Deliverable D1.1

Intermediate Progress Report

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

Due date of deliverable:	February 1st, 2012
Actual submission date:	May 4th, 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 first year (February 1st, 2011 to January 31st, 2012) 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

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: May / 04 / 2012

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)
CUDA	Compute Unified Device Architecture (Programming Interface for Nvidia GPGPUs)
Dimemas	Message passing performance analysis and prediction tool (BSC)
Extrae	Instrumentation and measurement component for Paraver visualizer (BSC)
GRS	German Research School for Simulation Sciences GmbH, Aachen, Germany
GPGPU	General Purpose Graphical Processing Unit
GUI	Graphical User Interface
HMPP	Hybrid Multicore Parallel Programming (Programming Model for Heterogeneous Architectures)
HOPSA	HOlistic Performance System Analysis. EU FP7 project
HPC	High Performance Computing
H4H	Hybrid Programming For Heterogeneous Architectures. EU ITEA2 project
I/O	Input/Output
JSC	Jülich Supercomputing Centre (of Forschungszentrum Jülich GmbH), Germany
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 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 is carried out by two coordinated projects funded by the EU under call FP7-ICT-2011-EU-Russia and the Russian Ministry of Education and Science. Its objective is the new innovative integration of application tuning with overall system diagnosis and tuning to maximize the scientific output of our HPC infrastructures. While the Russian consortium focuses on the system aspect, the EU consortium focuses on the application aspect.

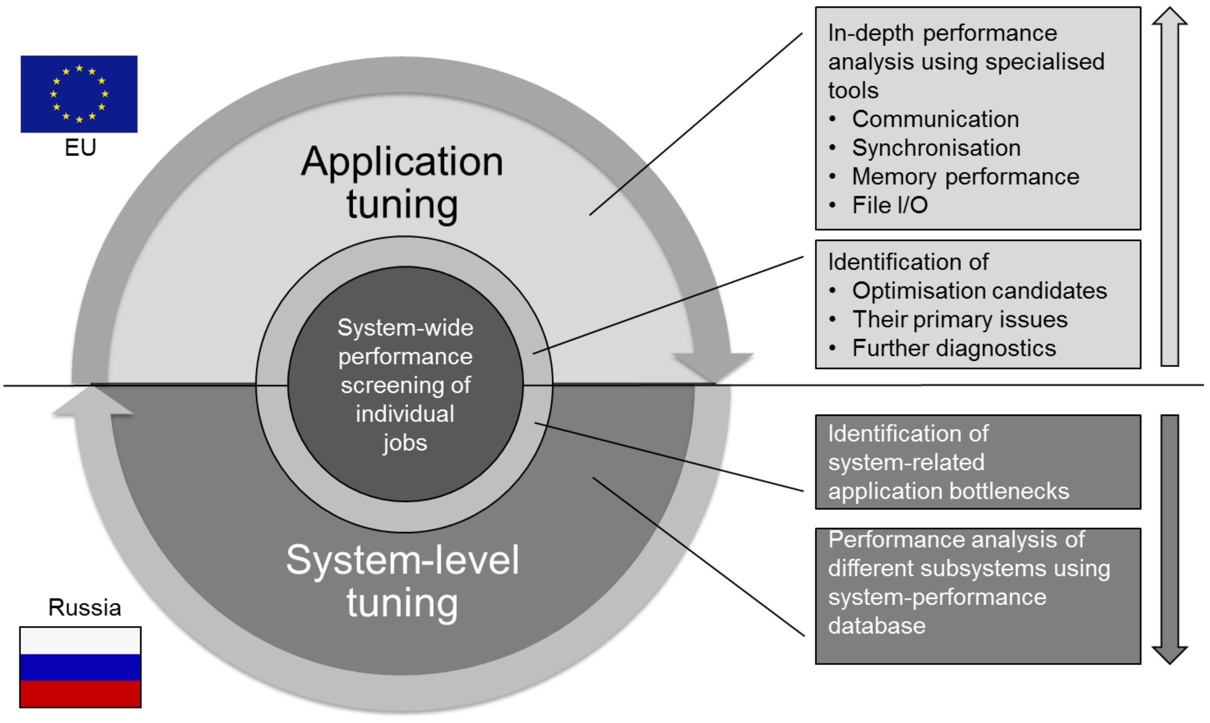

EU
Application tuning
System-wide performance screening of individual jobs
System-level tuning
Russia
In-depth performance analysis using specialised tools
• Communication
• Synchronisation
• Memory performance
• File I/O
Identification of
• Optimisation candidates
• Their primary issues
• Further diagnostics
Identification of system-related application bottlenecks
Performance analysis of different subsystems using system-performance database

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). At the end of the project (January 2013), a single unified installation package for all tools will be provided.

The HOPSA project has fully achieved its objectives and technical goals for this period (February 1st, 2011 to January 31st, 2012). All work packages are within their planned schedule. The main results for the first year are:

- ***WP1: Project management***

  The necessary infrastructure to manage and coordinate the project (mailing lists, wiki, source code revision systems, document templates, logo, etc.) was created and project work both between the EU project partners and with our HOPSA-RU partner was successfully initiated. A project website has been created (***http://www.hopsa-project.eu***) and we 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).

- ***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. Work has begun and working prototypes exist which demonstrate the growing integration between the separate tool sets of the project partners. All enhancements are either already part of the latest public releases of the software packages, or at least are available to the other partners within the project as prototype deliverable D2.1 ("Intermediate Tool Set").

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

  Considerable progress has also been made to integrate the software for system-level monitoring and analysis from our Russian HOPSA-RU partners with the HOPSA-EU tool set from WP2. Work has begun to define and further refine the overall performance-analysis workflow implementing the overall objective of the HOPSA project. 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"). For both Paraver/Extrae and Vampir/Score-P tool sets, an early prototype exists where the performance analysis results and visualizations are enriched with system-level sensor data. A prototype of the central HOPSA component, the so-called light-weight monitoring module (LWM2), which measures and extracts an application-level performance signature and delivers it to the system-level cluster monitoring layer, was implemented and is currently tested at the various partner sites. Finally, progress was also made to provide an integrated package of all tools with consistent documentation and one unified installation mechanism called UNITE.

All deliverables were delivered on time and all milestones have been reached as planned.

The HOPSA project delivers 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 and the resulting application performance improvements will be documented in a final report.

Taking an integrated approach for the first time worldwide, the involved 7 universities and research institutions considerably strengthened their scientific position as competence centres in HPC. Dresden University and Rogue Wave Software enrich 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 will 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 already in a tighter collaboration of HPC researchers from the EU and Russia.

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

2.1 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.

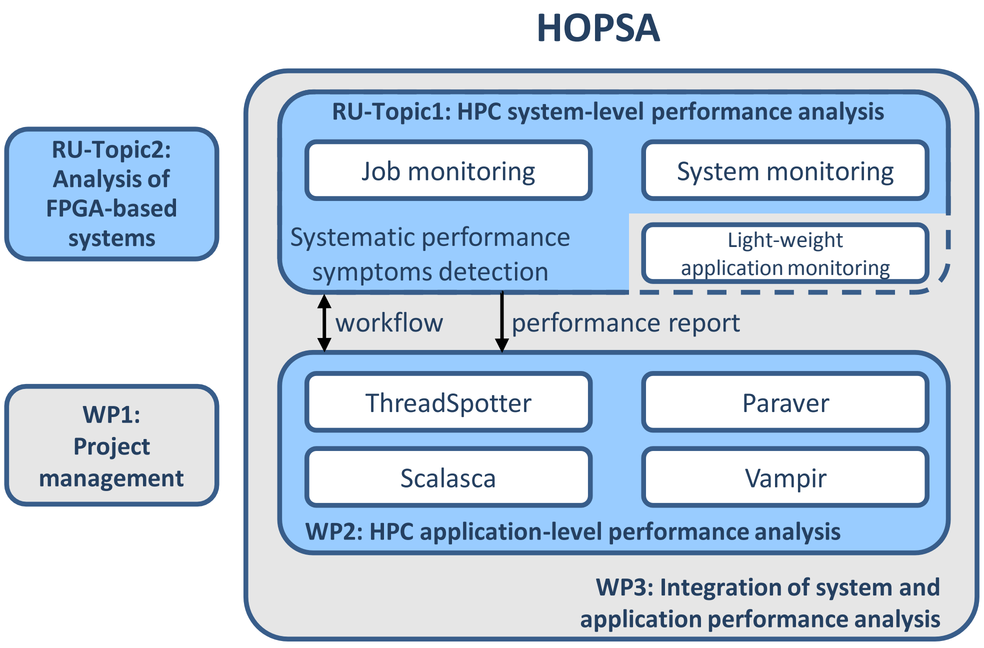

HOPSA
RU-Topic1: HPC system-level performance analysis
Job monitoring
System monitoring
Systematic performance symptoms detection
Light-weight application monitoring
RU-Topic2: Analysis of FPGA-based systems
workflow
performance report
ThreadSpotter
Paraver
Scalasca
Vampir
WP2: HPC application-level performance analysis
WP1: Project management
WP3: Integration of system and application performance analysis

Figure 2. Overall structure of the HOPSA project.

2.1.1 WP1: Project management

The objectives of this work package are 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

2.1.2 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 is 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 is not to start new research directions but rather to finalize (i.e., "productize") current research ideas and make them part of the regular tool products. 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 first year of the project was to make significant progress in the first three of the listed tasks. The results in this work package are milestone 1 ("Intermediate Tool Set") represented by deliverable D2.1 scheduled for the end of the reporting period. Task 2.4 is scheduled for the end of the 2nd and final year of the project.

2.1.3 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 is 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 will provide 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 is 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 for this work package in the first year was to make significant progress in the first 3 of the listed tasks. Task 3.1 was most critical to get the interaction going and therefore its corresponding deliverable D3.1 was scheduled for month 6.

2.2 Work progress and achievements during the period

2.2.1 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. Work has begun and working prototypes exist which demonstrate the growing integration between the separate tool sets of the project partners. All enhancements are either already part of the latest public releases of the software packages, or at least are available to the other partners within the project as prototype deliverable D2.1 ("Intermediate Tool Set"). The milestone 1 ("Intermediate Tool Set") has been reached as scheduled.

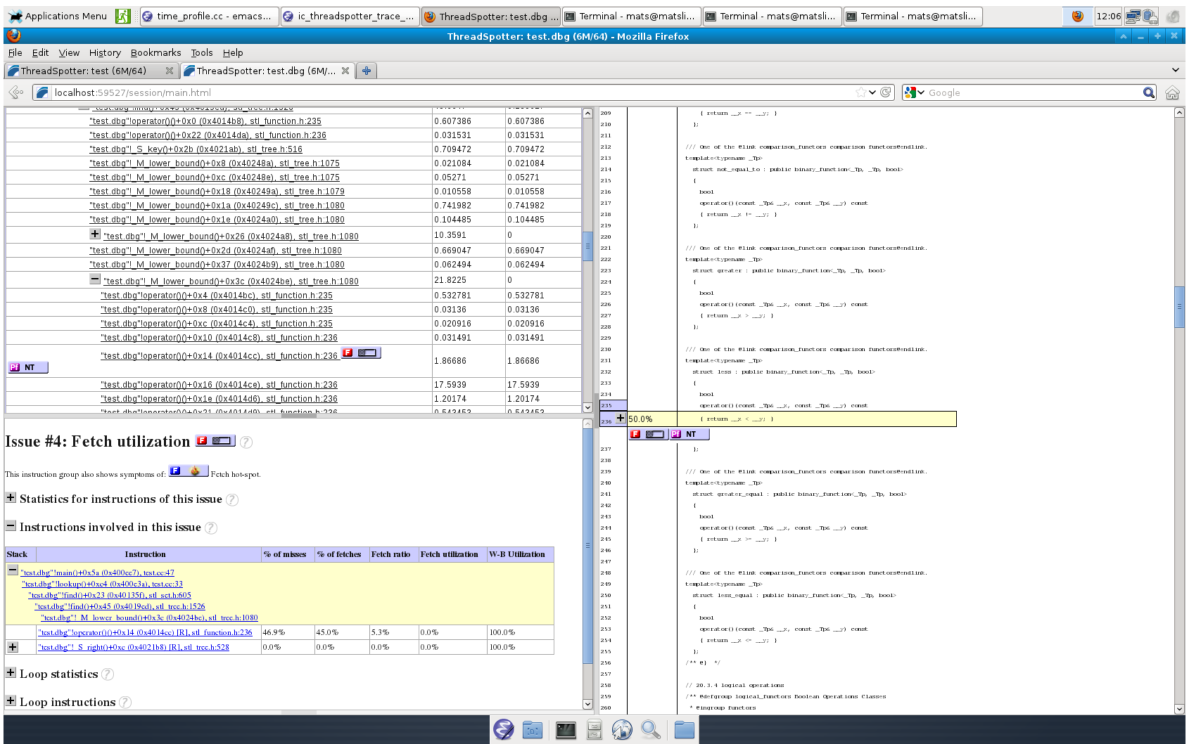
For a more detailed description of the HOPSA-EU toolset please see the deliverable D2.1.

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

- Rogue Wave's objective is to extend ThreadSpotter to collect and analyze time profile-based performance data for a shared-memory process, such as an MPI rank. Further, they plan to design a new presentation view of this data, for example a new tab entry in the ThreadSpotter view, to present this analysis and integrate with the existing issues-based (bandwidth, latency, inter-thread and pollution issues) and loop-based views. So far Rogue Wave has explored different technologies for collation of the new runtime data for the time-based view of the execution. A prototype has been implemented that collects instruction address and call stack information. The solution is interrupt-driven, instead of relying on the ThreadSpotter instrumentation, and will result in a low-overhead implementation. Different options for automatic phase detection have also been explored.

- Rogue Wave has developed an early prototype extending 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.


Applications Menu
time_profile.cc - emacs...
ic_threadspotter_trace_...
ThreadSpotter: test.dbg ...
Terminal - mats@matsli...
Terminal - mats@matsli...
Terminal - mats@matsli...
12:06
ThreadSpotter: test.dbg (6M/64) - Mozilla Firefox
File Edit View History Bookmarks Tools Help
ThreadSpotter: test (6M/64)
ThreadSpotter: test.dbg (6M/...
localhost:59527/session/main.html
Google
Pf NT
F

"test.dbg"!operator()()+0x0 (0x4014b8), stl_function.h:235	0.607386	0.607386
"test.dbg"!operator()()+0x22 (0x4014da), stl_function.h:236	0.031531	0.031531
"test.dbg"!_S_key()+0x2b (0x4021ab), stl_tree.h:516	0.709472	0.709472
"test.dbg"!_M_lower_bound()+0x8 (0x40248a), stl_tree.h:1075	0.021084	0.021084
"test.dbg"!_M_lower_bound()+0xc (0x40248e), stl_tree.h:1075	0.05271	0.05271
"test.dbg"!_M_lower_bound()+0x18 (0x40249a), stl_tree.h:1079	0.010558	0.010558
"test.dbg"!_M_lower_bound()+0x1a (0x40249c), stl_tree.h:1080	0.741982	0.741982
"test.dbg"!_M_lower_bound()+0x1e (0x4024a0), stl_tree.h:1080	0.104485	0.104485
+ "test.dbg"!_M_lower_bound()+0x26 (0x4024a8), stl_tree.h:1080	10.3591	0
"test.dbg"!_M_lower_bound()+0x2d (0x4024af), stl_tree.h:1080	0.669047	0.669047
"test.dbg"!_M_lower_bound()+0x37 (0x4024b9), stl_tree.h:1080	0.062494	0.062494
− "test.dbg"!_M_lower_bound()+0x3c (0x4024be), stl_tree.h:1080	21.8225	0
"test.dbg"!operator()()+0x4 (0x4014bc), stl_function.h:235	0.532781	0.532781
"test.dbg"!operator()()+0x8 (0x4014c0), stl_function.h:235	0.03136	0.03136
"test.dbg"!operator()()+0xc (0x4014c4), stl_function.h:235	0.020916	0.020916
"test.dbg"!operator()()+0x10 (0x4014c8), stl_function.h:236	0.031491	0.031491
"test.dbg"!operator()()+0x14 (0x4014cc), stl_function.h:236	1.86686	1.86686
"test.dbg"!operator()()+0x16 (0x4014ce), stl_function.h:236	17.5939	17.5939
"test.dbg"!operator()()+0x1e (0x4014d6), stl_function.h:236	1.20174	1.20174

Issue #4: Fetch utilization

This instruction group also shows symptoms of: Fetch hot-spot.

+ Statistics for instructions of this issue

− Instructions involved in this issue

Stack	Instruction	% of misses	% of fetches	Fetch ratio	Fetch utilization	W-B Utilization
− "test.dbg"!main()+0x5a (0x400cc7), test.cc:47 "test.dbg"!lookup()+0xc4 (0x400c3a), test.cc:33 "test.dbg"!find()+0x23 (0x40135f), stl_set.h:605 "test.dbg"!find()+0x45 (0x4019cd), stl_tree.h:1526 "test.dbg"!_M_lower_bound()+0x3c (0x4024be), stl_tree.h:1080						
	"test.dbg"!operator()()+0x14 (0x4014cc) [R], stl_function.h:236	46.9%	45.0%	5.3%	0.0%	100.0%
+	"test.dbg"!_S_right()+0xc (0x4021b8) [R], stl_tree.h:528	0.0%	0.0%	0.0%	0.0%	100.0%

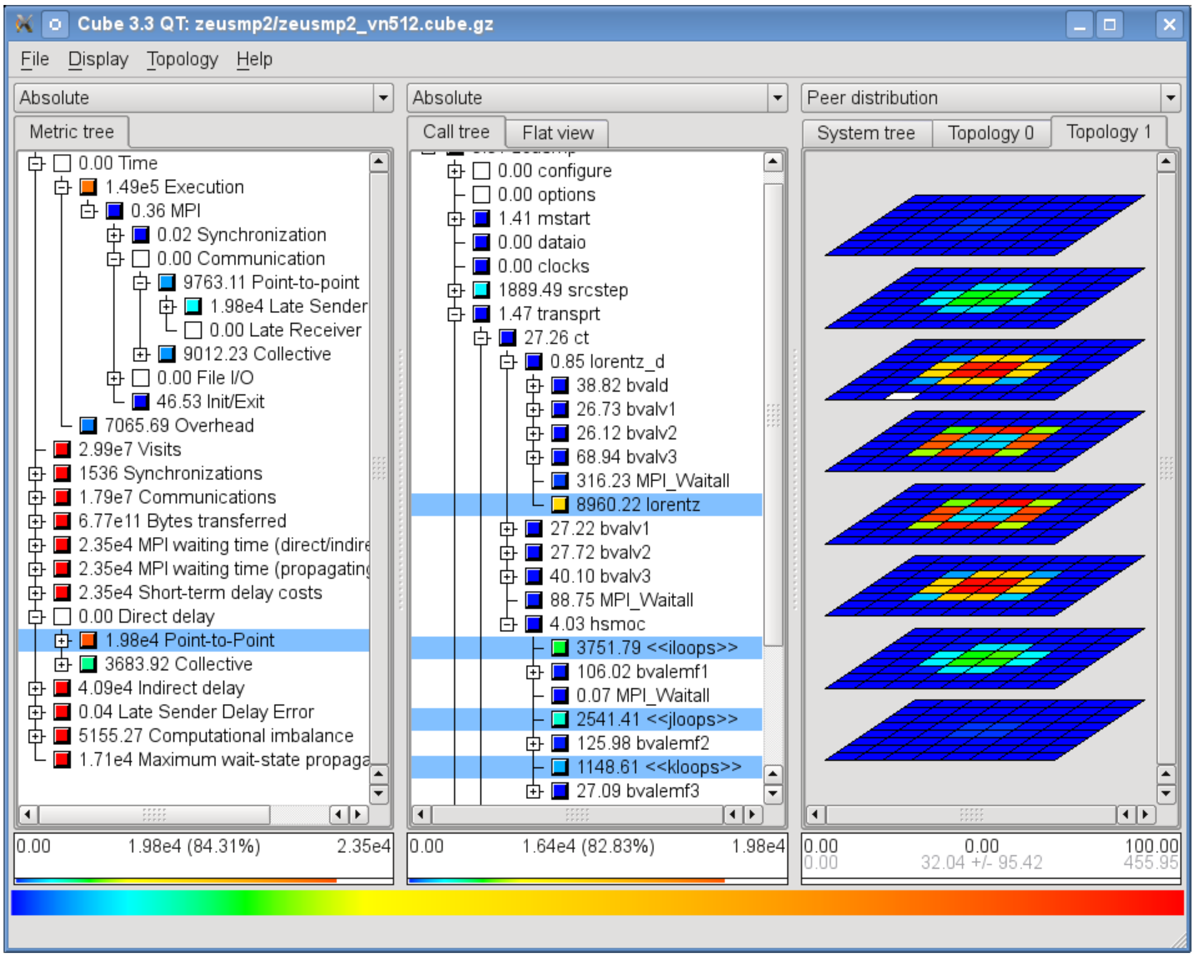
+ Loop statistics

+ Loop instructions

```
        { return __x == __y; }
      };

  /// One of the @link comparison_functors comparison functors@endlink.
  template<typename _Tp>
    struct not_equal_to : public binary_function<_Tp, _Tp, bool>
    {
      bool
      operator()(const _Tp& __x, const _Tp& __y) const
      { return __x != __y; }
    };

  /// One of the @link comparison_functors comparison functors@endlink.
  template<typename _Tp>
    struct greater : public binary_function<_Tp, _Tp, bool>
    {
      bool
      operator()(const _Tp& __x, const _Tp& __y) const
      { return __x > __y; }
    };

  /// One of the @link comparison_functors comparison functors@endlink.
  template<typename _Tp>
    struct less : public binary_function<_Tp, _Tp, bool>
    {
      bool
      operator()(const _Tp& __x, const _Tp& __y) const
50.0% { return __x < __y; }
    };

  /// One of the @link comparison_functors comparison functors@endlink.
  template<typename _Tp>
    struct greater_equal : public binary_function<_Tp, _Tp, bool>
    {
      bool
      operator()(const _Tp& __x, const _Tp& __y) const
      { return __x >= __y; }
    };

  /// One of the @link comparison_functors comparison functors@endlink.
  template<typename _Tp>
    struct less_equal : public binary_function<_Tp, _Tp, bool>
    {
      bool
      operator()(const _Tp& __x, const _Tp& __y) const
      { return __x <= __y; }
    };
  /** @} */

  // 20.3.4 logical operations
  /** @defgroup logical_functors Boolean Operations Classes
   * @ingroup functors
```

Figure 3. The new prototype of ThreadSpotter including time-based sampling

- JSC and GRS worked on the design and implementation of a so-called root-cause analysis for Scalasca. So far, the tool identified where wait-states triggered by inefficient usage of parallel programming constructs (MPI, OpenMP) occurred. In contrast, the new root-cause analysis identifies the original regions in the program which introduce delays (e.g., by bad load balance) that lead to wait-states at subsequent synchronization points. Incorporating long-distance effects through wait-state propagation, our method assigns costs in terms of the overall waiting time incurred to the process and source-code location where they were caused.



```
Cube 3.3 QT: zeusmp2/zeusmp2_vn512.cube.gz
File  Display  Topology  Help

Absolute                          Absolute                          Peer distribution
Metric tree                       Call tree | Flat view             System tree | Topology 0 | Topology 1

0.00 Time                         0.00 configure
  1.49e5 Execution                0.00 options
    0.36 MPI                      1.41 mstart
      0.02 Synchronization        0.00 dataio
      0.00 Communication          0.00 clocks
        9763.11 Point-to-point    1889.49 srcstep
          1.98e4 Late Sender      1.47 transprt
          0.00 Late Receiver        27.26 ct
        9012.23 Collective            0.85 lorentz_d
      0.00 File I/O                     38.82 bvald
      46.53 Init/Exit                   26.73 bvalv1
    7065.69 Overhead                    26.12 bvalv2
2.99e7 Visits                           68.94 bvalv3
1536 Synchronizations                   316.23 MPI_Waitall
1.79e7 Communications                   8960.22 lorentz
6.77e11 Bytes transferred             27.22 bvalv1
2.35e4 MPI waiting time (direct/indire 27.72 bvalv2
2.35e4 MPI waiting time (propagating  40.10 bvalv3
2.35e4 Short-term delay costs         88.75 MPI_Waitall
0.00 Direct delay                     4.03 hsmoc
  1.98e4 Point-to-Point                 3751.79 <<iloops>>
  3683.92 Collective                    106.02 bvalemf1
4.09e4 Indirect delay                   0.07 MPI_Waitall
0.04 Late Sender Delay Error            2541.41 <<jloops>>
5155.27 Computational imbalance         125.98 bvalemf2
1.71e4 Maximum wait-state propaga       1148.61 <<kloops>>
                                        27.09 bvalemf3

0.00   1.98e4 (84.31%)   2.35e4   0.00   1.64e4 (82.83%)   1.98e4   0.00   0.00              100.00
                                                                    0.00   32.04 +/- 95.42   455.95
```

Figure 4. Scalasca result browser display.

The screenshot in Figure 4 shows how the root-cause analysis of the ZeusMP/2 code identifies imbalanced functions, where delayed processes that are located around the centre of the applications' virtual 3D topology cause wait states later on.

A working prototype for MPI point-to-point and collective communication has been completed. It implements the root-cause analysis for wait states incurred in send and receive operations ("Late Sender" and "Late Receiver" wait states) as well as in collective broadcast, scatter/gather, barrier, and all-to-all operations. Using Scalasca's scalable event-trace replay approach, our prototype shows exceptional scalability: in a scaling study with the Sweep3D benchmark, it has successfully completed a root-cause analysis on traces from up to 262,144 MPI processes in less than two minutes [8].

An extended prototype for basic OpenMP parallelism (limited to a simple fork-join model with a fixed number of threads) has also been implemented. This prototype supports pure OpenMP and hybrid MPI/OpenMP programs, and facilitates the analysis of wait-state propagation from MPI to OpenMP synchronization points and vice versa.

- BSC's Paraver is a tool without semantics and programmable by the user. This flexibility provides a lot of power for the analysis but on the other hand it is more difficult to use than other tools with less potential. For this reason, BSC considered it important to focus on functionality enhancements of Paraver for this first year to implement mechanisms to facilitate that new users get familiar with the tool. The two main developments have been:
  - To implement a new mode for the configuration files that abstracts from the Paraver model and parameters and offers an interface easy to use where the options provided to the user are limited and renamed to be more descriptive.

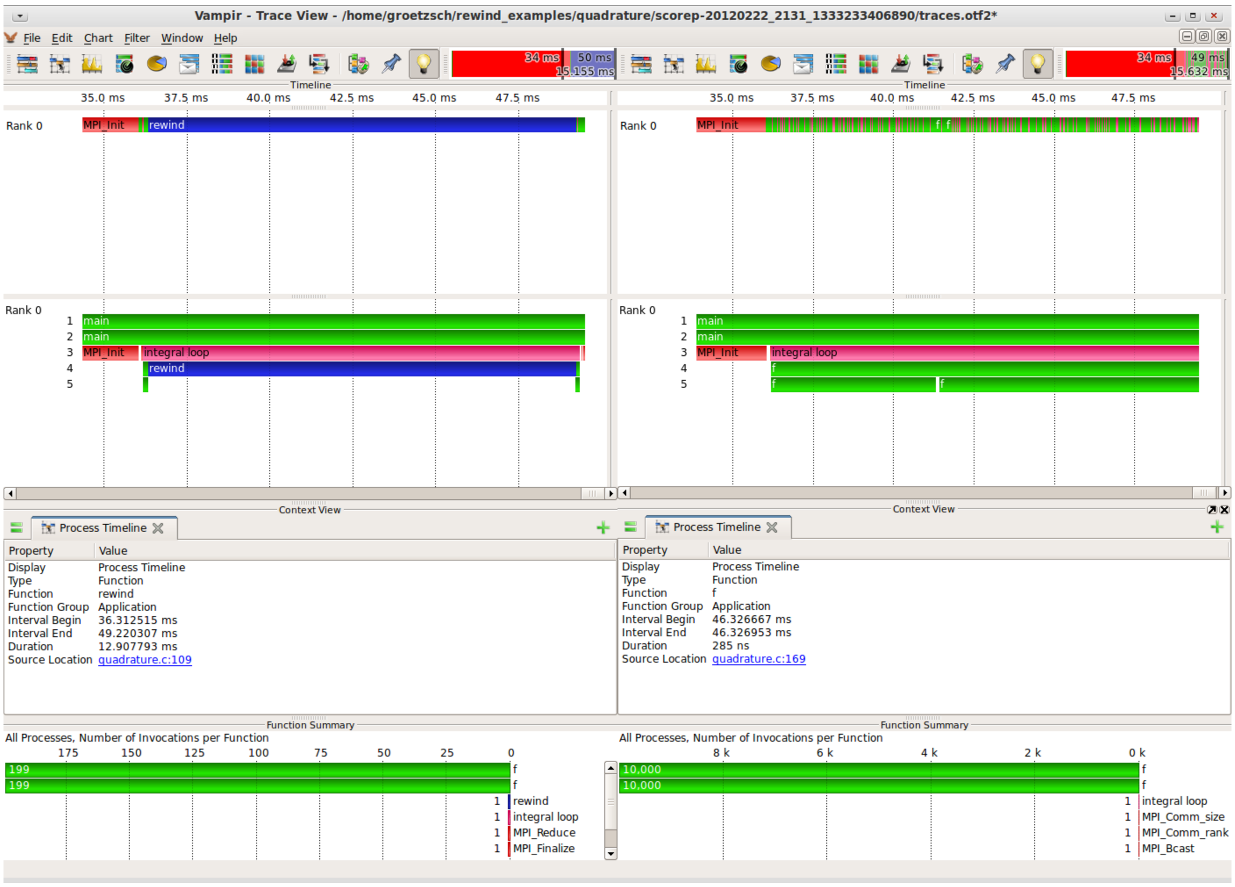
- To include an on-line mode to use the Paraver tutorials facilitating the users to follow the tutorial steps within the Paraver graphical interface. References to trace files and configurations are links that can be clicked to load the corresponding file.

- Also, BSC extended its instrumentation and measurement system Extrae to support new programming models. It has been extended to support CUDA instrumentation as well as the new OpenMP runtime for the Intel compiler (versions above 11.0)

- BSC's performance prediction tool Dimemas has been extended to include a prototype to simulate the StarSs programming model including a new scheduler and the internal communication and synchronizations from the StarSs runtime.

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 Sweep3D benchmark on 294,912 cores on the Jugene IBM BlueGene/P system at Jülich Supercomputing Centre. 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. 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.



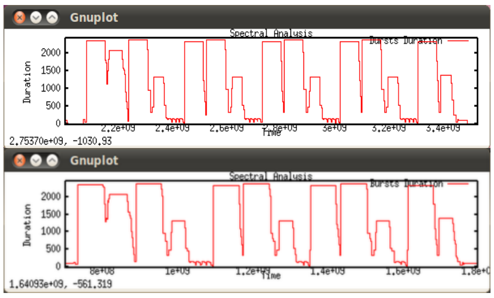
Property	Value
Display	Process Timeline
Type	Function
Function	rewind
Function Group	Application
Interval Begin	36.312515 ms
Interval End	49.220307 ms
Duration	12.907793 ms
Source Location	quadrature.c:109

Property	Value
Display	Process Timeline
Type	Function
Function	f
Function Group	Application
Interval Begin	46.326667 ms
Interval End	46.326953 ms
Duration	285 ns
Source Location	quadrature.c:169

Figure 5. Comparison between trace with (left) and without (right) rewind.

- TUD enhanced the Score-P monitoring component. The enhancement is a so-called rewind functionality, which means that it will allow discarding the preceding section of the event trace at certain control points. The implementation is based on the available region definition SCOREP_USER_REGION_[DEFINE|BEGIN|END]. The Score-P User API was extended with the function SCOREP_USER_REWIND_DEFINE to define a rewind point. The function SCOREP_USER_REWIND_POINT defines a control point in the trace and the function SCOREP_USER_REWIND_CHECK defines the next control point, the end of the rewind region. Based on a parameter for this function, the monitoring component will discard the rewind region in the trace buffer or will keep all data for this region. This makes the live decision, whether to keep or discard a rewind region, available to the user. Each rewind region has a name, so that nested rewind regions are available. For example, if two rewind regions are defined and the region 2 is inside of region 1 (point 1 → point 2 → check 2 → check 1), then a rewind of region 1 also deletes the recorded information inside of region 2, even if the check for region 2 did not result in a rewind. The implementation can also handle dispositions like point 1 → point 2 → check 1 → check 2, which means that a rewind of region 1 would delete the rewind point 2 and the positive check of rewind point 2 has no effects on the trace buffer, so that all traced information between check 1 and check 2 would stay in the trace. In case of a trace buffer flush, all defined rewind points are deleted and following checks have no effect. Rewind regions are only present in the trace buffer as regions if the check was positive and parts of the trace buffer were deleted. This feature improves storing or analyzing traces of long running applications that would otherwise generate too much trace information. For example, an application can decide whether to keep or discard the events generated by a computation iteration based on specific characteristics of that iteration e.g., computation results. This is in contrast to disabling / enabling tracing where the decision has to be made before the iteration.

- For the instrumentation and measurement system Extrae, BSC implemented a very preliminary prototype that determines which information is dumped on the trace file based on an on-line automatic analysis of the collected data. The current prototype integrates both the clustering and the time analysis modules and a paper was presented at ICPADS 2011 conference [7]. The new module looks for repetitive patterns across periodic intervals of the execution and automatically selects small, representative regions that describe the application phase behaviour. If the application enters a new computing phase, the tool will detect the change and will then trace another representative region for this new phase. Figure 6 shows one example of such analysis visualizing the periodicity of the signals used and the identification of two different program phases.



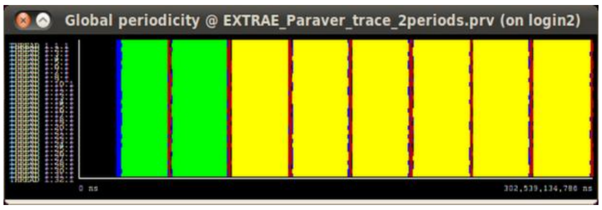
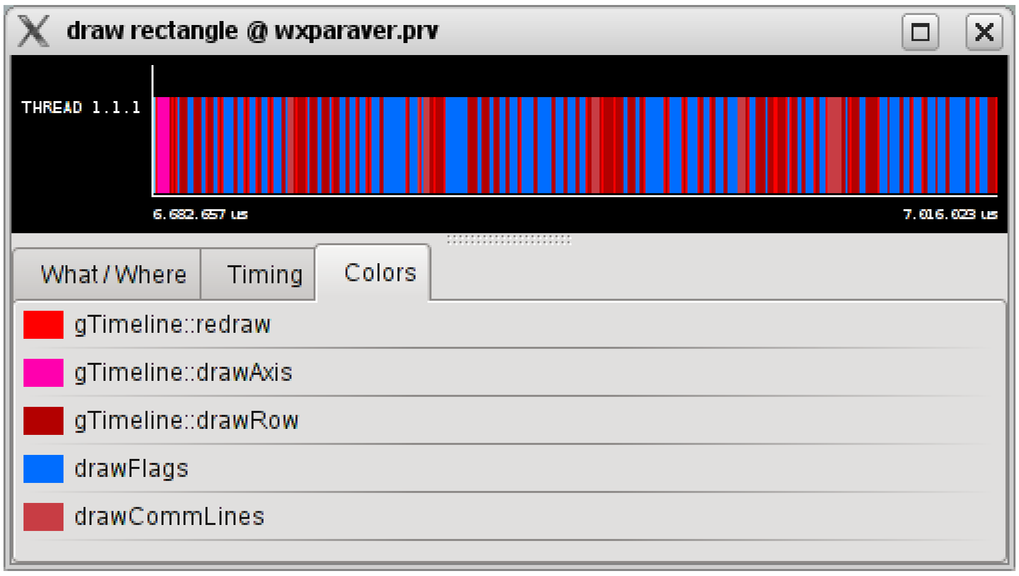

Global periodicity @ EXTRAE_Paraver_trace_2periods.prv (on login2)
0 ns
302,539,134,786 ns

Figure 6. Extrae on-line time analysis. Left side images display the signals computed while the right side shows the Paraver view colouring the different regions identified

- BSC also modified its performance prediction tool Dimemas to allow simulations of tens of thousands processes and threads. This required modifications on the tool structures and re-implementing the input trace module as well as the Paraver trace generation module. The first prototype is ready and it is currently being tested.

- Finally in the case of BSC's Paraver, the objective is to parallelize the computation kernel. The programming model StarSs was selected for this task. BSC had to devote a huge effort to be able to compile the sources with the Mercurium compiler of StarSs. The first target for the parallelization is the computation of the timeline. We obtained Paraver traces from Paraver executions with Extrae to analyze the potential impact of the proposed parallelization approach. Figure 7 shows a timeline view of the Paraver execution phases to compute the timeline data. The first OMPSs tasks have been generated but due to some problems with the runtime we have no results yet.



🟥 gTimeline::redraw
🟪 gTimeline::drawAxis
🟥 gTimeline::drawRow
🟦 drawFlags
🟥 drawCommLines

Figure 7. Timeline of the Paraver execution to compute the timeline data

- 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. Moreover, the new scheme was already integrated into the upcoming new Score-P measurement system [5, 6].

- TUD has worked on improvements of its performance analysis tool-chain. Several basic enhancements have been implemented in Score-P and its trace format OTF2 [4] in order to increase the scalability of the tool-chain. Partial trace loading contributes to enable the analysis of large traces generated by many processes running over a long period of time. Previously, analysis was limited by the need to fit the entire trace into (distributed) memory of the analysis processes. 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. Partial loading in Vampir will be using this information. It is currently implemented in a prototype that supports OTF1 and will be ported to OTF2.

Task 2.3 Tool integration

The objective of this task is to investigate in how much the existing HOPSA-EU tools can be integrated to work as a coherent performance measurement and analysis tool set.

- One envisioned scenario is to allow launching external tools from the Scalasca result browser CUBE. As a first step, it will be used to show the most severe instances of inefficiency patterns found by the Scalasca trace analysis in timeline displays of the trace visualizers Paraver and Vampir. 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. Work on adding support for OpenMP patterns has begun.

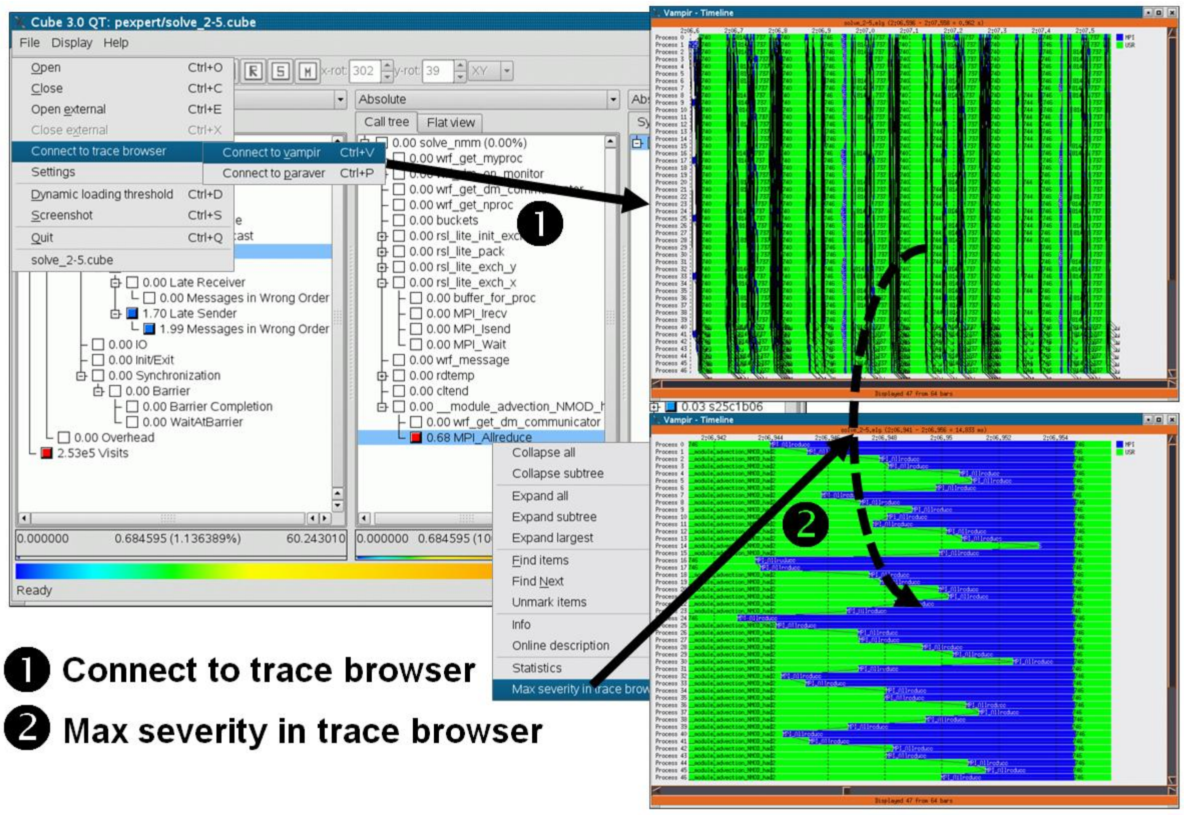


Figure 8. Scalasca →Vampir or Paraver Trace browser integration. In a first step, when the user requests to connect to a trace browser, the selected visualizer is automatically started and the event trace which was the basis of Scalasca's trace analysis is loaded. Now, in a second step, the user can 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.

To enable the necessary remote control of Vampir from inside CUBE, TUD implemented a remote control API via D-Bus commands. A prototype for the remote control existed for Scalasca and an old version of Vampir (based on Motif). The prototype was now also implemented in the latest version of Vampir (based on QT). The functionality includes the launch and termination of Vampir as well as the loading of trace files. Additional functions for a loaded trace are - among others - the zoom of a timeline view to a defined region in the trace. The controller (e.g., Scalasca) can open/close displays, can set processes in the displays which support this functionality, and can set counters in two displays, named Performance Radar and Counter Data Timeline. The controller is also able to request information from Vampir via the D-Bus interface, e.g., list open traces, open displays, available processes and counters.

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.

- In a second step, a more generic mechanism which would allow launching arbitrary tools (beyond the current hard-coded integration with Vampir and Paraver) was extensively discussed during the project meeting in Stockholm. BSC developed a prototype to integrate the sampled call stack analysis with CUBE. This prototype generates CUBE input from files, correlating the sources with the folded hardware performance counter metrics. The final goal is to launch gnuplot from CUBE to display the time evolution once its generic tool launch interface is available.
- Rogue Wave will investigate integration between timeline-based ThreadSpotter and the other European tools. So far, the architecture and interfaces for such integration has been discussed at the two meetings.

- We have also been working to understand and evaluate how we can apply OTF2 to BSC's Paraver trace files. As the end of the development for Score-P was scheduled for the end of 2011, we decided to wait until Score-P is available to make the final decisions. In the meantime we discussed internally about what are the potential weak points for us to use OTF2 and we prepared a list of topics to discuss with the developers of OTF2 during the next project meeting by the end of February.

Task 2.4 Tool validation

- No work was planned for this task for the reported period.

2.2.2 WP3: Integration of system and application performance analysis

Considerable progress has been made to integrate the software for system-level monitoring and analysis from our Russian HOPSA-RU partners with the HOPSA-EU tool set from WP2. Work has begun to define and further refine the overall performance-analysis workflow implementing the overall objective of the HOPSA project. 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"). For both Paraver/Extrae and Vampir/Score-P tool sets, an early prototype exists where the performance analysis results and visualizations are enriched with system-level sensor data. A prototype of the central HOPSA component, the so-called light-weight monitoring module (LWM2), which measures and extracts an application-level performance signature and delivers it to the system-level cluster monitoring layer, was implemented and is currently tested at the various partner sites. Finally, progress was also made to provide an integrated package of all tools with consistent documentation and one unified installation mechanism called UNITE.

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

The Russian ClustrX management software provides node-level sensor information that can give additional insight for performance analysis of applications with respect to the specific system they are running on. Based on discussions at the international kick-off meeting at MSU, an API between measurement tools and the node agent of ClustrX and the request interface of LAPTA was defined. A list of node-level metrics which can be provided by the measurement tools and the node agent was prepared. BSC and TUD identified potential scenarios where trace visualizers like Paraver or Vampir can contribute. The two main approaches BSC and TUD are interested in are to populate the current traces with system information (the granularity will depend on the overhead to obtain the data) and to analyze them in respect to the system-wide performance. This has been documented in a report attached to this progress report (EU Deliverable D3.1 "Requirements for the Interface between System-level and Application-level Performance Analysis"). The second approach did not raise enough interest within the Russian partners and we decided to focus our effort on the population of OTF2/Paraver traces with system information.

In January 2012, we received access to graphit (a Russian partner's platform) where we tested the access to the system data. The LAPTA system offers two different ways to access to the collected data:

- Historic information is stored with a given granularity for all the sensors and all the IP (nodes) on the system. The initial granularity was very coarse (one minute) and did not seem useful for the population of application trace files because there can be many different program phases in a one minute interval. On the other hand, the circular buffer provides historical information with fine-grained detail (coarser or equal to 1sec depending on the sensor) for the last minutes (300 measurements).

- Streamed information can be requested for any range of sensors and IPs. The interface provides at least a value every 10 seconds unless there is a change greater than a 10%. The finest granularity seems to also be 1 second.

Both mechanisms use a connection through an HTTP protocol that in the case of the streamed data has to be refreshed periodically or dies after 5 minutes. We wanted to evaluate both alternatives to see their potential and identify possible drawbacks.

TUD has investigated on how to integrate historical system-level performance information into application traces. The investigation benefits from prior experience with the Dataheap architecture. Dataheap is a server that collects system-level performance counters into a central database. In VampirTrace this information can be integrated using a counter plug-in module. It has been identified that the Russian HOPSA infrastructure can provide similar features than Dataheap. The integration design for Score-P is based on requesting node level performance data (external metrics) at the end of the application measurement run, integrating it into the trace using the Score-P infrastructure. Several general enhancements to the Score-P infrastructure have been proposed and implemented. To ensure a correct correlation between the time of the system performance data and application events a synchronized global clock has to be available. For the temporal granularity of 1 second, an NTP correction is sufficient. Currently it is assumed that Score-P uses a global clock internally, which can be enforced by the configuration of Score-P. A translation functionality from a global time source (e.g., clock_gettime) to the local Score-P internal ticks (e.g., a cycle count using PAPI) will be added to Score-P. Vampir will be improved to correctly display and associate the performance information that is in the scope of system nodes rather than individual processes.

BSC started work evaluating the streamed system data approach. Due to the current response time required to request system data, we decided to implement a server daemon (per node) to act as an intermediary which request the data from LAPTA and provides it to the application instrumentation library. This daemon can become part of LAPTA, or can be removed if an efficient API to access the local data directly from the measurement system is provided. A very preliminary prototype has been implemented during the last month of the reported period allowing to run first tests that include system sensor information into Extrae trace files. Two scenarios have been implemented:

- A loop to mimic feeding back the trace file with fine grain system data.
- A coarse granularity approach adding system sensor data only at flushing time.

Task 3.2 Definition of the overall performance analysis workflow

The basic overall performance analysis workflow has been discussed at the various project meetings. The performance analysis workflow will consist of two basic steps. During the first step, we will identify all those applications running on the system that may suffer from inefficiencies. This is done via system-wide job screening supported by a light-weight measurement module dynamically linked to every executable. The screening output will identify potential problem areas such as communication or memory, and issue recommendations on which of the four diagnostic tools can be used to explore the issue further. In general, the workflow will successively narrow the analysis focus and collect performance data on an increasing level of detail. At the same time, the measurement configuration will be optimized to keep intrusion low and limit the amount of data collected. To distinguish between system and application-related performance problems, some of the tools will allow also system-level data to be retrieved and displayed (see Task 3.1).

Task 3.3 Light-weight monitoring module

GRS developed a working prototype of the Light Weight Measurement Module (LWM2), a performance monitoring module that can be applied to every job on a system with minimal overhead, in a transparent way using library preloading. It profiles the application based on a range of data sources: It implements the PMPI interface to profile MPI function calls, uses PAPI for collecting hardware counter data, instruments standard I/O library calls and collects basic CUDA profiling data. It also takes periodic samples of the application execution and categorizes the signals based on the activity the application is performing at that time, e.g., MPI function call or OpenMP parallel region. The profiler also divides the execution of an application into time slices, aggregating the profiled data for each slice, enabling the analysis of the application's time-varying behaviour. Also, as we collect data from every job running on the system, time slicing enables us to correlate performance phenomena that occurred at the same time, even in different jobs. At the end of the measurement of a job, all the profiled data is aggregated and a summary output is presented to the user.

The detailed measurement data for each time slice is planned to be collected through the ClustrX interface from T-Platforms and stored in the database provided by MSU, along with detailed system monitoring data.

The protocol for the communication between ClustrX and LWM2 is close to finalization, and integration is expected to commence in the near future.

Task 3.4 Unified download, configuration, build and installation package

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 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. JSC's 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 standardized, well-tested tool configurations,
- but still being flexible enough to be able to coexist with site-local installations, restrictions, and policies.

Work on a ”meta”-installation tool continued by JSC which is capable of configuring, building, and installing all HOPSA tools as a common package while hiding tool-specific aspects of the various phases. Support for Extrae/Paraver and TAU has been added. The package was intensively tested at a VI-HPS tuning workshop in Aachen in September 2011 and is in use on JSC's production clusters. A first public release package of UNITE is planned for early 2012.

2.3 Project management during the period

Despite the unusual setup (coordination of two independently proposed, funded, and managed projects) and time constraints (due to the Russian restriction that projects are aligned with calendar years, HOPSA-EU had to start working without EU grant agreement and funding in place), we managed to start-up the project in time and to initiate project work both between the EU project partners and with our HOPSA-RU partners. A project website has been created (***http://www.hopsa-project.eu***).

2.3.1 WP1 Project management

Task 1.1 Administrative and financial management

- The negotiation meeting with the EC and the EU coordinator JSC has been done on March 15th, 2011 in Brussels.
- The Grant Agreement (contract) with the European commission (EC) was signed by the EC and the coordinator in June 2011.
- The Coordination Agreement (CooA) between all EU and Russian partners has been signed by all partners in June 2011.
- As required by EU regulation, also the Consortium Agreement (CA) between all EU partners has been signed by all partners in June 2011.

Task 1.2 Technical coordination

- A project wiki, a source-code revision control server (SVN), and a mailing list (***hopsa@fz-juelich.de***) have been created at JSC to facilitate work in the project.
- A kick-off meeting between the EU partners has been organized and hosted by JSC on March 3rd, 2011. All partners participated.
- An international kick-off meeting between all (EU and Russian) project partners has been organized by JSC and MSU and was hosted by MSU on May 25th and 26th, 2011. All partners participated.
- A project meeting between the EU partners has been organized and hosted by Rogue Wave Software AB in Stockholm on September 7th, 2011. All partners participated.
- An international project meeting between all (EU and Russian) project partners was prepared by JSC and MSU and was hosted by BSC on November 2nd and 3rd, 2011 in Barcelona. All partners participated.

2.3.2 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 in detail:

General dissemination actions

- 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.

- A two-page fact sheet was prepared summarizing the project. It is available at the project website.
- 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/faster-computations-with-hopsa.html***

  The press release got picked up by many other sides including HPCwire, InsideHPC, Primeur, Scientific Computing, and many more.

Dissemination at international HPC conferences and workshops

- Rogue Wave presented "How to port applications to new architectures" at the ***3rd PRACE Industrial Seminar***, Stockholm, Sweden, on March 29th, 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.
- BSC, JSC, and TUD presented "Score-P - A Joint Performance Measurement Run-time Infrastructure for Periscope, Scalasca, TAU, and Vampir" [1], "Trace-Based Performance Analysis of GPU Accelerated Applications" [2] and "Folding: Detailed Analysis with Coarse Sampling" [3] at the ***5th Parallel Tools Workshop***, HLRS/TUD, Dresden on Sep 26th to 27th, 2011.
- The project was also 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.

HPC training events

- JSC presented "Scalable Performance Analysis Tools for HPC Applications" at a ***HPC tools training at ETH Zurich***, Switzerland on January 17th to 18th, 2011.
- JSC presented a "Scalasca and Vampir Introduction" at the ***LinkSCEEM-2 Training Workshop, CaSToRC***, The Cyprus Institute, Nicosia, Cyprus on January 27th, 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.
- 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.
- 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.
- BSC presented Paraver and Dimemas at ***HPC-EUROPA2/TAM 2011***, Barcelona, Spain on June 8th to 9th, 2011.
- JSC gave an introduction into parallel performance analysis and presented the HOPSA tool set at the ***Russian Summer School on Supercompting Technologies*** at MSU, Moscow, Russia on July 1st, 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 DEISA/PRACE/TeraGrid ***2011 European-US Summer School on HPC Challenges in Computational Sciences*** at Lake Tahoe, USA on July 1st, 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.
- 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.
- BSC presented Paraver and Dimemas at the ***PRACE Autumn School 2011***, Bruyères-le-Châtel, France on October 25th to 27th, 2011.
- BSC presented Paraver and Dimemas at ***CAPAP-H winter seminar***, Valladolid, Spain on January 26th 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.

Refereed publications

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

[2] Guido Juckeland et al, Trace-Based Performance Analysis of GPU Accelerated Applications, *Proceedings of 5th Parallel Tools Workshop*, 2011, (to appear).

[3] H. Servat, G. Llort, J. Giménez, K. Huck, J. Labarta, Folding: detailed analysis with coarse sampling, *Proceedings of 5th Parallel Tools Workshop*, 2011, (to appear).

[4] D.Eschweiler, M. Wagner, M. Geimer, A. Knüpfer, W.E. Nagel, F. Wolf, Open Trace Format 2 - The Next Generation of Scalable Trace Formats and Support Libraries, *Proceedings of the International Conference on Parallel Computing (ParCo 2011)*, Ghent, Belgium, August 30th, 2011. (to appear)

[5] Markus Geimer, Marc-André Hermanns, Christian Siebert, Felix Wolf, Brian J. N. Wylie, Scaling Performance Tool MPI Communicator Management, *Proceedings of the 18th European MPI Users' Group Meeting (EuroMPI)*, Santorini, Greece, volume 6960 of Lecture Notes in Computer Science, pages 178-187, Springer, September 2011.

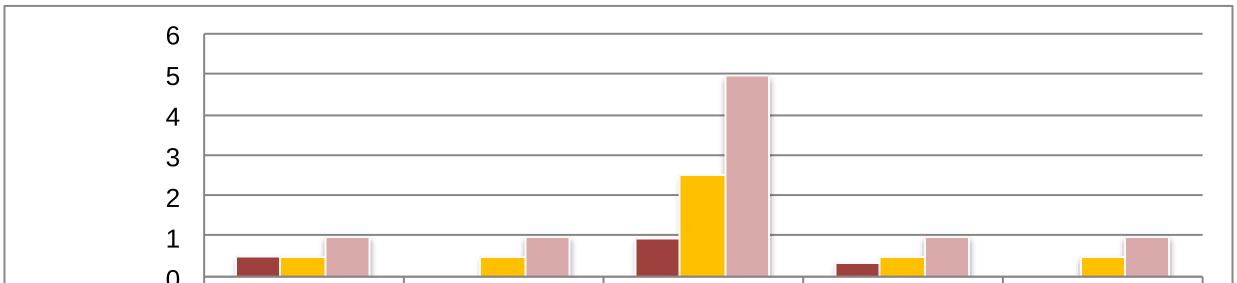
[6] M. Geimer, P. Saviankou, A. Strube, Z. Szebenyi, F. Wolf, B. J. N. Wylie, Further improving the scalability of the Scalasca toolset. *Proceedings of PARA 2010: State of the Art in Scientific and Parallel Computing, Part II: Minisymposium Scalable tools for High Performance Computing, Reykjavik, Iceland, June 6–9 2010*, volume 7134 of *Lecture Notes in Computer Science*, pages 463-474, Springer, 2012.

[7] G. Llort, G., et al. Trace Spectral Analysis toward Dynamic Levels of Detail. 17th IEEE International Conference on Parallel and Distributed Systems, ICPADS 2011, Tainan, Taiwan 332 - 339 (2011).

[8] D. Böhme, B. R. deSupinski, M. Geimer, M. Schulz, and F. Wolf. Scalable critical-path based performance analysis. *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS)*, Shanghai, China. IEEE Computer Society, May 2012. (accepted for publication)

2.4 Explanation of the use of the resources

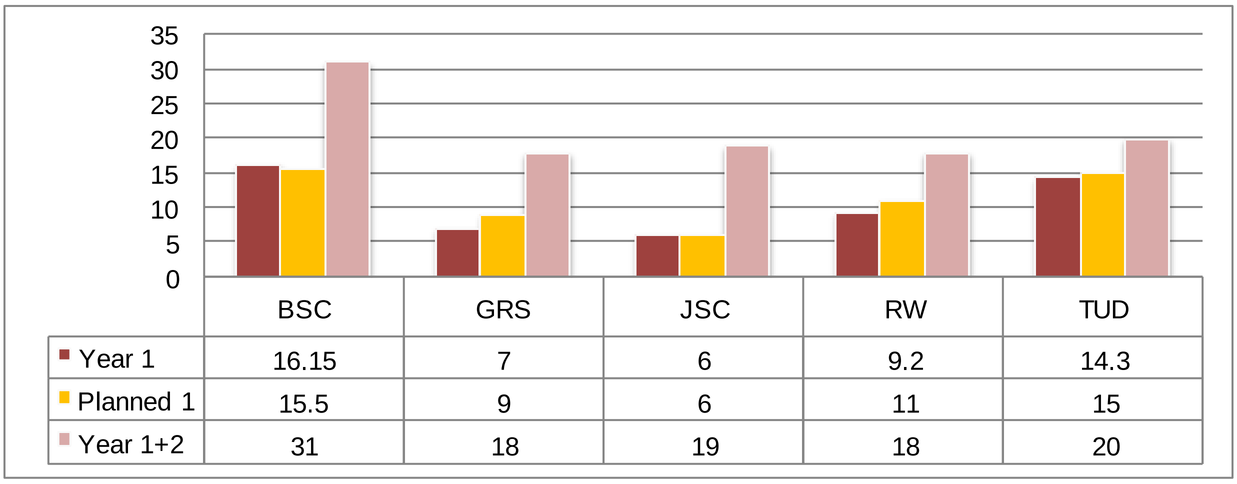
Despite the unusual setup (coordination of two independently proposed, funded, and managed projects) and time constraints (due to the Russian restriction that projects are aligned with calendar years, HOPSA-EU had to start working without EU grant agreement and funding in place), we managed to start-up the project in time and to initiate project work both between the EU project partners and with our HOPSA-RU 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
🟨 Planned 1	0.5	0.5	2.5	0.5	0.5
🟪 Year 1+2	1	1	5	1	1

Figure 9. *Total Efforts (PM): WP1 (MGMT)*

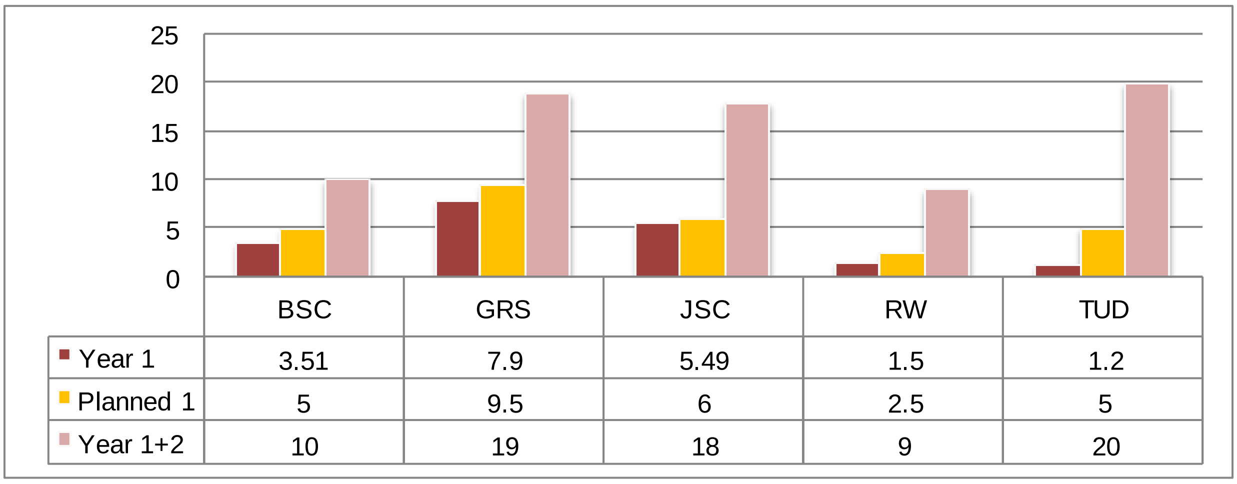
There is very little effort planned for work package 1 (“Project management”). Figure 9 shows the total efforts (in person months) related to WP1. BSC and RW are roughly in plan. The project principal investigators of GRS, JSC, and TUD were not able to charge all of their efforts to the HOPSA project, although never-the-less the necessary work was done.



	BSC	GRS	JSC	RW	TUD
🟥 Year 1	16.15	7	6	9.2	14.3
🟨 Planned 1	15.5	9	6	11	15
▨ Year 1+2	31	18	19	18	20

Figure 10. *Total Efforts (PM): WP2 (RTD)*

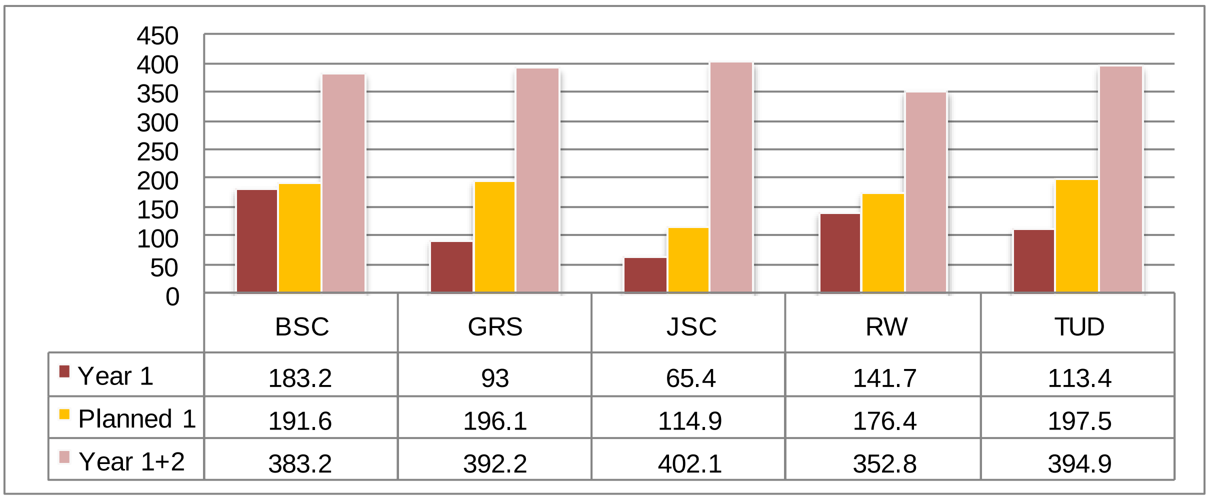
Figure 10 shows the total efforts (in person months) related to work package 2 (“HPC application-level performance analysis”). Efforts in research and development for WP2 were mostly spent as planned. Only GRS and TUD are slightly behind due to delays in hiring appropriate people at the start of the project. Provisions are in place to catch-up in the second period of the project.



	BSC	GRS	JSC	RW	TUD
🟥 Year 1	3.51	7.9	5.49	1.5	1.2
🟨 Planned 1	5	9.5	6	2.5	5
▨ Year 1+2	10	19	18	9	20

Figure 11. *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”). Efforts in research and development for WP3 were also mostly spent as planned although to a lesser degree than in WP2 due to the difficulty to get the necessary access to the development platforms of our Russian partners. This affected mostly BSC and TUD. Meanwhile, access to the Russian systems is established (January 2012) and provisions are in place to catch-up in the second period of the project.



	BSC	GRS	JSC	RW	TUD
🟥 Year 1	183.2	93	65.4	141.7	113.4
🟨 Planned 1	191.6	196.1	114.9	176.4	197.5
🟪 Year 1+2	383.2	392.2	402.1	352.8	394.9

Figure 12. *Total Costs (k€)*

Finally, Figure 12 provides an 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. The numbers are lower than planned 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).

3. Deliverables and milestones tables

Deliverables

TABLE 1A. DELIVERABLES 1ST YEAR

Del. no.	Deliverable name	Version	WP no.	Lead beneficiary	Nature	Dissemi-nation 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	submitted	yes	In time Renamed from "Intermediate Activity Report"
D2.1	Intermediate Tool Set	1	WP2	BSC	P	PU	M12	24/02/2012	submitted	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	submitted	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	Dissemi-nation 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				
D2.2	Final Tool Set	1	WP2	TUD	P	PU	M21				
D2.3	Tool Validation Report	1	WP2	TUD	R	PP	M23				
D3.2	Workflow Report	1	WP3	RW	R	PU	M15				
D3.3	Light-weight Monitoring Module	1	WP3	GRS	P	PU	M18				
D3.4	UNITE Package	1	WP3	JSC	P	PU	M22				Lead changed from GRS

Milestone s

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			

4. Annex: Use of Resources Tables from NEF