facelift, some restructuring, better crosslinking and inclusion of additional as well as more effective presentation of the generated content.

A much more elaborate and graphically enhanced website has been set up. It features now separate subpages for Partners, Workflow, Tools, Documentation, Publications, Training, News, and Contact.

The Tools page provides a one-stop-place for finding documentation, specifications, and links to package downloads for all HOPSA tools. Great effort has been put into keeping the News page up-to-date with pointers to HOPSA related events. Finally, the training page provides material from the last HOPSA tools training.

[2] Provide more details regarding dissemination and awareness raising activities within the scientific community in Russia, within the scope of the overall dissemination efforts.

Besides organizing the APOS/HOPSA Workshop at JSCC, Moscow in May 2012 and the APOS/HOPSA training week at MSU RCC, Moscow in November 2012 which was also open to scientific users of the MSU computing facilities, our Russian partners published several papers on various Russian conferences (see ***http://hopsa.parallel.ru/?q=node/120*** for details).

In addition, another HOPSA performance tool training event is planned for the 2013 MSU HPC summer school.

[3] Explore the potential for publishing some papers aggregating and summarising the combined contributions of the EU and Russian partners.

A paper on the HOPSA workflow and the HOPSA performance tools written by the EU partners and MSU has been published [2]. A paper about new innovative system monitoring including the LWM2 and LAPTA integration written by GRS and MSU is in preparation.

[4] Link the project website to the Russian one, when rolled out, effectively linking the results of the coordinated efforts.

The Russian website has been linked on the HOPSA-EU website on the Overview (Main), Tools, Publications, and Contact subpages.

[5] Evaluate possibilities to participate in the HPC Advisory Council, a community effort support center for HPC end-users (http://www.hpcadvisorycouncil.com/), or other similar initiatives, to extend the breadth of dissemination efforts and consider establishing a special interest subgroup in the EU and Russia, focusing on performance analysis and combined application & system diagnosis and tuning.

HOPSA principal investigators from BSC, GRS, JSC and TUD are heavily involved as work package or task leaders in the European Exascale Software Initiative (see ***http://www.eesi-project.eu/***). BSC and JSC are also involved in the new European HPC technology platform ETP4HPC (***http://www.etp4hpc.eu/***). In both cases, the HOPSA partners bring in their rich experiences from performance measurement and analysis for parallel programs.

JSC representatives have been participating in recent meetings of the HPC Advisory Council. A deeper involvement is internally discussed, as much as constraints on time and manpower allow this involvement.

[6] Explore complementarities and potential synergies with the sister project APOS and jointly consider concrete opportunities for closer cooperation in the run up to the planned joint technical meeting in Moscow in May, 2012. Such cooperation might entail but is not limited to provision for inclusion of tools developed within APOS project (e.g. HMPP) in the unified HOPSA suite as well as provision for usage and validation of early versions of HOPSA tools by APOS partners.

Besides the common APOS/HOPSA Workshop in Moscow, in May 2012, we organized a APOS/HOPSA training week at MSU RCC, Moscow, in November 2012, where APOS application developers (among other Russian HPC users) were trained in the HOPSA workflow and the use of the HOPSA performance tools. During the workshop, the APOS members applied the HOPSA tools to their applications which are investigated as part of the APOS project. It was successful in the sense that several issues with the Score-P and Cube tools were uncovered, which actually delayed the delivery of the final tool set (deliverable D2.2 and milestone 2).

HOPSA also participated in a common workshop organized by APOS on Exploiting Heterogeneous HPC Platforms in conjunction with the HiPEAC 2013 conference, in Berlin, Germany in January 2013.

Finally, the UNITE tools package was installed on an internal EPCC cluster system which is dedicated for project work.

In the EU ITEA2 project H4H (see ***http://www.h4h-itea2.org/***) which shares the project partners JSC, RW and TUD with HOPSA, an integration of the performance tools with the HMPP compiler is implemented based on a newly developed performance monitoring interface within HMPP. H4H will finish in September 2013.

[7] In the context of the project's timelines, work plans during the forthcoming period should allow for sufficient time to conduct multiple rounds of testing and validation, feeding back into further development and integration of tools as well as refinements of the performance analysis workflow, bearing in mind the intended integration of application tuning with overall system diagnosis.

We tried our best to allow for a thorough testing and validation of the integrated tool set which was planned in last half year of the project. The testing and validation efforts were also strengthened by the extra testing by APOS partners during the APOS/HOPSA Training week. JSC and GRS added extra manpower to help in this regard. Especially MSU (as the Russian part of the project ended already by November 2012) but also the EU project partners worked beyond the project end on their own resources on further improving and testing the final project prototype.

[8] Given the need to process and present large amounts of complex interrelated data, that can be easily quantified, analyzed, and processed by a human user, explore opportunities to use novel and innovative graphical techniques for visualization, over and above those available in individual tools.

The tight and already packed project plan did not allow investigating novel and innovative performance data visualizations. However, the project partners plan to do research in the future on visualizing performance data on application topologies. JSC (in cooperation with LLNL) is organising a Dagstuhl perspective seminar on "Connecting Performance Analysis and Visualization to Advance Extreme Scale Computing" (see ***http://www.dagstuhl.de/en/program/calendar/semhp/?semnr=14022***) to which the other HOPSA tool developers got invited.

Finally, GRS, TUD and JSC (and others) submitted a proposal to an HPC call of the German BMBF in January 2013 which includes a work package on visualizing performance data on application topologies.

[9] Given the capability of the EU and Russian partners, particularly the academic ones, attempt to establish a stronger theoretical [mathematical] basis for various elements of performance analysis. In addition, consider possibilities of adopting some formal techniques to study systems that evolve at different temporal scales.

The tool community (especially tool groups working on what they call “automatic performance analysis” like the Scalasca team at JSC/GRS and the Periscope team at Technical University Munich) worked already for quite some time on formalizing so-called “performance properties”. See for example T. Fahringer, M. Gerndt, B. Mohr, F. Wolf, G. Riley, J.L. Träff: Knowledge Specification for Automatic Performance Analysis. APART Technical Report, Revised Version Technical Report IB-2001-08, 2001. Various conference papers have been published since then to update this specifications and formalisms to new programming models.

However, it is certainly true that performance analysis often results in capturing huge amounts of raw data but then just reports first order moments such as the average duration of routines. The need and possibility to increase the intelligence in the tool itself to process data in order to obtain information has been clearly identified. At BSC we have used the term Performance Analytics to refer to all those mathematical techniques from other areas of science and engineering that can be leveraged for the analysis of raw data coming from performance probes. We have used spectral analysis techniques, clustering algorithms, image tracking concepts, sequence alignment algorithms. The sampling+folding tool that has been integrated with CUBE is an example of combining several such mathematical techniques.

Also, during the project meeting in Moscow in May 2012, the theoretical mathematics behind ThreadSpotter and its corresponding publications in academia were presented by RW and discussed extensively [Berg SIGMETRICS 05; Berg ISPASS 06; Eklov SPASS 10]. Also, different mathematical models for clustering were discussed.

We are certainly committed in this direction and envisage a huge boom of Performance Analytics examples in the near future.

[10] Consider introducing some business metric to maximise the efficiency of HPC infrastructures and facilitate related business decisions. Within that context, consider ways to incentivize stakeholders (users and system administrators) to optimize applications and infrastructures based on the results of the integrated, end-to-end performance analysis.

It is probably true that current practice does not monitor the actual productivity/efficiency of the use of the resources on every single job run in the system. Detailed performance analysis is done on some test cases but then the efficiency in real production cases (which in many codes can be quite different) is not analyzed. HOPSA aimed at developing infrastructure that could close the gap between the fine grain performance analysis and the operational performance measurement. From our point of view an appropriate metric from the system point of view would be useful instructions/cycle (u_ipc). It would account the number of instructions excluding MPI or OpenMP runtimes executed per cycle.

The global productivity of the system should actually multiply this system efficiency by an algorithmic efficiency factor which could characterize the algorithm. Unfortunately this factor would probably be more difficult to agree on by system operators and allocation bodies. It would be field specific (i.e., the Molecular Dynamics area might be a normalization of nanoseconds/instruction simulated) and different weights might be assigned to each field depending on strategic management decisions.

[11] Last but not least, explore opportunities to apply some of the results of the integrated performance analysis to more effective design and optimized architectures of the future exaflop HPC infrastructures.

HOPSA project partners are already partner in the EU FP7 Exascale projects DEEP (BSC, GRS, JSC, see ***http://www.deep-project.eu***), Mont-Blanc (BSC, JSC, see ***http://www.montblanc-project.eu/***) and CRESTA (TUD, see ***http://cresta-project.eu/***) which all explore hardware and software architectures for future EU HPC Exascale efforts. BSC, GRS, and JSC are also part of the HPC pillar of the EU FET flagship project Human Brain (see ***http://www.humanbrainproject.eu/***) which investigates the use of Exascale computing to accelerate neurosciences. All listed projects feature work packages for performance measurement, analysis and visualization tools and their integration into the system software stack.

Follow-up project proposals to DEEP (DEEPER) and Mont-Blanc (Mont-Blanc2) have been submitted to a recent EU FP7 call in January 2013.

3. Core of the report for the period: Project objectives, work progress and achievements, project management

3.1 Project objectives for the period

Given the rather small number of partners (five) and the short duration (two years) of the project, we structured the work plan of the project into just three work packages:

- WP1: Project management
- WP2: HPC application-level performance analysis. This contains all research and technical development which only involves EU partners.
- WP3: Integration of system and application performance analysis. This contains all research and technical development which is done in cooperation with the partners of the coordinated Russian project.

The coordinated Russian project HOPSA-RU works on two topics:

- RU-Topic1: HPC system-level performance analysis
- RU-Topic2: Analysis of FPGA-based systems

Figure 2 shows the overall structure of the project and the major software packages which are developed and enhanced in the course of the project.

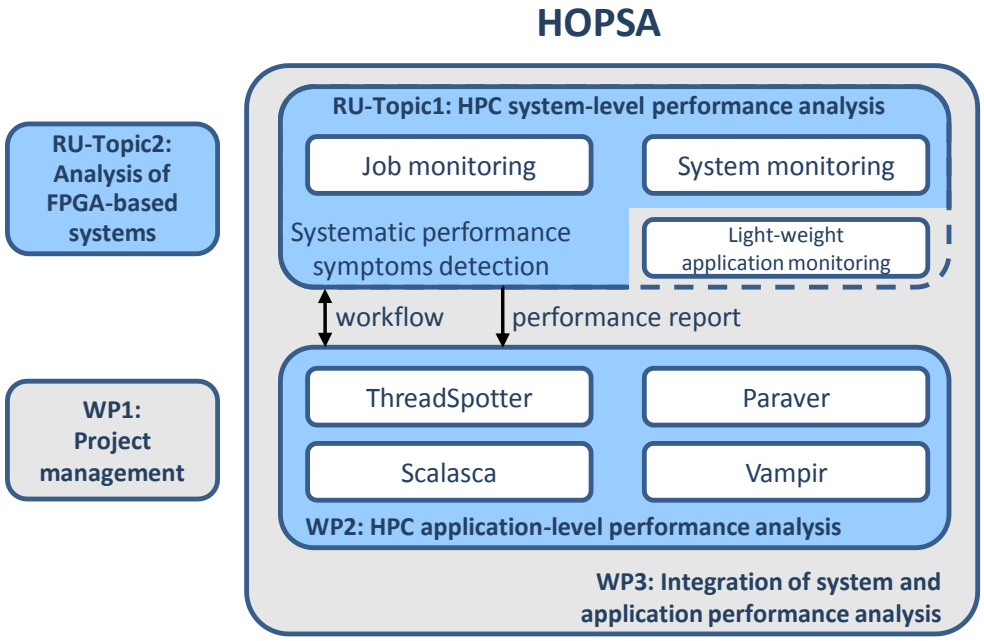


Figure 2: Overall structure of the HOPSA project.

3.1.1 WP1: Project management

The objectives of this work package were to perform the technical coordination of the project, to monitor the progress of the partners, to detect possible problems and to perform risk management, to ensure the quality management and assurance, and to synchronize the activities of the EU and Russian coordinated projects. It is decomposed into the tasks:

Task 1.1 Administrative and financial management

Task 1.2 Technical coordination

3.1.2 WP2: HPC application-level performance analysis

This work package contains all research and technical development which only involves EU partners. The overall objective of work package 2 was to enhance and extend the already existing individual performance measurement and analysis tools (ThreadSpotter, Paraver, Extrae, Dimemas, Scalasca, CUBE, Vampir, Score-P) 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. It is decomposed into the tasks:

- ***Task 2.1*** Enhancing functionality of the tools
- ***Task 2.2*** Enhancing scalability of the tools
- ***Task 2.3*** Tool integration
- ***Task 2.4*** Tool validation

The goal of the second year of the project was to complete outstanding tasks. The results in this work package are milestone 2 (“Final Tool Set”) represented by deliverable D2.2 scheduled for month 21, and the final tool validation report (D2.3) documenting and validation the progress made in work package 2 scheduled in month 23.

3.1.3 WP3: Integration of system and application performance analysis

This work package contains all research and technical development which is done in cooperation with the partners of the coordinated Russian project. Its objective was to combine and integrate the work done for the HPC system-level performance analysis (by RU-Topic1 in Russia) and for application-level performance analysis (by WP2 in the EU) into a coherent and holistic performance analysis environment. It provides low-overhead end-to-end performance analysis for all jobs running on a given system from their submission to their completion, identification of key performance issues and notification of the user and system performance database after job completion, and detailed scalable performance analysis for petascale applications based on well-accepted and robust performance measurement and analysis tools. It was decomposed into the tasks:

- ***Task 3.1*** Definition of the interface between system- and application-level performance analysis
- ***Task 3.2*** Definition of the overall performance analysis workflow
- ***Task 3.3*** Light-weight monitoring module
- ***Task 3.4*** Unified download, configuration, build and installation package

Again, the goal of the second year of the project was to complete outstanding tasks. The results of Task 3.2 are documented in deliverable D3.2 (“Workflow Report”) scheduled for month 15, of Task 3.3 in deliverable D3.3 (“Light-Weight Monitoring Module”) scheduled in month 18, and finally Task 3.4 in deliverable D3.4 (“UNITE Package”) scheduled in month 22.

3.2 Work progress and achievements during the period

The text inside this section contains text parts from the DoW (marked the same way as this paragraph) to allow an easy comparison between plan and work performed.

3.2.1 WP2: HPC application-level performance analysis

The individual tools of the HOPSA-EU tool set (Extrae, Paraver, Dimemas, Scalasca, CUBE, Score-P, Vampir, ThreadSpotter) have been considerably enhanced in their functionality and regarding scalability, enabling them to analyze parallel real-world applications executed with very large numbers (ten to hundred thousands) of processes and threads. Integration between the separate tool sets of the project partners also has been considerably improved. All enhancements are either already part of the latest public releases of the software packages, or at least are scheduled to be included in the next public. Milestone 2 (“Final Tool Set”) also has been reached. For a more detailed description of the HOPSA-EU toolset please see the deliverable D2.2 (“Final Tool Set”). The achievements in this work package have been tested, validated and documented in deliverable D2.3 (“Tool Validation Report”).

In the following the results achieved in the different tasks of this work package are described in detail.

Task 2.1 Enhancing functionality of the tools

DoW Subtask 2.1.A
To increase the depth of the analysis offered by Scalasca, we will add new functionality to locate the root cause of wait states. The existing prototypical implementation of the delay analysis [4], which received the Best Paper Award of the ICPP Conference 2010, will be consolidated and further optimized with respect to runtime and memory consumption. In addition, the approach, which was originally developed for MPI only, will be extended to hybrid applications that use MPI and OpenMP in combination. (GRS, JSC)

JSC and GRS continued implementation work and evaluation of the root-cause analysis in Scalasca. So far, Scalasca identified where wait-states in parallel programming constructs (MPI, OpenMP) occur. In contrast, the root-cause analysis locates the imbalances that lead to wait-states later on, and determines their cost in terms of the overall waiting time they cause. By incorporating long-distance effects through wait-state propagation in the model, we can effectively trace the causal chain of wait-state creation from the point where wait states occur back to their root causes.

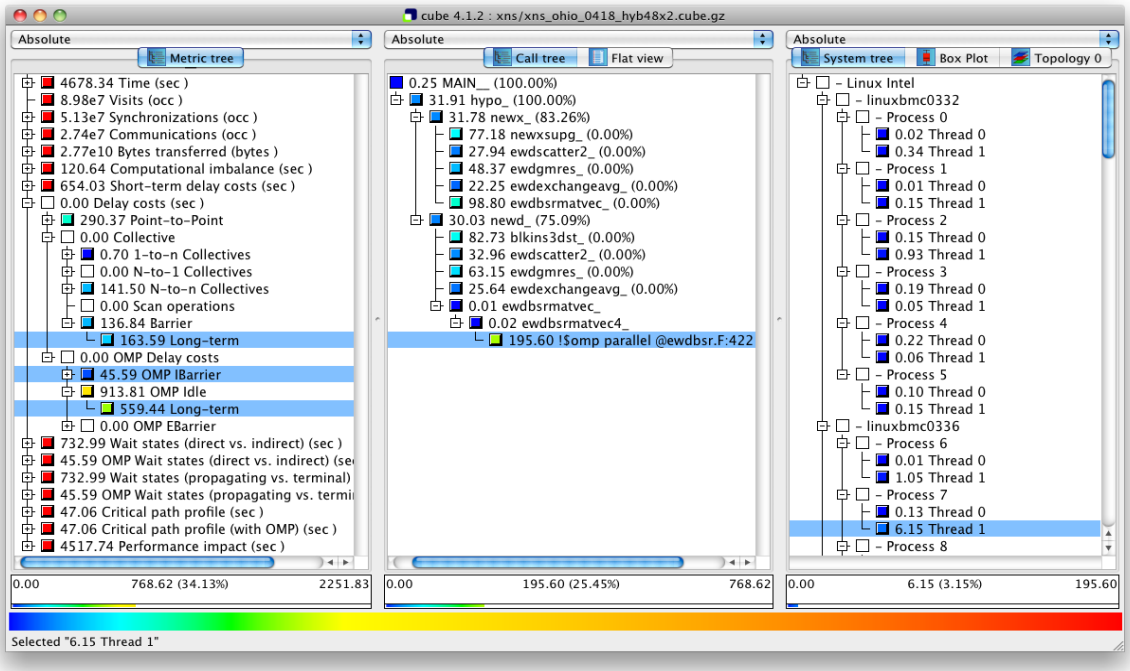


Figure 3: Results of a Scalasca root-cause analysis.

The consolidated implementation of the root-cause analysis for MPI point-to-point and collective communication was completed in the first period. This version detects root causes of wait states for "Late Sender" and "Late Receiver" type wait states in MPI point-to-point communication, as well as the root causes of wait states in collective broadcast, scatter/gather, reduction, barrier, and all-to-all communication. In contrast to the first prototype that was the basis for the ICPP publication, the refined version included a more scalable implementation for the detection of delays that cause wait states in MPI collective communication. A study with the Sweep3D benchmark demonstrated exceptional scalability of our approach: the extended Scalasca completed the root-cause analysis on traces with up to 262,144 MPI processes in less than two minutes.

Next, we implemented a prototype that also covers OpenMP and hybrid OpenMP/MPI programs. With respect to OpenMP, this prototype so far only supports a simple fork-join model with a fixed number of threads. However, an especially interesting use case for the root-cause analysis is hybrid OpenMP/MPI programs, where wait-states may spread across synchronization points belonging to different parallelization paradigms. Such inter-linked causal chains are especially difficult to uncover without tool support. Capturing the wait-state propagation between OpenMP and MPI constructs, which is required to model these multi-paradigm interactions, also emerged as a particular implementation challenge, as it required significant extensions to the underlying abstractions and to the existing implementation of the MPI root-cause analysis. The hybrid version of the root-cause analysis now specifically supports the analysis of wait-state root causes across paradigm borders for the combination of MPI and OpenMP. It can, for example, determine the total costs of an imbalance that leads to waiting time in an OpenMP barrier, which in turn delays a subsequent MPI communication and leads to more waiting time in MPI on a remote process. Moreover, we added the ability to characterize delays that either directly or through wait-state propagation cause OpenMP slave threads to sit idle in between OpenMP parallel regions.

The screenshot in Figure 3 shows a Scalasca root-cause analysis report for a hybrid OpenMP/MPI application with 48 MPI processes and 2 OpenMP threads on each process. The selected entries highlight delay costs of a computational imbalance inside an OpenMP parallel loop. This imbalance causes wait states at the implicit OpenMP barrier at the end of the loop. As indicated by the non-zero long-term costs in MPI barriers and OpenMP idle time, propagation of the wait states in the OpenMP barrier causes further waits in subsequent MPI barriers, which in turn increases the time that non-master threads on remote processes where the MPI barrier wait states occur sit idle.
Finally, we evaluated our implementation with benchmark and real-world case studies. Our experiences in applying the root-cause analysis to the CESM sea ice model can be found in the tool validation report in deliverable D2.3.

In the same way, Dimemas simulations can be used to perform the root cause analysis as the tool allows to analyse different what-if scenarios. The tool will be extended to give details on the selected scenario to an external tool (e.g.,Scalasca) (BSC)

The objective of this task was to use Dimemas as an external tool from the Scalasca analysis to implement the root cause analysis based on the Dimemas what-if scenarios. As the functionality implemented within Scalasca satisfied the needs of the tool, this task was discarded. The efforts were dedicated to other tasks that were considered more relevant like the integration of the CUBE visualizer in the folding analysis that was not initially planned. For more details on this work please see Subtask 2.3.E and deliverable D2.3.

In addition, in the first year, BSC worked on

- Mechanisms to facilitate new users getting familiar with the Paraver tool. A new mode for the configuration files that abstracts from the Paraver model and parameters and offers an easier-to-use interface was implemented. In addition, an on-line tutorial mode for Paraver has been added facilitating users to follow tutorial steps within the Paraver graphical interface. References to trace files and configurations are links that can be clicked to load the corresponding file.
- Extending its instrumentation and measurement system Extrae to support CUDA instrumentation as well as the new OpenMP runtime for the Intel compiler.
- Integrate the support for the StarSs applications simulation into Dimemas, including a new scheduler and the internal communications and synchronizations from the StarSs runtime.

Dow Subtask 2.1.B
Introduce a ”system background view” for Vampir. Besides the usual process-related events it will provide information on background activities which cannot be mapped to a single process/thread, but which are influencing all/several processes/threads in a node/partition/system. Examples are the network throughput or the storage system read/write rates. It will present performance counter values in fixed or dynamic intervals or newly defined events to the user. This requires extensions to the measurement system, the event trace format, and the visualisation displays. (TUD)

Vampir was improved to select, display, and associate performance information in the scope of system nodes rather than individual processes. This new feature called “System Background View” has been integrated into Vampir’s "Counter Data Timeline" and "Performance Radar". It has been successfully applied on the Russian system “GraphIT” by visualizing HOPSA node level metrics. For more details please see deliverable D2.2 and D2.3.

DoW Subtask 2.1.C
Collect and analyse profile-based performance data for a shared-memory process, such as an MPI strand, based on timer interrupts. Design a new entry into ThreadSpotter based on this analysis and integrate with the existing issues-based (bandwidth, latency, inter-thread and pollution issues) and loop-based views. (RW)

Rogue Wave has developed a feature which extends their sampling technology into capturing data in the time domain. The time domain will provide important data to qualify the access-oriented and cache-centric statistics previously produced by ThreadSpotter. Several alternatives to gather the new data have been weighed against each other in terms of performance, fairness, intrusiveness, and richness of information. Another goal was to simplify the implementation of a multiplexing mode where the old and the new sampling modes are interleaved. Special consideration has been given to the complex task of merging partial call stack information. The new sampling model has different properties than the existing sampling and they may both produce call stack data incomplete in their own respects. This part of the research has developed new methods for merging such fragments and distributing collected data from each domain statistically over the reconstructed call stack space.

This subtask was partially completed already in the first period. For more details see deliverable D1.1 and D2.1, D2.2. Final adjustments to the feature have been made in this last period:

- In addition to interleaved sampling of profiling and cache usage data, ThreadSpotter also supports capturing data with different modalities (time profile vs. cache usage) at different times (as in separate executions) and then reliably combining the data afterwards. This will improve fidelity for short enough executions where the interleaving does not produce a good coverage. ThreadSpotter can also make use of profile data alone.

- Additional improvements to the report viewer to allow the user to view execution time with other metrics.

Task 2.2 Enhancing scalability of the tools

The tools Paraver, Scalasca and Vampir and their corresponding measurement systems have already demonstrated extreme scalability in the analysis of parallel programs. The largest measurement and analysis ever accomplished by Scalasca was an experiment with the NAS BT-MZ benchmark on 1,048,576 cores on the Jugene IBM BlueGene/Q system at Jülich Supercomputing Centre. To our knowledge, this is the first successful performance measurement and analysis experiment world-wide with more than one million threads. The Vampir team recently visualized a 200,448 core measurement of the S3D code on the Jaguar Cray XT system at Oak Ridge National Laboratory using a VampirServer setup with 20,000 processes. The Paraver team also regularly uses Paraver to analyze measurement of codes running on a few 10,000 cores. Please see deliverable 2.3 (“Tool Validation Report”) for more details for these experiments. While this shows that the tools are able to handle very scalable systems already, a lot of resources and experience is required to perform these experiments. In the HOPSA project, the project partners work on enhancing the scalability of various components which currently are limiting the ability to use the tools on large cores counts more easily.

DoW Subtask 2.2.A

To improve Scalasca's scalability in terms of the number of cores and to reduce its runtime overhead, we will develop a more space-efficient distributed scheme to record MPI communicators. Avoiding rank translation at runtime will be a key requirement. After successfully testing a reference implementation for the native Scalascameasurement system, the new scheme will be transferred to the SILC measurement system so that it can also benefit Vampir. (GRS, JSC)

This work was completed in the first year of the project. To improve Scalasca's scalability in terms of the number of cores and to reduce its runtime overhead, a more space-efficient distributed scheme to record MPI communicators was designed and integrated by GRS and JSC into the Scalasca measurement system. Also, the Scalasca trace analyzer was modified to work with the enhanced measurement system. The new scheme was also integrated into the Score-P measurement system. A particular challenge of this task arose from the numerous dependencies affected by the changes, which required many modules of Scalasca to be modified.

DoW Subtask 2.2B

To improve Scalasca's scalability along the time axis, we will integrate a new method for the semantic runtime compression of time-series profiles [31], which so far exists only as an offline prototype. Finally, the new approach, which so far only supports MPI, will be extended towards hybrid MPI/OpenMP codes. Again, after successfully testing a reference implementation for the native Scalasca measurement system, the new scheme will be transferred to the SILC measurement system. Together, these changes will enable the detailed analysis of time-dependent performance behavior even for long-running experiments. (GRS, JSC)

The goal of time-series profiling is to investigate the variation of performance over time. As long as the application executes different code regions for every time period, the profile shows distinct statistics for each time period. However, many applications contain a main loop in which the application spends most of the time. Usually, profiling tool designs assume that the iterations of an iterative application behave basically the same, and thus, all visits are well represented by the resulting statistics. However, this assumption is not always true. Using the SPEC MPI benchmarks and the coulomb solver PEPC as examples, Szebebyi [7] has shown that some applications change their behavior for different iterations.

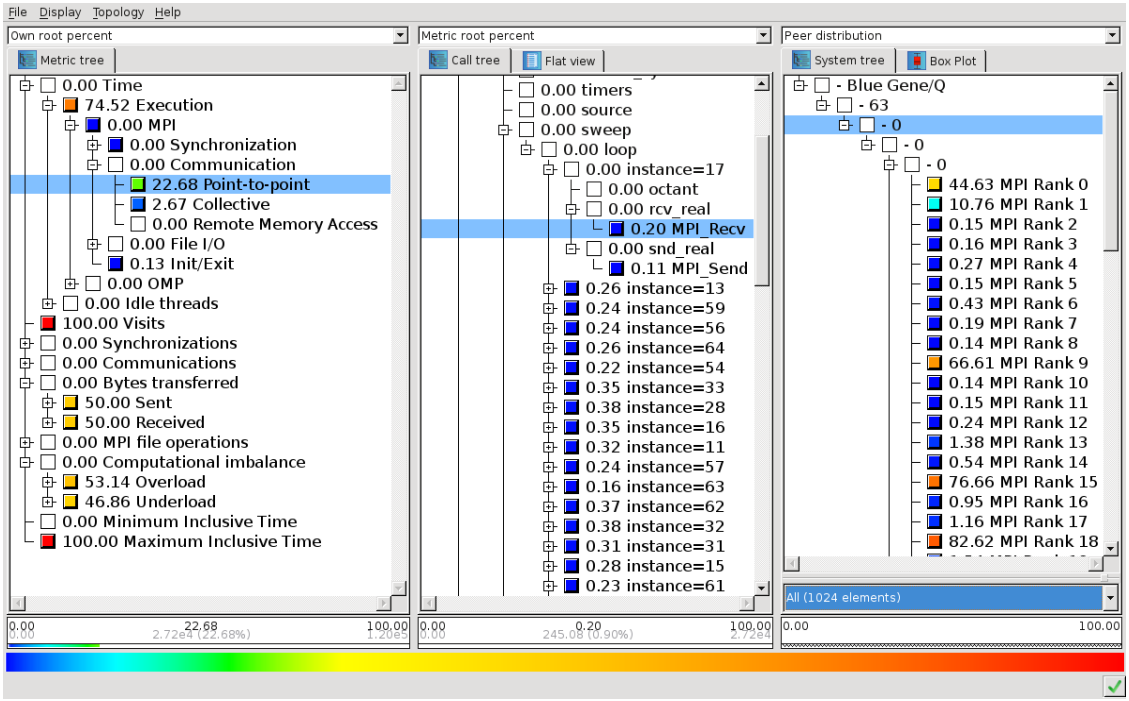


Figure 4: Results of a Score-P time-series profile measurement of the ASC benchmark sweep3d on the JSC BlueGene/Q machine. The call path panel in the middle show that data is available for analysis for various cluster instances of the "loop" inside the "sweep" routine.

In order to support the analysis of time-dependent behavior of iterative applications, the body of the main loop can be marked as a *dynamic* region. This causes the profiling system to record a separate profile for each iteration of the loop. However, for long running applications with a large number of iterations, this may lead to prohibitively high memory consumption.
To reduce the memory requirements of the profile, we added a mechanism that clusters similar iterations into a single profile sub-tree instance [7]. The maximum number of clusters is configurable. However, we only cluster iterations with a similar structure. Iterations that contain differences beyond a certain threshold in their call paths are not clustered together. The user can select from a set of modes that specify the criteria to determine whether iterations are similar. The time dynamics can be reconstructed from a mapping table, which stores the cluster associated with each iteration.
In the case of the SPEC MPI benchmarks, the clustered profiles match the profiles without clustering very well, even if only 64 clusters are used. In the case of clustering the more complex PEPC coulomb solver, count based metrics show already a good match when using 64 clusters. However, time based metrics require more clusters for a good match [7].
A prototype was implemented in the Scalasca measurement system EPIK. For HOPSA, we implemented the same mechanisms in the Score-P measurement system. We can cluster one dynamic region. However, the user can specify multiple dynamic regions. Time-series profiling is fully supported for MPI. OpenMP and hybrid applications are supported with the limitation that the dynamic regions are not clustered if they are inside a parallel region. Thus, the iterations create distinct sub trees, but the compression is not applied. However, if the clustered region contains a parallel region the clustering works as expected. The implementation in Score-P was merged into the trunk in December 2012. The Score-P 1.2 release will contain the clustering functionality.

DoW Subtask 2.2C
Extend the Open Trace Format for profile snapshots and restart points. The profile snapshots will allow a coarse summary about the performance behaviour of predetermined time intervals. The restart points need to provide all status information required to start reading trace events from this point without the preceding events. (TUD in cooperation with previous task)

Partial trace loading contributes to enable the analysis of large traces generated by many processes running over a long period of time. The new support for snapshots (restart points) in OTF2 allows Score-P to write state information about the trace monitoring. The snapshot information is added to a trace in a post-processing step after the actual application execution. This snapshot information allows an OTF2 analysis tool to start reading in the middle of a trace rather than from the beginning, and still have consistent information about the context, e. g., call-stack and performance metrics. Furthermore, statistical profiling information gives a coarse summary about the performance of the trace before loading it. This information is generated during the post-processing step that also adds the snapshots and can be used to select a specific period of interest before loading the trace. These requirements for partial loading of OTF2 traces are now completely implemented and will be part of the next OTF2 release 1.2 that is planned for June 2013.

DoW Subtask 2.2.D
Based on the previous task, Vampir will be extended to allow partial loading and visualisation in the time and space (processes/threads) dimensions. Based on profile snapshots, a pre-selection of time intervals and processes will be offered to the user. (TUD)

Using Vampir's partial loading feature it is now possible to perform a selective analysis of specific hot spot areas without the need of loading the complete trace data. The trace data to be loaded can be restricted in time and space dimensions. Therefore, a user-friendly trace overview allows selecting an arbitrary time range as well as processes/threads of interest for loading. For more details please see deliverable D2.3.

DoW Subtask 2.2.E
Extend the run-time measurement for selective trace recording for Vampir with respect to time and location (processes/threads). The selection may be specified in fixed form before the trace recording or by adding control statements within the target application similar to phase markers. (TUD)

For selective trace recording Score-P provides several possibilities. The first one is a configuration file that specifies regions that should be traced. Second, the recording can be manually switched on or off via user instrumentation. This work was done within the SILC project. A new third option is the long-term event-trace recording mode that was introduced in the HOPSA project (see next subtask 2.2.F). The latter also provides the facility of selective tracing with respect to locations (processes/threads).

DoW Subtask 2.2.F
Introduce a long-term event-trace recording mode to the SILC monitoring component. It will allow to discard the preceding section of the event trace at certain control points or phase markers. The live 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. (TUD)

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 behavior patterns as well as on similarity or difference with other sections. This mode is part of the latest public release. For more details please see deliverable D2.3.

DoW Subtask 2.2.G
Design a scalable method for collecting ThreadSpotter's performance fingerprint data from each of the MPI strands running in a scalable system. This involves designing a filtering function that identifies MPI strands 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 strands and can concentrate on the unique behaviour. (RW)

The purpose of MPI rank clustering is to reduce the number of reports and analyses that the user has to view after sampling a distributed job. The idea is that many ranks will show the same behavior and it would only be necessary to look at a few reports to gain sufficient knowledge of all interesting information, provided that the selected reports are well chosen. The feature extracts a condensed subset of information from each sample file acquired across the cluster, and then searches for groupings of the features. It then proceeds to create reports only for one representative member of each group, and present them to the user. For more details, see deliverable D.2.3, section 3.2.2.
The latest improvements is that ThreadSpotter can now achieve scalable launching in a MPI environment through the use of a tree based overlay network (TBON) which connects and synchronizes the various sampler agents on the different nodes. This together with improvements in the user interface makes launching sampling and reporting jobs in a cluster as easy as when running on a single node or standalone workstation.

DoW Subtask 2.2.H
Increase the scalability of CEPBA-Tools by intelligently deciding which information is emitted. This work will be based on the current on-line analysis implemented within the Extrae instrumentation library. The scalability would also be enhanced by parallelising the Paraver kernel and improving the file management. (BSC)

Within the scope of HOPSA Extrae was extended to include a time analysis module on the on-line mode. The new module looks for repetitive patterns across periodic intervals of the execution and automatically selects small, representative regions that describe the application phase behavior. This work was reported on a paper presented at ICPADS 2011 conference [18]. This subtask/task was completed already in the first period. For more details please see deliverable D2.3.
The parallelization of Paraver engine has been implemented targeting the multi-core architectures that are nowadays in most of our laptops/desktops. To easily fulfill this approach we selected the programming model OmpSs that it is based on tasks with dependencies. The computation of the two Paraver main views (timelines and tables) has been parallelized with a very good efficiency between 80 and 95%. Small modifications were required on the code to enable the tasks to work with local data and to minimize the dependencies between tasks. After these changes, the taskification of the code required very few new lines of code (less than 20).

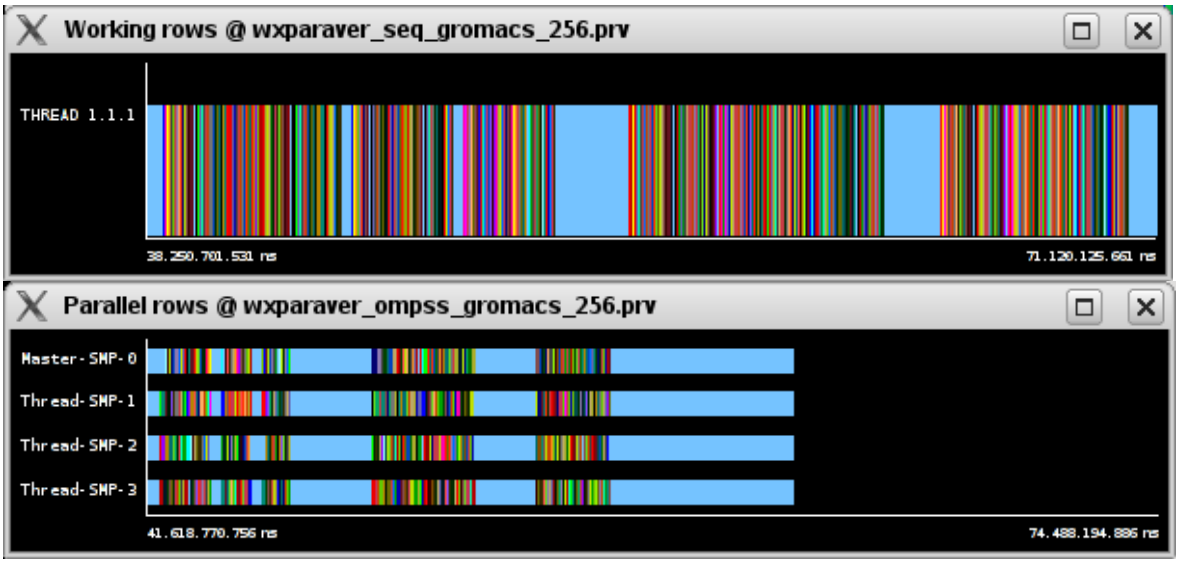


Figure 5: Timelines of Paraver processing a 256 -process trace of GROMACS. Top: sequential version, Bottom: parallel version of Paraver. Each colour corresponds to a row processing.

Extrae and Paraver were used to analyze the executions of the parallel version and to compare with the original sequential version. To provide an example and compare both versions we have captured loading of a Paraver session with both the sequential and the parallel binaries. The session is used to mimic a user execution where after loading the trace file, multiple views are created. The main benefit of this approach is that the gap between the windows creation is similar for both executions so we can compare them visually setting both timelines at the same time duration.

Figure 5 shows the timeline for both runs where we can see the improvements on time reduction. On the two top captures, the colored regions correspond to the Paraver kernel computations that have been parallelized (each color correspond to a row of either the timeline or the table). The light blue regions between colored regions correspond to the windows drawing. In Figure 6 all the colors have been masked to dark blue to easily identify the 5 regions that correspond to the computation of 3 timelines followed by the computation of 2 tables.

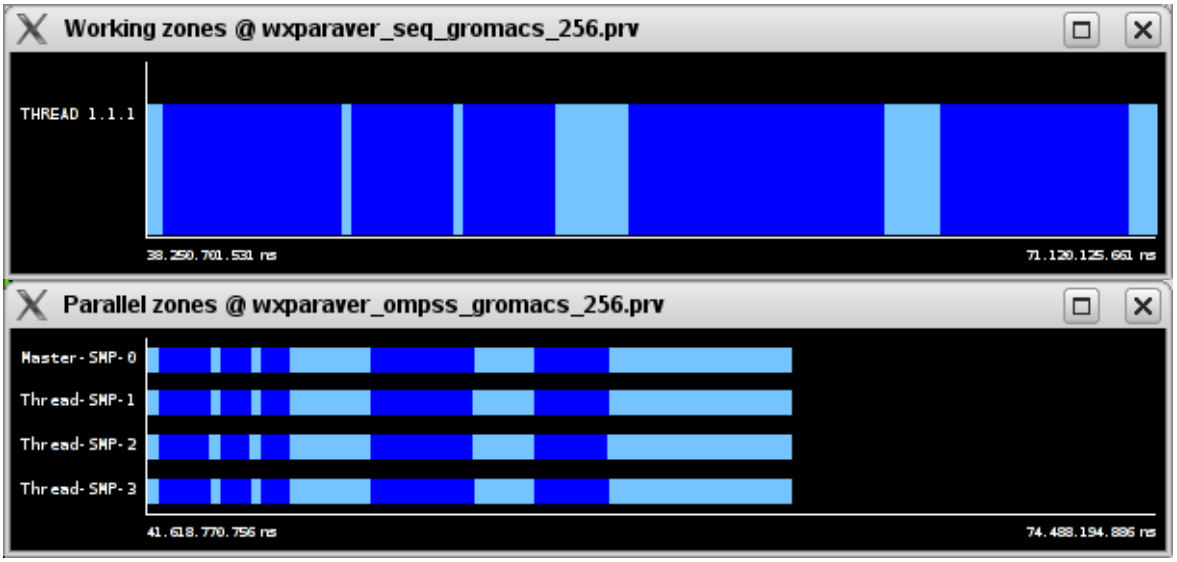


Figure 6: Mask of Figure 5. Each dark blue region corresponds to a window computation. Top: sequential version, Bottom: parallel version of Paraver

Figure 7 shows the histogram of the duration of the different kernel calls. Both histograms are at the same scale so a perfect speed-up would represent to move the duration of the computation (each colored cell) to ¼ the distance to the left.

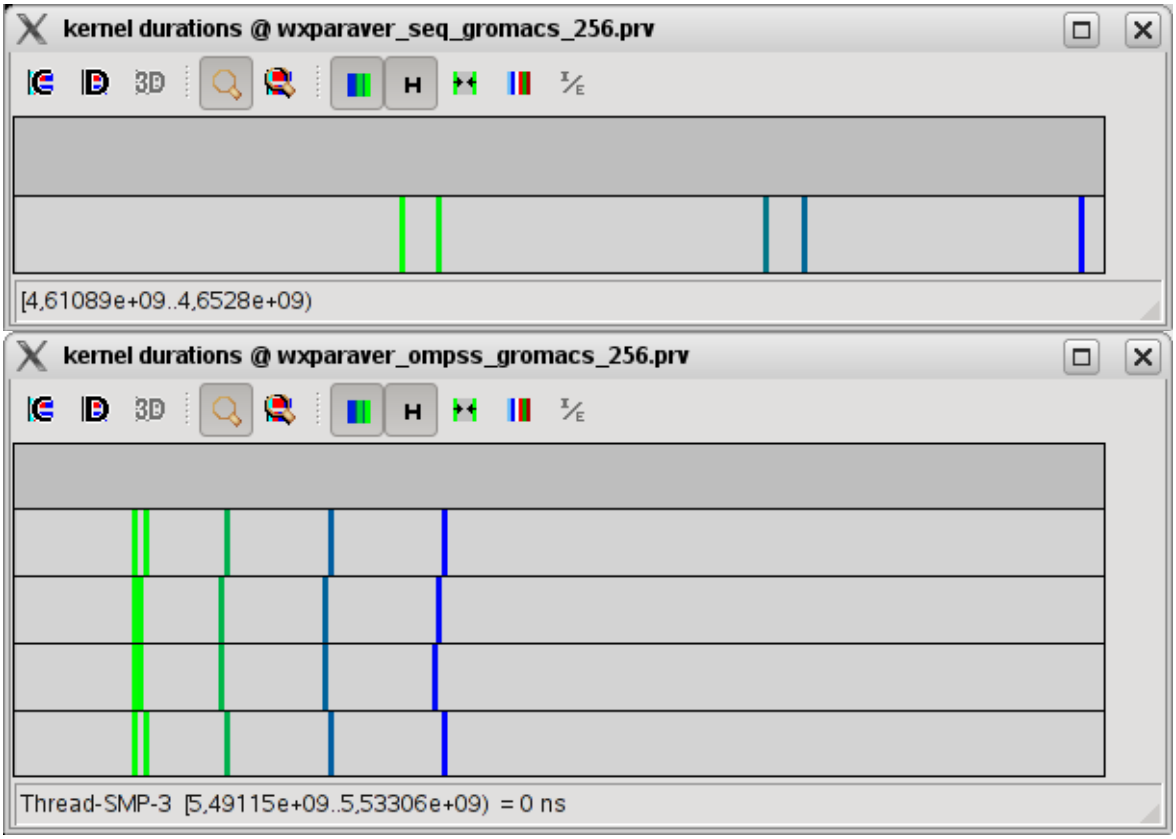


Figure 7: Histogram of duration of different kernel calls in sequential (top) and parallelized (bottom) Paraver version

We can see that the timelines computation achieve a better speed-up than the tables computation but we can consider both of them are efficient.

DoW Subtask 2.2.I

Currently, the Dimemas simulator only scales up to few thousand processes and it is necessary to extend its scalability to be used at large scale. This implies the redesign of some internal structures of the simulator and the reimplementation of the module that generate the output trace. (BSC)

Before HOPSA project Dimemas scalability was limited to very few thousand processors due to the high number of file descriptors required and some constrains on the implementation. There were even more restrictions when the user wanted to generate a Paraver trace file as output of the simulation. The improvements implemented in the project allowed to move the limits at least one order of magnitude with a huge performance improvement. For more details please see deliverable D2.3.

Task 2.3 Tool integration

The objective of this task was to investigate in how much the existing HOPSA-EU tools can be integrated to work as a coherent performance measurement and analysis tool set. This includes the implementation of the integration of Scalasca's result browser Cube with the timeline-based visualizers Paraver and Vampir, the conversion of OTF2 trace files into the Paraver trace format, and the Paraver / Dimemas integration as well as to investigate new innovative additional tool integrations.

DoW Subtask 2.3.A
Integration of Scalasca's interactive report explorer on the one side with Vampir or Paraver on the other side to allow detailed investigation of the most severe instances of performance problems located by the Scalasca analysis with the comprehensive statistical and visualisation features of Vampir and Paraver. A demonstration prototype of this has been implemented based on KOJAK (a sequentially-working predecessor of Scalasca) and an older version of Vampir (based on Motif). The analysis/location part has to be parallelised and integrated into the parallel Scalasca trace analyser. The remote-trace-browsing control part has to be re-implemented for the latest version of Vampir (based on Qt) and Paraver. The implemented tool interaction protocols (DBUS, UNIX signals) will be enhanced. (BSC, JSC, TUD)

This subtask has been completed in year 1. A prototype implementation from the sequential predecessor tool of Scalasca (KOJAK) was ported over by JSC to the new parallel trace analyzer of Scalasca, which required the development of a new parallel algorithm to determine the most severe instances in a scalable way. The current implementation supports all implemented MPI patterns of Scalasca. To enable the necessary remote control of Vampir from inside CUBE, TUD implemented a remote control API via D-Bus commands. The remote control of Paraver is implemented by CUBE based on the ability of Paraver to load a specified configuration file on request by sending Paraver a UNIX USR1 signal. It is already available to users in current versions of the CUBE, Vampir and Paraver tools.

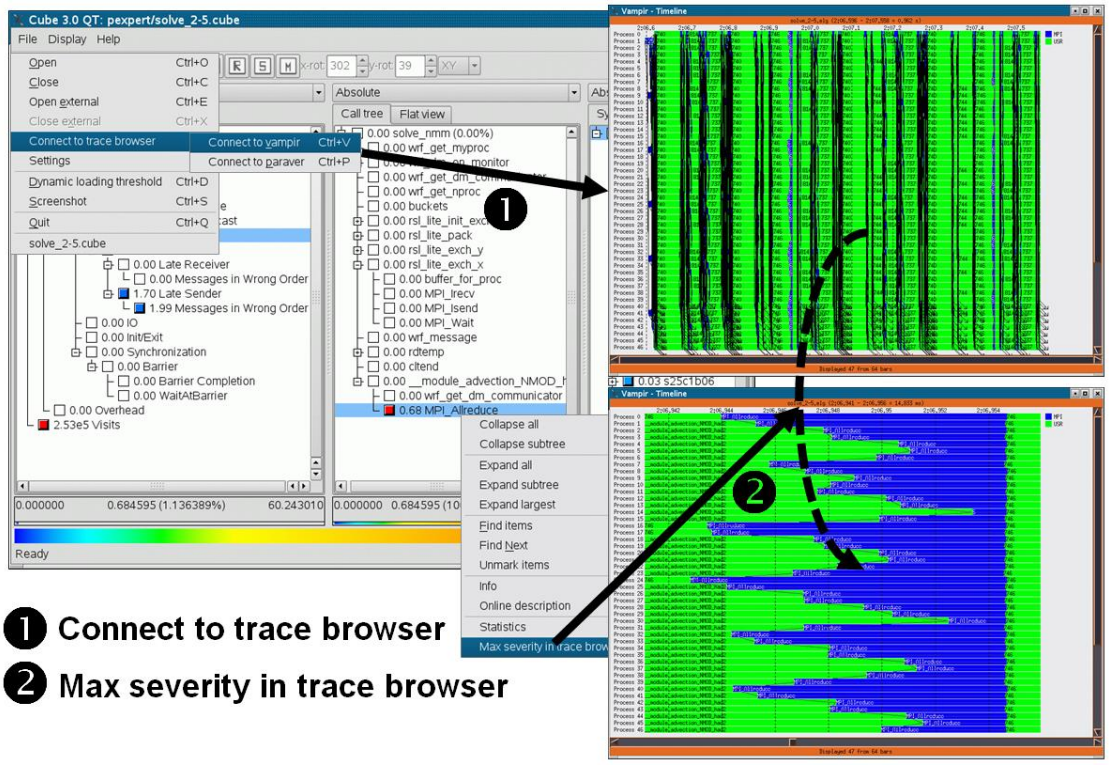


Figure 8: Scalasca →Vampir or Paraver Trace browser integration.

Figure 8 shows the basic Scalasca →Vampir or Paraver Trace browser integration. In a first step, when the user requests to connect to a trace browser, the selected visualizer is automatically started and the event trace which was the basis of Scalasca's trace analysis is loaded. Now, in a second step, the user can request to see a timeline view of the worst-instance of each performance

bottleneck located by Scalasca. The trace browser view automatically zooms to the right time interval. Now the user can use the full analysis power of these tools to investigate the context of the located performance problem.

DoW Subtask 2.3.B

Extend CEPBA-Tools to work with Open Trace Format (OTF) [18] traces. The objective is to allow Paraver to load and work with OTF traces and to implement a new version of Extrae that supports the generation of the traces in OTF format. (BSC, TUD)

During the first year of HOPSA, the OTF2 format was analyzed and different alternatives of integration were evaluated. Due to the short duration of the project and the high number of differences between the current Paraver trace format and OTF2, the final decision was to implement a translator from OTF2 to Paraver. During the second year, BSC has been working on a new module to implement this trace-to-trace translator.

The translator re-enumerates the events to generate a trace file, as similar as possible, to the one that would have been obtained instrumenting the same execution with Extrae. This way, the same Paraver configuration files can be applied to the traces obtained through Score-P. To achieve this, the translator has to recognize the events and has to implement a translation table. The events that are not included on the translation table are grouped on a unique Paraver event type. We decided to use the "user functions" event because it was not possible to identify such events from the original OTF2 trace file but it used to be an information included by Score-P. To allow a flexible and incremental approach, the translator allows the user to specify new entries of the translation table by using a special flag, however, it is not possible to overwrite the translation of known events.

The current list of translated events include the MPI calls, the OpenMP runtime calls, hardware counters metrics and user functions (as the default for the not know events). The translator has been validated instrumenting the execution of small programs with both Score-P and Extrae and comparing the two Paraver trace files obtained. Figure 9 shows a zoom on the ring example showing the MPI calls and the communication lines with both approaches. In these views, light blue corresponds to computation, dark blue to *MPI_Send* and white to *MPI_Recv*. Top capture corresponds to the Score-P trace file and the bottom one to Extrae instrumentation.

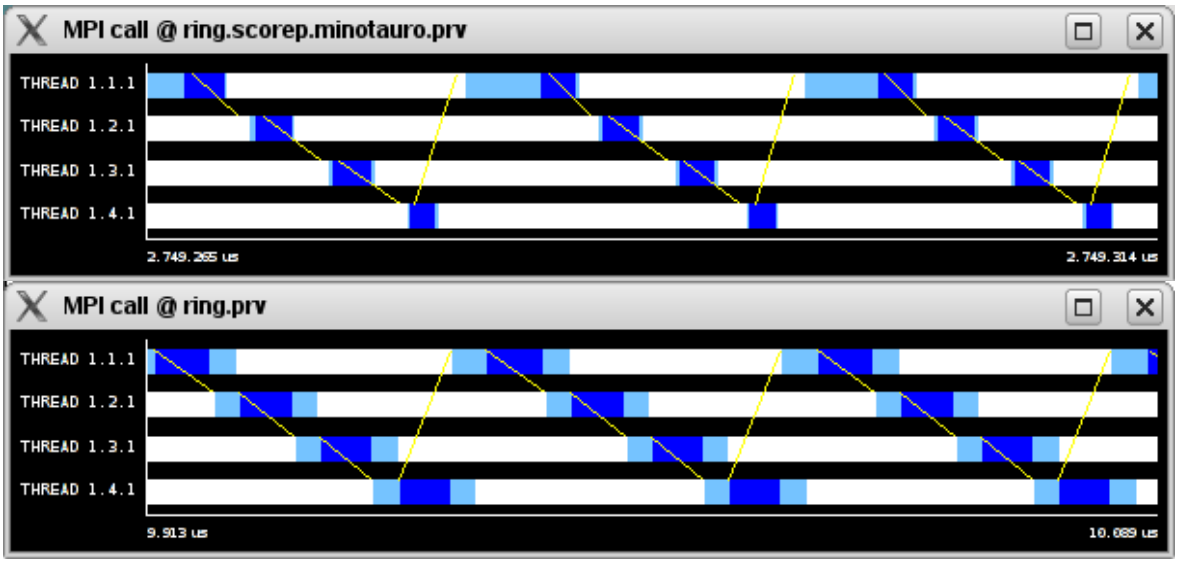


Figure 9: Comparison of experiments of MPI ring example measured with Score-P (top) and Extrae (bottom)

Considering that they correspond to two different runs, they are easily comparable although some minor differences are present. For instance, the trace file obtained using Score-P shows a clear difference on the computation duration of rank 0 with respect to the other, while all look similar in the Extrae instrumentation. The difference that causes this small variation is due to the computation done by the Score-P itself is not assigned to the *MPI_Send* call, while it is assigned in Extrae.

DoW Subtask 2.3.C
Analysis of asynchronous tasking: Emerging programming models employ the concept of asynchronous tasks. Examples are OpenMP 3.0 or StarSs tasks, CUDA, OpenCL, and generic uncoordinated (POSIX) threading. In HOPSA, we will develop an abstract characterisation of the performance of asynchronous tasking which will allow all tools to support these new models in a coherent way so that analysis results obtained with one tool can be interpreted in the light of results obtained with another. This is an important prerequisite for assigning individual tools their role in an analysis workflow capable of tracking down inefficiencies related to asynchronous tasking. Key elements of this unification will be a common terminology as well as common methaphors for representing analysis results to the user. (All-EU, JSC)

Emerging programming models employ the concept of asynchronous tasks. One goal is the development of an abstract concept to measure and analyze the performance of tasks. As a prerequisite, we developed an abstract event model for tasks. For the event model, we identified the categories task creation, task execution, task dependencies, and global synchronization:

- ***Task creation:*** Defines who can create tasks, and how they are created. At program execution start, the runtime creates one task which starts executing the main function, or task per thread that are execution the thread routine. Further tasks can be created by explicit statements, called explicit tasks. In some systems (e.g. OpenMP) all tasks may create other tasks. On other systems explicit tasks cannot create other tasks, e.g., if GPU kernels are represented as tasks and the GPU kernels cannot start other GPU kernels.
- ***Task execution:*** Defines whether a executing task may be suspended and resumed later on, where the suspension may happen, and whether a task can migrate (whether the thread that resumes a task can be different than the thread that suspended a task). Some systems (e.g. OpenMP, OmpSs) provide two task types, one that can migrate and a type that cannot migrate. Tasking systems that do not allow task suspension do not migrate tasks.
- ***Task dependencies:*** In order to ensure that the results of task executions can be used in subsequent computations, tasking systems provide means to define dependencies between tasks. A task may not start/resume execution until all its dependencies are completed. The way how a user can specify dependencies, varies. E.g. OmpSs allows do specify input and output data for every task. Furthermore, it allows to manually specifying dependencies to particular tasks. In OpenMP the only possible dependency is that a task waits for its direct child tasks. Besides, the explicit task dependencies, there might be implicit task dependencies. E.g., in OmpSs child tasks complete only if, all their children are completed, while in OpenMP, no implicit dependencies exist, if we do not count the always present task creation dependency.
- ***Global synchronization:*** A very common structure is a global barrier that waits for the completion of all explicit tasks.

For the performance analysis of tasks, we identified the task size and dependency chains as paradigm-specific, performance critical issues.

- ***Task size:*** The first performance critical goal for tasking paradigms is the task size [12]. If tasks are executed on external units, like GPUs, the data transfer and management time to execute a task/kernel, is significant. Thus, too small tasks may cost more management time, than the execution of task on the host system would take.
  On pure host sided systems, tasking may provide means for automated load balancing, however, experiments [8] show that the overhead can become a significant contributor and reduce the scaling the performance of an application significantly.
  Thus, tasks must have a minimum size for efficient execution. Because, it is hard to guess the execution time of a task, the execution time of tasks may differ, the execution time may depend on the input of the tasks, and is also system dependent, tools are needed to identify task with an inefficient runtime.
  If task are created by only one master thread, and the execution time of the tasks is too small, the task creation may become a bottleneck, causing threads to idle. On the other hand, too few, too large tasks may reduce the load balancing effects.
- ***Dependency chains:*** A task dependency structure that has too little parallelism on parts of its critical path causes threads to idle, and thus, incurs inefficiencies.

Based on the previous considerations, we can conclude that we want the following information to be covered by the event model:

- The runtime of the task.
- The time for task creation.
- The management time associated with a task.
- The creation relationship between tasks.
- The dependencies between tasks.
- Time spend in synchronization constructs.

In order to measure timing, we associate each code region of interest with a region definition which contains region type information. Because measuring time difference is a very common method, Score-P and OTF2 already provide events for enter and exit of regions which we reuse. However, we need to extend the available region types. Thus, we only need new events for information that is not already covered. In, particular, we still need:

- An event that carries information about the creation relationship.
  - An event when on task switches, which tells us, which task is resumed. If the measurement system maintains information on the current task, it can derive which task was suspended.
  - An event for every task dependency pair.

With this set of events we can cover all proposed analysis:

- Task execution: The task is a new region that has enter and exit events. If the task is suspended and resumed in between, we get switch events on suspension and resumption and thus, can subtract the suspended time. Task start and end are marked by enter/exit events and switches from/to the task.
- Task creation: The duration is captured by enter/exit events to a task creation region. Inside this region we provide an event that transports information about the task creation relationship.
- The management time: Here we can measure the time spend in task creation regions, memory transfer regions, and task scheduling points. These regions can be marked by region types.
- Task dependencies: To abstract from the particular specification method, we can model the whole dependency system, if we create pairs of a tasks that is waiting and a task that is waited on.

With respect to the performance analysis, we developed a mechanism to record information about the task size in the Score-P profiles. Its use is demonstrated with OpenMP tasks [8]. It provides information about the execution time of tasks, overhead/waiting time in synchronization points and creation time. Parameter based profiling may differentiate the tasks on their input data, to identify appropriate input data sizes. Inside the tasks, a complete profile may be recorded to give detailed insight into the task. It is also possible to provide a time-line view of the task execution as demonstrated in [12]. In his master thesis [1], Youssef Hatem demonstrated that the full task dependency graph can be generated from the presented event model for OpenMP.

DoW Subtask 2.3.D

Paraver and Dimemas are integrated at the level of traces: Paraver traces can be translated to Dimemas and Dimemas simulation can produce a Paraver trace as part of its output. We will extend this integration to allow that Paraver calls Dimemas to analyse what-if scenarios producing new Paraver traces that can be loaded on the visualizer. (BSC)

One of the HOPSA targets for increasing the tools interoperability has been to integrate Paraver analyzer and Dimemas simulator. The goal is to enable a user to raise what-if analysis from the Paraver GUI. For instance when looking at an application with large point to point communications between processes, the user can consider interesting to evaluate if having a larger network bandwidth would represent a significant improvement or not. BSC has implemented a new window in Paraver to launch external applications that includes a predefined set-up for the Dimemas tool. With this new functionality a user can now easily launch Dimemas simulations from the Paraver GUI and load the resulting trace. Figure 10 shows the new Paraver window and the typical input for the

Dimemas runs that include the simulator configuration and the original Paraver trace file. It also includes a button to launch the Dimemas GUI. On the text box at the bottom of the window the user will see the output of the application run.

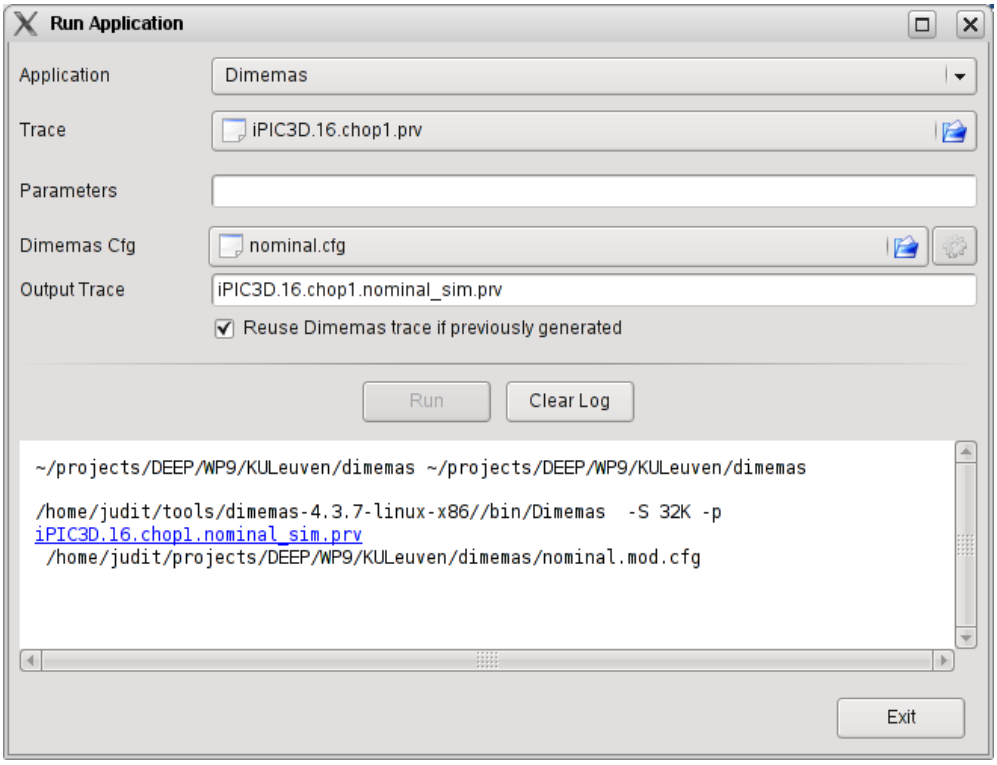


Figure 10: Paraver → Dimemas Integration:
Preparation of launching Dimemas on Paraver trace file

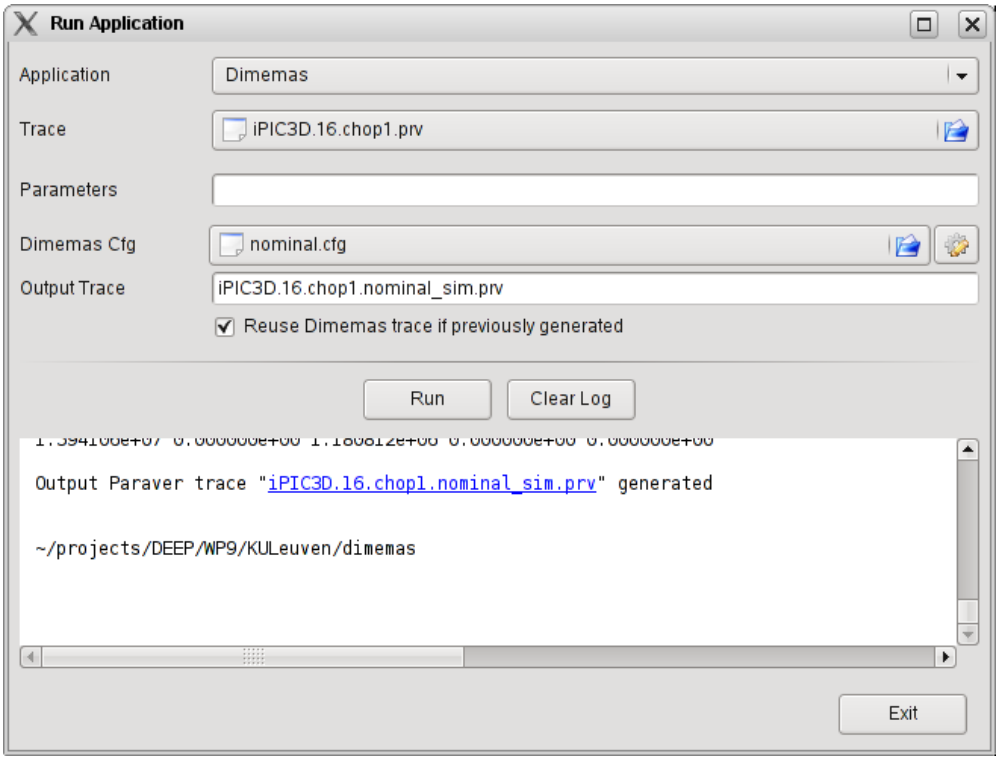


Figure 11: Paraver → Dimemas Integration: Result output after Dimemas launch.
The simulation trace file can be loaded by clicking on the trace name link.

Paraver does a very simple processing of the application output identifying for instance the files with extension prv that correspond to Paraver traces. These files are translated to links that can be used to easily load the simulated trace file. Figure 11 shows the last lines of output from the Dimemas run and the link indicating the trace file generated.

Currently this functionality can be run from the main Paraver menu or from the filters menu so it can be applied to a subset of the trace file. In the future this functionality will be included on the contextual menu of the timeline, so the user can easily apply the what-if analysis to the area that is currently looking.

DoW Subtask 2.3.E
Investigate integration between timeline-based visualizers (Paraver, Vampir) and/or Scalasca and ThreadSpotter. For example, use ThreadSpotter's identification of MPI strands with unique behavior to superimpose regularity/irregularity information on Paraver/Vampir displays. Alternatively, reduce the Paraver/Vampir displays to show only one representative per behaviour group. (RW, BSC, TUD, JSC).

The integration within the HOPSA performance tools in new and innovative ways, beyond what was already designed and put in the work plan (Subtasks 2.3.A to 2.3.D), has been a discussion point on most technical meetings.

Work first concentrated on the design of a more generic mechanism which allows launching arbitrary tools from the Cube tool (in contrast to the hardcoded interaction with Paraver and Vampir). After JSC implemented a working prototype, BSC integrated their tool for the sampled call stack analysis with CUBE. This tool generates CUBE input from Paraver trace files, correlating the sources with the folded hardware performance counter metrics. This allows launching gnuplot from CUBE to display the time evolution data. This new functionality is described in detail in section "Integration of Cube in the BSC Folding analysis" in the deliverable 2.3 ("Tool Validation Report").

Based on these positive experiences, work concentrated on a possible design to link Score-P's / Scalasca's time-based profiling results to a ThreadSpotter analysis of the same executable. The idea is to link call paths in Cube with a high execution time to the corresponding memory and threading analysis of ThreadSpotter, if it has data about the same call path. As Score-P/Scalasca only has summary metrics data about the whole execution of the called function but no further details, it might be useful for the application developer to understand the high resource usage by investigating the memory and threading analysis of ThreadSpotter for the same function. However, because of different measurement techniques (direct, instrumentation-based in the case of Score-P, and binary instrumentation in case of ThreadSpotter), it is not easy to determine the equivalence of two call paths coming from the different tools. For example, the Score-P instrumentation does not cover low-level system calls or compiler runtime functions which are of course visible on the binary level. Also, a working comparison needs to match OpenMP constructs (as seen by Score-P) to outlined functions and OpenMP runtime system functions implementing these constructs seen on the binary level. This is unfortunately very compiler specific. Finally, ThreadSpotter does not see manually instrumented regions and phases from Score-P. All-in-all, a working equivalence test would require some intelligent fuzzy matching. This issue was further investigated and successful matches have been performed in case of simple C or Fortran MPI codes. However, a reliable, portable implementation would require much more work and further experiments. Once the working equivalence test is found and implemented, the implementation of the actual interaction between Cube and ThreadSpotter is technically straight forward. RW already has done successful experiments with controlling the Threadspotter user interface from outside agents. We hope to find funding for this interaction in a follow-up project.

The integration of Paraver and Vampir with Threadspotter would work in the same way. Both trace analyzers provide visualizations of metric values over time. Again, if hardware counters related to memory (e.g. cache or TLB misses) or threading have been measured, and the corresponding visualization shows unusually (high or low) values, these could be linked to the memory and threading analysis of ThreadSpotter for the function for which these values have been measured. Again, this requires the matching of call stack representations between the different tools.

Task 2.4 Tool validation

Dow Task 2.4

The performance tools of project (ThreadSpotter, Paraver, Dimemas, Scalasca, Vampir) are already in daily use on the production systems of the computing centres of the partners (BSC, JSC, ZIH) or by customers of Rogue Wave where they will be applied to application codes of computing centre users. This will allow to validate whether the proposed enhancements of work package 2 (described by Tasks T2.1 to T2.3) are working and are useful in the analysis of real-world applications. These use cases will be documented. In addition, we will apply the tools to a set of benchmark codes (e.g. from the SPEC MPI, NAS, ASCI benchmark suites) at the begin and end of the project to investigate and report the improved efficiency and scalability of the tools. (All-EU, JSC)

The daily use of the HOPSA performance tools on the production systems of the computing centres of the project partners BSC, JSC, and ZIH or by Rogue Wave customers of course helps the application owners to understand and improve the performance of their codes, but it also allows the HOPSA tool developers to test and validate whether the enhancements of work package 2 (described by Tasks T2.1 to T2.3) are working and are useful in the analysis of real-world applications. Some of these use cases were documented in the deliverable D2.3 (“Tool Validation Report”). In addition, the results of applying the HOPSA tools to a set of benchmark codes (SPEC MPI) and large scale application suites (DEEP, Mont-Blanc, Cresta, TEXT) from relevant Exascale EU FP7 projects are reported. Please see the extensive deliverable for more details.

3.2.2 WP3: Integration of system and application performance analysis

We integrated the software for system-level monitoring and analysis from our Russian HOPSA-RU partners with the HOPSA-EU tool set from WP2. After the definition of the interfaces between the system- and application-level components in the first year, which have been documented in deliverable D3.1 (“Requirements for the Interface between System-level and Application-level Performance Analysis”), work concentrated on the refinement and specification of the overall performance-analysis workflow which was documented in deliverable D3.2. The implementation of the central HOPSA component, the so-called light-weight monitoring module (LWM^2), which measures and extracts an application-level performance signature and delivers it to the system-level cluster monitoring layer, was finalized and documented in deliverable D3.3. In addition, both the Extrae/Paraver and Score-P/Vampir tool sets have been enhanced to allow the enrichment of performance analysis results and visualizations with system-level sensor data. Finally, an integrated package of all HOPSA tools with consistent documentation and one unified installation mechanism was provided and documented in deliverable D3.4.

In the following the results achieved in the different tasks of this work package are described in detail.

Task 3.1 Definition of the interface between system- and application-level performance analysis

DoW Task 3.1

Define an interface to interchange performance related results between the system level, job level, and low-level application analysis on the one hand and the high-level performance tools on the other hand.

Part of this interface will be a performance report which compiles essential information from system-level monitoring and application-centric measurement into a performance report document. This document will give an overview about the essential performance properties, including hints about potential performance deficiencies and how to verify their presence with other tools. Furthermore, it should allow to compare the performance behaviour with past reports of the same application to survey the success of analysis and tuning. Another part of the interface is to specify how performance related results of the system-, job-, and low-level application analysis is made available to the high-level performance tools. (All-EU and All-RU, JSC)

The interfaces between the system- and application-level components have been defined and documented in deliverable D3.1 (“Requirements for the Interface between System-level and Application-level Performance Analysis”) already in the first year. The work in the second year concentrated on the implementation of the central light-weight monitoring module LWM2 and its integration with the Russian ClustrX.Watch monitoring system and LAPTA data analysis and

visualization layer. Various variants of communication resulting in live updating of LAPTA database through the communication network were investigated but were found to be stressful on system resources. Finally a file-based data-exchange solution, providing post-mortem data update was found appropriate and has been implemented. If desired, LWM2 can provide a simple form of the performance report (job digest) at the end of job execution. More elaborate reports as well as cross-job analysis is available in the LAPTA system from our Russian partners. It provides a language-based query interface (HOPLANG) as well as a web-based portal.

Finally, both the Extrae/Paraver and Score-P/Vampir tool sets have been enhanced to allow the enrichment of performance analysis results and visualizations with system-level sensor data. For this, Vampir was improved to select, display, and associate performance information in the scope of system nodes rather than individual processes in the form of the new feature called “System Background View” (see Task 2.1.B). Paraver flexibility allowed displaying the new system metrics with no changes on the visualizer. BSC efforts were focused on the evaluation of the different sources of system metrics (streamed and historic data). The Paraver and Vampir enhancements are described in more detail in deliverables D1.1, D2.2 and D2.3

Task 3.2 Definition of the overall performance analysis workflow

DoW Task 3.2

Definition of an HOPSA overall performance tool process and workflow which guides the application developers in the process of tuning and optimising their codes for performance in the form of a written user guide. The guide will give an overview of all tools available for the performance analysis of user applications and will describe which tools and in which order they should be used to accomplish specific common typical performance analysis tasks; also taking into account the results of experiments already performed (i.e. historic data) (All-EU and All-RU, JSC)

The indented usage and application of the HOPSA performance tools was specified by the HOPSA performance-analysis workflow and documented in deliverable D3.2. It was also successfully used to structure training classes on the use of HOPSA tools, as it nicely captures the high integration of our tools set.

The workflow 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, Scalasca, ThreadSpotter, and Vampir. The data collected by LWM^2 is also fed into the Clustrx.Watch hierarchical cluster monitoring system which combines it with system and hardware data and forwards it to the LAPTA cluster monitoring and analysis system 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.

Task 3.3 Light-weight monitoring module

DoW Task 3.3

A light-weight monitoring module, which will be implemented as a shared library so that it can be loaded prior to the execution of the parallel job by the job launcher, will collect basic performance metrics such as execution time, hardware counters, and message-passing statistics. To keep the overhead at a minimum, only those metrics will be collected that do not require expensive instrumentation. (GRS, JSC, TUD and All-RU, GRS)

The Light Weight Measurement Module (LWM^2) was developed by GRS during the course of the project. It is designed to transparently monitor and profile every job running on an HPC system, with minimal overhead. It profiles MPI, pthreads-based multithreading, POSIX file I/O, hardware counters through PAPI and CUDA based applications. It further enables correlating application performances through the concept of globally synchronized time-slices. The profiling data from LWM^2 is fed into the LAPTA database from MSU, enabling a central repository with rich system-wide profiling information. LWM2 is also deployed at a number of other HPC sites. LWM^2 combines the hybrid profiling method with its novel approach of synchronized time-sliced profiles, resulting in a 3-dimensional profiling strategy. A considerable challenge was to reconcile the diverse concepts of sampling, direct instrumentation and threading based time-slices while maintaining data integrity. For more details please see deliverable D3.3.

Task 3.4 Unified download, configuration, build and installation package

DoW Task 3.4

High-performance clusters often provide multiple MPI libraries and compiler suites for parallel programming. This means that parallel programming tools which often depend on a specific MPI library, and sometimes on a specific compiler, need to be installed multiple times, once for each combination of MPI library and compiler which has to be supported. In addition, over time, newer versions of the tools also get released and installed.

One way to manage many different versions of software packages used by many computing centres all over the world is the ”module” software (see http://www.modules.org). However, each centre provides a different set of tools, has a different policy on how and where to install different software packages, and how to name the different versions. Our proposal ”UNiform Integrated Tool Environment” (UNITE) will improve this situation for debugging and performance tools by

- specifying exactly how and where to install the different versions of tool software packages (including
- integrating the tools to the maximum possible degree),
- defining standard module names for tools and their different versions, and
- supplying predefined module files which provide standardised, well-tested tool configurations,
- but still being flexible enough to be able to coexist with site-local installations, restrictions, and policies.

In addition, a ”meta”-Installation tool will be developed capable of configuring, building, and installing all HOPSA tools as a common package but hiding tool-specific aspects of the various phases. This work will be based on a successful prototype which can handle Scalasca and Vampir as well as a few other open-source tools developed as part of the EU ITEA-2 project ParMA. (JSC in cooperation with All-EU and All-RU, JSC)

The work in this task was based on a successful prototype developed as part of the EU ITEA-2 project ParMA (***http://www.parma-itea2.org/***) which could handle Scalasca, Cube3, Vampir, and VampirTrace as well as a few other open-source tools (PDToolkit, Marmot). In this project, the prototype was enhanced in many ways:

- The UNITE module file package was re-designed and re-implemented so that an existing UNITE installation can be updated with newer versions without losing site-specific adaptations and changes.
- Support for configuring, building and installing new packages has been added:
    - The HOPSA light-weight monitoring module LWM^2
    - The BSC tools Extrae and Paraver which made it necessary to also support the additional development packages libdwarf, libunwind, boost, wxpropgrid

  - The Score-P components scorep, otf2, cube4, and opari2 as well as scalasca2
  - The TAU package of the University of Oregon (a contributor and member of the Score-P community project)
  - The new binary installation packages of Vampir and VampirServer introduced with version 7
- The much larger number of packages and the trend to “componentize” tool modules allowed to greatly enhancing the integration among the different packages. This required implementing much more complicated configuration, build and install procedures so that for example when installing an updated version of a tools package it is still integrated with already existing and installed versions of other tool components.
- Support for detecting more MPI library variants (MPICH version 3, BullX MPI, IBM POE MPI library for x86 and x86_64 Linux (intelpoe) and PPC Linux (ibmpoe), and IBM Platform MPI) and configuring the packages accordingly.

Finally, various versions of the UNITE package have been extensively tested on a large variety of HPC Linux clusters all over the world.

On the UNITE website (***http://apps.fz-juelich.de/unite/***), a single package consisting of the UNITE installer, the UNITE module files, as well as a large set of open-source performance tools together with a user guide (for application developers) and an installation guide (for system administrators) was provided. For more details see the deliverable D3.4 (“UNITE Package”)

3.3 Project management during the period

The project was managed and coordinated as planned. Quarterly project meetings have been organized including our Russian project partners every other meeting. All deliverables were submitted and all project milestones have been reached. The project website has been greatly improved and enhanced (***http://www.hopsa-project.eu***) and has been linked with the website of our Russian partners (***http://hopsa.parallel.ru***). Project dissemination over all was extremely successful: the project was presented at seven events (including ISC and SC), often by multiple partners (1st year: four), 13 training events involving HOPSA tools haven been organized (1st year: 12), and 11 project-related publications have been published (1st year: 2) with four completed additional publications waiting for publication.

3.3.1 WP1: Project management

Task 1.1 Administrative and financial management

- The first year project review at the EC, Brussels, on March 13, 2012 was organized and the corresponding financial and administrative reporting was performed.
- The second year project review at JSC, Jülich, on April 18, 2013 was organized as well as the corresponding final financial and administrative reporting was performed.

Task 1.2 Technical coordination

- A project meeting between the EU partners has been organized and hosted by GRS in Aachen on February 22nd and 23rd, 2012. All partners participated.
- An international project meeting between all (EU and Russian) project partners was prepared by JSC and MSU and was hosted by JSCC, RAS on May 31st and June 1st, 2012 in Moscow. All partners participated.
- A project meeting between the EU partners has been organized and hosted by TUD in Dresden on September 6th, 2012. All partners participated.
- A final international project meeting between all (EU and Russian) project partners was prepared by JSC and MSU and was hosted by MSU on November 26th, 2012 in Moscow. All partners participated.

3.3.2 Project Dissemination

The HOPSA project partners were very successful in promoting both the HOPSA project as well as the HOPSA software tools as part of presentations, posters, BoFs, and flyers at major international HPC conferences (ISC, SC, EuroPar) and training activities of the leading European initiatives (PRACE, DEISA, VI-HPS). The following lists all dissemination actions for both years of the project in detail (2012 and 2011 items have partially already reported in the first year progress report):

General dissemination actions

2013

- Article ***HOPSA erleichtert Optimierung*** (German) in Exascale Newsletter 1/2013, Forschungszentrum Jülich, Germany

2011

- Article ***HOPSA Project launched*** in InSiDE newsletter Nov 2011, GCS, Germany
- Article ***HOPSA Project launched*** in JSC News Feb 2011, JSC, Germany

- A coordinated press release about the HOPSA project was prepared in cooperation with the public relation departments of all EU and Russian partners. It was published on June 21st, 2011 just in time for the ISC 2011 conference in Hamburg. The press releases are available at
  - ***http://www.t-platforms.com/about-company/press-releases/183-hopsa.html***
  - ***http://www.fz-juelich.de/SharedDocs/Pressemitteilungen/UK/EN/2011/11-06-20hopsa.html***
  - ***http://www.bsc.es/media/4469.pdf***
  - ***http://tu-dresden.de/die_tu_dresden/zentrale_einrichtungen/zih/publikationen/dateien/hopsa_en.pdf***
  - ***http://www.grs-sim.de/news-events/news-archive/faster-computations-with-hopsa.html***

  The press release got picked up by many other sides including HPCwire, InsideHPC, Primeur, Scientific Computing, and many more.
- A two-page fact sheet was prepared summarizing the project. It is available at the project website.
- A basic project website is available since June 2011 under the URL ***http://www.hopsa-project.eu*** describing the objectives and goals of the project and listing the project partners. The web site is embedded in the web site of the Virtual Institute – High-Productivity Supercomputing (VI-HPS) as the HOPSA project has many synergies with other funded project and training activities of VI-HPS.

Dissemination at international HPC conferences and workshops

In addition to the events listed in this section there also have been presentations at the 5th and 6th International Parallel Tools Workshop, PROPER 2012, PSTI 2012, IWOMP 2012, IPDPS 2012, ParCo 2012, EuroMPI 2011, and ICPADS 2011 (see section ***Refereed publications*** below).

2013 (planned)

- BSC will present the Tutorial ***Understanding applications performance with Paraver*** at EuroMPI 2013, Madrid, Spain on September 15th, 2013.
- The project will be well presented at the ***International Supercomputing Conference*** (ISC 2013) in Leipzig, June 16th to 20st, 2013. The EU partners BSC, JSC, Rogue Wave and TUD will have booths in the research or commercial exhibition of the conference. Our Russian partners MSU and T-Platforms will also have exhibition booths. The project will be presented with posters, flyers, and on-line demonstrations on the show floor. Furthermore:
  - JSC will present a half-day tutorial ***Performance Analysis & Optimization on Extreme-scale Systems*** on June 16th, 2013.
  - JSC, RW, and TUD will participate in the BoF ***Execution Analysis & Optimization of Parallel Applications*** on June 19th, 2013.
- JSC and TUD presented Score-P, Scalasca, and Vampir in the Tutorial ***Trace-based Performance Analysis with Vampir and Scalasca*** at SEA 2013, UCAR, Boulder, USA on April 4th and 5th, 2013.

2013

- TUD presented the HOPSA project at the APOS/HOPSA Workshop ***Exploiting Heterogeneous HPC Platforms*** in conjunction with HiPEAC 2013, Berlin, Germany on January 22nd, 2013.

2012

- The project was well presented at the ***International Conference for High Performance Computing, Networking, Storage and Analysis*** (SC 2012) in Salt Lake City, November 10th to 16th, 2012. The EU partners BSC, JSC, Rogue Wave and TUD and for the first time MSU had booths in the research or commercial exhibition of the conference. The project was presented with posters, flyers, and on-line demonstrations on the show floor. In addition, the HOPSA project and the HOPSA tool set was introduced and promoted in the following activities:

  - A half-day tutorial ***Supporting Performance Analysis and Optimization on Extreme-Scale Computer Systems*** on November 12, 2012. Involved partners: JSC.
  - GRS organized a Workshop ***Extreme-Scale Performance Tools*** on November 16, 2012. Involved partners: BSC, GRS, JSDC, TUD.

- JSC presented the HOPSA project at the Workshop ***International Research Collaboration in Computing Systems*** at HiPEAC Computing Systems Week, Gent, Belgium on October 17th, 2012.
- BSC, JSC, RW and TUD presented their tools at the ***Workshop on Tools for Exascale*** at CEA, Bruyères-le-Châtel, France on October 1st and 2nd, 2012.
- TUD and JSC presented Score-P [4], Scalasca [3], Vampir [5] and the HOPSA workflow [2] at the ***6th International Parallel Tools Workshop*** at HLRS, Stuttgart, Germany on September 25th and 26th, 2012.
- TUD and JSC presented Score-P, Vampir and Scalasca at the tutorial ***Practical Hybrid Parallel Application Performance Engineering*** at EuroMPI 2012, Vienna, Austria on September 23rd, 2012.
- The project was well presented at the ***International Supercomputing Conference*** (ISC 2012) in Hamburg, June 17th to 21st, 2012. The EU partners BSC, JSC, Rogue Wave and TUD had booths in the research or commercial exhibition of the conference. Our Russian partners MSU and T-Platforms also had exhibition booths. The project was presented with posters, flyers, and on-line demonstrations on the show floor. Furthermore:
  - A half-day tutorial ***Supporting Performance Analysis & Optimization on Extreme-Scale Computer Systems*** on June 17, 2012. Involved partners: JSC
  - A BoF ***Parallel Application Execution Analysis & Optimization*** on June 20th, 2012. Involved partners: RW, JSC
  - A BoF ***Automatic Node & System Performance Analysis at Scale*** on June 20th, 2012. Involved partners: RW, JSC
- JSC, MSU, and EPCC organized the EU-Russia APOS/HOPSA workshop on May 30th, 2012 at the Academy of Sciences, Moscow,

2011

- The project was well presented at the ***International Conference for High Performance Computing, Networking, Storage and Analysis*** (SC 2011) in Seattle, November 14th to 17th, 2011. Again, the EU partners BSC, JSC, Rogue Wave and TUD had booths in the research or commercial exhibition of the conference. The project was presented with posters, flyers, and on-line demonstrations on the show floor. In addition, the HOPSA project and the HOPSA tool set was introduced and promoted in the following activities:
  - A full day tutorial including hands-on training ***Hands-on Practical Hybrid Parallel Application Performance Engineering*** was organized and held on November 13th, 2011. Involved partners: GRS, JSC, TUD.
  - A Birds-of-a-Feather (BoF) session ***System Monitoring Meets Application Performance Analysis - The HOPSA EU-Russia Project*** was organized and held on November 17th, 2011. Involved partners: GRS, JSC, TUD.
  - A Birds-of-a-Feather (BoF) session ***The Score-P Community Project -- An Interoperable Infrastructure for HPC Performance Analysis Tools*** was organized and held on November 17th, 2011.
- BSC, JSC, and TUD presented “Score-P - A Joint Performance Measurement Run-time Infrastructure for Periscope, Scalasca, TAU, and Vampir” [9], “Trace-Based Performance Analysis of GPU Accelerated Applications” and “Folding: Detailed Analysis with Coarse Sampling” [10] at the ***5th Parallel Tools Workshop***, HLRS/TUD, Dresden on Sep 26th to 27th, 2011.
- The project was well presented at the ***International Supercomputing Conference*** (ISC 2011) in Hamburg, June 19th to 23rd, 2011. The EU partners BSC, JSC, Rogue Wave and TUD had booths in the research or commercial exhibition of the conference. Our Russian partners MSU and T-Platforms

also had exhibition booths. The project was presented with posters, flyers, and on-line demonstrations on the show floor. Furthermore:

  - A full day tutorial ***Hands-On Course on Debugging & Optimizing Parallel Programs*** which included ThreadSpotter was organized and held on June 19th, 2011 by Rogue Wave.

- Rogue Wave presented "How to port applications to new architectures" at the ***3rd PRACE Industrial Seminar***, Stockholm, Sweden, on March 29th, 2011.

HPC training events

2013 (planned)

- BSC will introduce Extrae, Paraver, and Dimemas at the PRACE Advanced Training Centre (PATC) training course ***Performance Analysis and Tools***, Barcelona, Spain on May 13th to 14th, 2013.
- JSC, GRS and TUD will provide a PATC "Score-P, Scalasca and Vampir Introduction and Hands-on Training" at the ***11th VI-HPS Tuning Workshop***, CEA, Saclay, France on April 22nd to 26th, 2013.

2012

- RW provided a ***ThreadSpotter Training Day***, Jülich Supercomputing Centre, Jülich, Germany on December 4th, 2012.
- All EU project partners introduced the complete HOPSA workflow and toolset at the ***APOS/HOPSA Training Week***, Research Computing Center, Moscow State University, Russia on November 27th to 30th, 2012.
- JSC, GRS and TUD provided a PATC "Score-P, Scalasca and Vampir Introduction and Hands-on Training" at the ***10th VI-HPS Tuning Workshop***, LRZ, Munich, Germany on October 16th to 19th, 2012.
- JSC and TUD the Scalasca and Vampir toolsets at the Training workshop ***Scalable Performance Analysis Tools for HPC Applications,*** CSCS, Lugano, Switzerland on August 9th and 10th, 2012.
- JSC gave an introduction into parallel performance analysis and presented the HOPSA tool set at the ***2012 European-U.S. Summer School on HPC Challenges in Computational Sciences***, Dublin, Ireland on June 26, 2012.
- JSC presented the Scalasca and Vampir toolsets at a ***Program Analysis and Tuning Workshop***, DKRZ, Hamburg, Germany on June 25th and 26th, 2012.
- BSC introduced Extrae, Paraver, and Dimemas at the PATC training course ***Performance Analysis and Tools***, Barcelona, Spain on May 21st to 22nd, 2012.
- JSC, GRS and TUD provided a PATC "Score-P, Scalasca and Vampir Introduction and Hands-on Training" at the ***9th VI-HPS Tuning Workshop***, Université de Versailles, St-Quentin-en-Yvelines, France on April 23rd to 27th, 2012.
- JSC gave an introduction into parallel performance analysis and presented the HOPSA tool set at the ***Doctoral School: Computational Interdisciplinary Modelling*** of the University of Innsbruck at Obergurgl, Austria on January 28th to 31st, 2012.
- BSC presented Paraver and Dimemas at ***CAPAP-H winter seminar***, Valladolid, Spain on January 26th to 27th, 2012.

2011

- BSC presented Paraver and Dimemas at the ***PRACE Autumn School 2011***, Bruyères-le-Châtel, France on October 25th to 27th, 2011.
- BSC, JSC, GRS and TUD provided a "Scalasca, Paraver and Vampir Introduction and Hands-on Training" at the ***8th VI-HPS Tuning Workshop***, RWTH, Aachen, Germany on September 5th to 9th, 2011.
- JSC presented "Performance analysis tools for massively parallel applications" and a "Scalasca Introduction" at the ***PRACE Summer School: Taking the Most Out of Supercomputers***, Helsinki, Finland on August 29th to September 1st, 2011.

- JSC gave an introduction into parallel performance analysis and presented the HOPSA tool set at the DEISA/PRACE/TeraGrid ***2011 European-US Summer School on HPC Challenges in Computational Sciences*** at Lake Tahoe, USA on August 9th, 2011.
- JSC presented an "Introduction to Performance Engineering and Scalasca" at the ***Joint HP-SEE, LinkSCEEM-2 and PRACE HPC Summer Training***, Athens, Greece on July 15th, 2011.
- JSC gave an introduction into parallel performance analysis and presented the HOPSA tool set at the ***Russian Summer School on Supercomputing Technologies*** at MSU, Moscow, Russia on July 1st, 2011.
- BSC presented Paraver and Dimemas at ***HPC-EUROPA2/TAM 2011***, Barcelona, Spain on June 8th to 9th, 2011.
- BSC and GRS presented Paraver, Dimemas and Scalasca at the INRIA Sumner School ***Toward petaflop numerical simulation on parallel hybrid architectures***, INRIA, Sophia Antipolis, France on June 6th to 10th, 2011.
- JSC presented "Parallel application performance analysis with Scalasca" at the ***DEISA/PRACE Spring School on Tools and Techniques for Extreme Scalability***, EPCC, Edinburgh, UK, on March 29th to 31st, 2011.
- RW, JSC, GRS and TUD provided a "ThreadSpotter, Scalasca and Vampir Introduction and Hands-on Training" at the ***7th VI-HPS Tuning Workshop***, HLRS, Stuttgart, Germany on March 28th to 30th, 2011.

Refereed publications

Please note that if one of the publications below were part of a conference or workshop proceedings that of course the paper was also presented at the corresponding workshop or conference even if the event is not explicitly listed under the section ***Dissemination at international HPC conferences and workshops*** above.

2013

[1] Youssef Hatem, ***Critical Path Analysis of Parallel Application Using OpenMP Tasks***. Master thesis. GRS Aachen, Forschungszentrum Jülich, to appear.

[2] 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.

[3] Daniel Lorenz, David Böhme, Bernd Mohr, Alexandre Strube, and Zoltán Szebenyi. ***New developments in Scalasca***. In: Proceedings of the 6th International Parallel Tools Workshop, Stuttgart, September 2012, Springer. To appear.

[4] Andreas Knüpfer, Robert Dietrich, Jens Doleschal, Markus Geimer, Marc-André Hermanns, Christian Rössel, Ronny Tschüter, Bert Wesarg and Felix Wolf. ***Generic Support for Remote Memory Access Operations in Score-P and OTF2***. In: Proceedings of the 6th International Parallel Tools Workshop, Stuttgart, September 2012, Springer. To appear.

[5] Holger Brunst und Matthias Weber. ***Custom Hot Spot Analysis of HPC Software with the Vampir Performance Tool Suite***. In: Proceedings of the 6th International Parallel Tools Workshop, Stuttgart, September 2012, Springer. To appear.

[6] Harald Servat, Xavier Teruel, Germán Llort, Alejandro Duran, Judit Giménez, Xavier Martorell, Eduard Ayguadé and Jesús Labarta.***On the Instrumentation of OpenMP and OmpSs Tasking Constructs***. In: Proceedings of the PROPER2012 Workshop, Springer. ***DOI:10.1007/978-3-642-36949-0_47***

2012

[7] Zoltán Szebenyi. ***Capturing Parallel Performance Dynamics***. PhD thesis, RWTH Aachen University, volume 12 of IAS Series, Forschungszentrum Jülich, 2012, ISBN 978-3-89336-798-6. ***URI:http://hdl.handle.net/2128/4603***

[8] Daniel Lorenz, Peter Philippen, Dirk Schmidl, Felix Wolf. ***Profiling of OpenMP tasks with Score-P***. In Proc. of the 41st International Conference on Parallel Processing Workshops (ICPPW 2012), Pittsburgh, PA, USA, 2012.
DOI:10.1109/ICPPW.2012.62

[9] Andreas Knüpfer, Christian Rössel, Dieter an Mey, Scott Biersdorff, Kai Diethelm, Dominic Eschweiler, Markus Geimer, Michael Gerndt, Daniel Lorenz, Allen D. Malony, Wolfgang E. Nagel, Yury Oleynik, Peter Philippen, Pavel Saviankou, Dirk Schmidl, Sameer S. Shende, Ronny Tschüter, Michael Wagner, Bert Wesarg, Felix Wolf. ***Score-P – A Joint Performance Measurement Run-Time Infrastructure for Periscope, Scalasca, TAU, and Vampir***. In Proc. of 5th Parallel Tools Workshop, 2011, Dresden, Germany, pages 79-91, Springer Berlin Heidelberg, September 2012.
DOI:10.1007/978-3-642-31476-6_7

[10] Harald Servat, Germán Llort, Judit Giménez, Kevin Huck, Jesús Labarta., ***Folding: detailed analysis with coarse sampling***. In Proc. of 5th Parallel Tools Workshop, 2011, Dresden, Germany, Springer Berlin Heidelberg, September 2012.
DOI:10.1007/978-3-642-31476-6_9

[11] Marc-André Hermanns, Markus Geimer, Bernd Mohr, Felix Wolf. ***Scalable detection of MPI-2 remote memory access inefficiency patterns***. Intl. Journal of High Performance Computing Applications (IJHPCA), 26(3):227–236, August 2012.
DOI:10.1177/1094342011406758

[12] Dirk Schmidl, Peter Philippen, Daniel Lorenz, Christian Rössel, Markus Geimer, Dieter an Mey, Bernd Mohr, Felix Wolf. ***Performance Analysis Techniques for Task-Based OpenMP Applications***. In Proc. of the 8th International Workshop on OpenMP (IWOMP), Rome, Italy, volume 7312 of Lecture Notes in Computer Science, pages 196–209, Berlin / Heidelberg, Springer, June 2012.
DOI:10.1007/978-3-642-30961-8_15

[13] David Böhme, Bronis R. de Supinski, Markus Geimer, Martin Schulz, Felix Wolf. ***Scalable Critical-Path Based Performance Analysis***. In Proc. of the 26th IEEE International Parallel & Distributed Processing Symposium (IPDPS), Shanghai, China, IEEE Computer Society, May 2012.
DOI:10.1109/IPDPS.2012.120

[14] David Böhme, Markus Geimer, Felix Wolf. ***Characterizing Load and Communication Imbalance in Large-Scale Parallel Applications***. In Proc. of the 26th IEEE International Parallel & Distributed Processing Symposium (IPDPS) PhD Forum, Shanghai, China, IEEE Computer Society, May 2012.
DOI:10.1109/IPDPSW.2012.321

[15] Dominic Eschweiler, Michael Wagner, Markus Geimer, Andreas Knüpfer, Wolfgang E. Nagel, Felix Wolf. ***Open Trace Format 2 - The Next Generation of Scalable Trace Formats and Support Libraries***. In Proc. of the Intl. Conference on Parallel Computing (ParCo), Ghent, Belgium, August 30 – September 2 2011, volume 22 of Advances in Parallel Computing, pages 481–490, IOS Press, 2012.
DOI:10.3233/978-1-61499-041-3-481

[16] Markus Geimer, Pavel Saviankou, Alexandre Strube, Zoltán Szebenyi, Felix Wolf, Brian J. N. Wylie. ***Further improving the scalability of the Scalasca toolset***. Proc. PARA 2010, Minisymposium Scalable tools for High Performance Computing, Reykjavik, Iceland, LNCS 7134, pp. 463-474, Springer, 2012.
DOI:10.1007/978-3-642-28145-7_45

2011

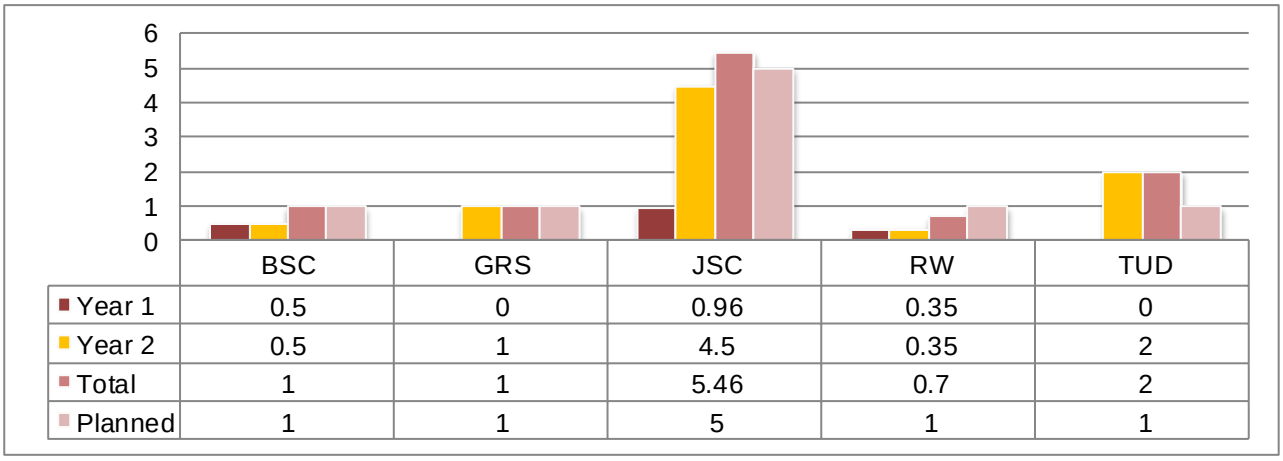
[17] Markus Geimer, Marc-André Hermanns, Christian Siebert, Felix Wolf, Brian J. N. Wylie. ***Scaling Performance Tool MPI Communicator Management***. Proc. EuroMPI 2011, Santorini, Greece, LNCS 6960, pp.178-187, Springer, 2011.
DOI:10.1007/978-3-642-24449-0_21

[18] Germán Llort, et al. ***Trace Spectral Analysis toward Dynamic Levels of Detail***. Proc. ICPADS 2011, Tainan, Taiwan, pp. 332 - 339, 2011.
DOI:10.1109/ICPADS.2011.142

3.4 Explanation of the use of the resources

In the second year of the project we finished all planned work as well as spending efforts where in the first year work was delayed due to delays in hiring appropriate people at the start of the project by some of the partners.

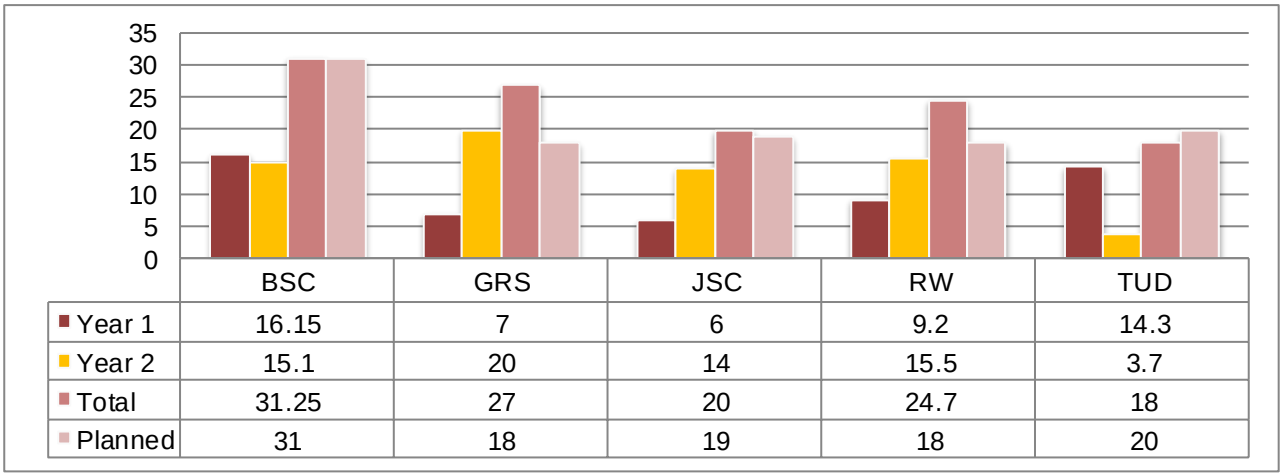
The following figures show how the work done in the project is spread over the different work packages and how well the numbers are aligned with the plan.



	BSC	GRS	JSC	RW	TUD
Year 1	0.5	0	0.96	0.35	0
Year 2	0.5	1	4.5	0.35	2
Total	1	1	5.46	0.7	2
Planned	1	1	5	1	1

Figure 12. **Total Efforts (PM): WP1 (MGMT)**

There was very little effort planned for work package 1 (“Project management”). Figure 12 shows the total efforts (in person months) related to WP1. BSC, GRS and RW are roughly in plan. JSC spent some extra efforts in organizing the originally not planned APOS/HOPSA training week in November 2012 and in improving the HOPSA website. TUD was responsible for two extensive and important deliverables (D2.2 "Final Tools Set" and D2.3 "Tool Validation Report") requiring extra coordination and management efforts. TUD also was responsible for organizing technical arrangements with MSU for the APOS/HOPSA training week.



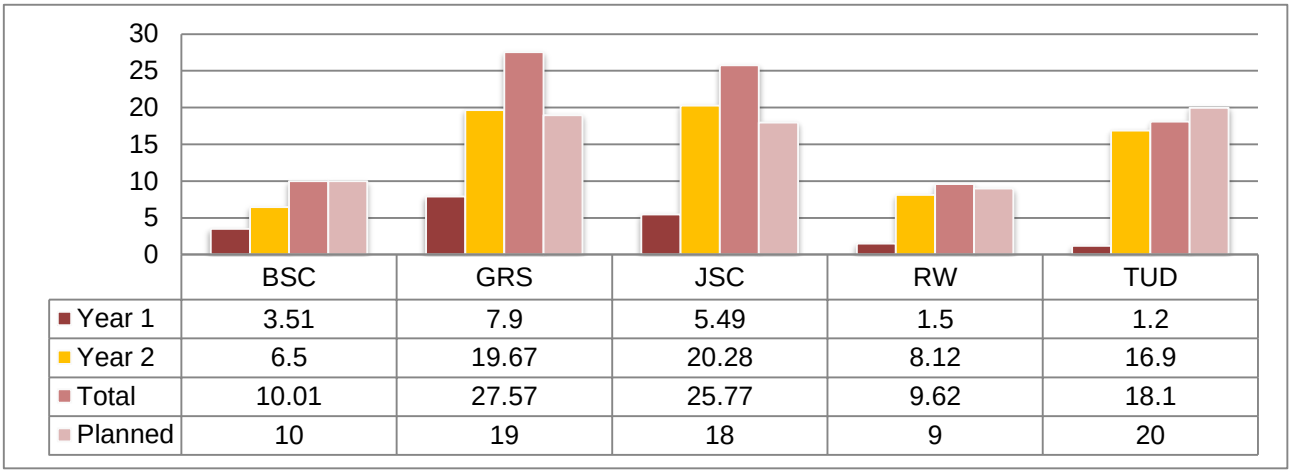
	BSC	GRS	JSC	RW	TUD
Year 1	16.15	7	6	9.2	14.3
Year 2	15.1	20	14	15.5	3.7
Total	31.25	27	20	24.7	18
Planned	31	18	19	18	20

Figure 13. **Total Efforts (PM): WP2 (RTD)**

Figure 10 shows the total efforts (in person months) related to work package 2 (“HPC application-level performance analysis”). All objective and goals of work package 2 were reached. Overall, BSC, JSC and TUF were roughly in plan.

GRS had to invest much more effort than planned into WP2 for several reasons. First, the changes required to implement the scalable communicator handling affected more parts of the code than expected. Second, the changes of the underlying abstractions required to implement the root-cause analysis turned out to be more profound than calculated. Finally, the original developer of the time-series profiling technique left the team in the middle of 2012 after obtaining his PhD degree. For this reason, new staff had to be trained to implement the required extensions. This also included familiarizing with the relatively complex code written by the original developer.

RW spent more effort than planned since the task of launching and sampling MPI jobs is a scalable fashion proved itself more difficult than originally planned. Still, the preliminary results were encouraging enough to motivate us to put in the extra effort to complete the necessary tasks.



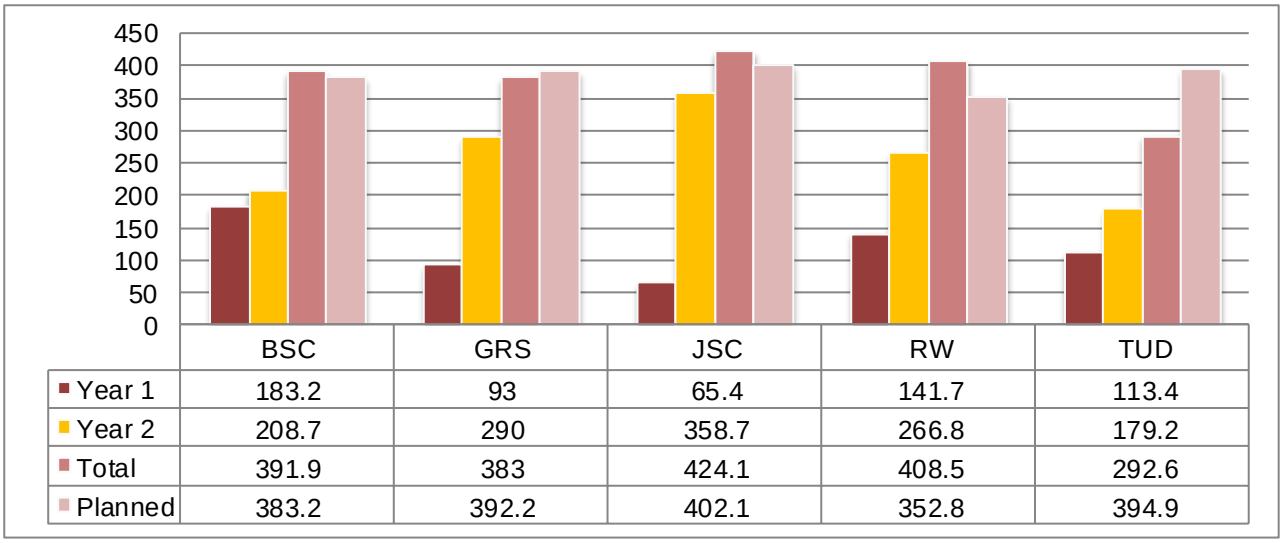
	BSC	GRS	JSC	RW	TUD
Year 1	3.51	7.9	5.49	1.5	1.2
Year 2	6.5	19.67	20.28	8.12	16.9
Total	10.01	27.57	25.77	9.62	18.1
Planned	10	19	18	9	20

Figure 14. **Total Efforts (PM): WP3 (RTD)**

Figure 11 shows the total efforts (in person months) related to work package 3 (“Integration of system and application performance analysis”). All objective and goals of work package 3 were reached. BSC, GRS and TUD successfully could catch-up the delayed start in work package 3 in the first year due to the difficulty to get the necessary access to the development platforms of our Russian partners. Overall, BSC, RW, and TUD were roughly in plan.

JSC had to spent more efforts to finish the UNITE package as planned. Integration of Extrae and Paraver was more difficult as expected due to the need for adding support for extra development packages (libdwarf, libunwind, boost, wxpropgrid). In addition, the testing of UNITE involved a much higher effort as planned due to the release of much more than expected new tool package versions and greater than expected variability in HPC cluster system configurations.

GRS developed the LWM^2 tool during the course of the project. The breadth of technologies covered along with the novel 3-dimensional profiling approach created a scenario requiring more efforts on part of GRS to finish the tool. In particular, the in-situ aggregation of thread-level performance data posed a big challenge to the developers because it is non-trivial how this aggregation can occur without introducing race conditions. Several solutions were tried and discarded until an efficient and correct solution had been found. Finally, testing of the tool and verifying its overhead requirements demanded more effort due to natural system variation and overhead anomalies for a few cases.



	BSC	GRS	JSC	RW	TUD
Year 1	183.2	93	65.4	141.7	113.4
Year 2	208.7	290	358.7	266.8	179.2
Total	391.9	383	424.1	408.5	292.6
Planned	383.2	392.2	402.1	352.8	394.9

Figure 15. **Total Costs (k€)**

Finally, Figure 15 provides a rough overview of the total costs (in k€) for the overall project. The detailed numbers can be found in the tables and Form C generated by NEF attached to this report. Overall the costs are roughly in plan (BSC, GRS, and JSC) with a slight over-spending of RW and under-spending TUD. The much higher under-spending in year 1 due to delays in hiring appropriate people at the start of the project (GRS, TUD), and due to lower than planned costs for employees (GRS, JSC, TUD) could be corrected in the second year.

In total, the HOPSA project spent 25 k€ (1.3%) less than planned.

4. Deliverables and milestones tables

Deliverables

TABLE 1A. DELIVERABLES 1ST YEAR

Del. no.	Deliverable name	Version	WP no.	Lead beneficiary	Nature	Dissemination level[2]	Delivery date from Annex I (proj month)	Actual / Forecast delivery date Dd/mm/yyyy	Status No submitted/ Submitted	Contractual Yes/No	Comments
D1.1	Intermediate Progress Report	1	WP1	JUELICH	R	PP	M12	24/02/2012	accepted	yes	In time Renamed from "Intermediate Activity Report"
D2.1	Intermediate Tool Set	1	WP2	BSC	P	PU	M12	24/02/2012	accepted	yes	In time
D3.1	Requirements for the Interface between System-level and Application-level Performance Analysis	1	WP3	GRS	R	PP	M6	15/08/2012	accepted	yes	In time Renamed from "API Requirements Report"

[2] **PU** = Public
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 1B. DELIVERABLES 2ND YEAR

Del. no.	Deliverable name	Version	WP no.	Lead beneficiary	Nature	Dissemination level	Delivery date from Annex I (proj month)	Actual / Forecast delivery date Dd/mm/yyyy	Status No submitted/ Submitted	Contractual Yes/No	Comments
D1.2	Final Progress Report	1	WP1	JUELICH	R	PP	M24	05/04/2013	Submitted	yes	In time
D2.2	Final Tool Set	1	WP2	TUD	P	PU	M21	15/01/2013	Submitted	yes	2 months late
D2.3	Tool Validation Report	1	WP2	TUD	R	PP	M23	15/03/2013	Submitted	yes	2 months late
D3.2	Workflow Report	1	WP3	RW	R	PU	M15	15/06/2013	Submitted	Yes	In time
D3.3	Light-weight Monitoring Module	1	WP3	GRS	P	PU	M18	31/01/2013	Submitted	Yes	5 months late
D3.4	UNITE Package	1	WP3	JSC	P	PU	M22	05/04/2013	Submitted	Yes	4 months late Lead changed from GRS

Additional preparation for and testing during the originally unplanned APOS/HOPSA week in November 2012 resulted in delaying the deliverable of the final tool set (D2.2) by two months. This in turn delayed the work on the Tool Validation Report (D2.3) because all contributors were also part of the work on D2.2.

Due to major changes in design and implementation of LWM2 in the last six months of the project, D3.3 was delayed to reflect the actual functionality of LWM^2 at the end of the project. The delayed availability of LWM^2 resulted in a late delivery of the UNITE package as the necessary support for configuring, building and installation of these package could not be implemented earlier.

Milestones

TABLE 2. MILESTONES							
Milestone no.	**Milestone name**	**Work package no**	**Lead beneficiary**	**Delivery date from Annex I dd/mm/yyyy**	**Achieved Yes/No**	**Actual / Forecast achievement date dd/mm/yyyy**	**Comments**
MS1	**Intermediate Tool Set**	**WP2, WP3**	**TUD**	**31/01/2012**	**yes**	**24/02/2012**	**In time**
MS2	**Final Tool Set integrated as UNITE package**	**WP2, WP3**	**TUD**	**31/12/2012**	**yes**	**05/04/2013**	**4 months late**

5. Annex: Use of Resources Tables from NEF