Report on the maturity evaluation of three software components and on potential centre model

CONTRACT NO EESI2 312478
INSTRUMENT CSA (Support and Collaborative Action)
THEMATIC INFRASTRUCTURE

Due date of deliverable: October 1st, 2015
Actual submission date: July 1st, 2015

Start date of project: 1 September 2013 Duration: 30 months

Name of lead contractor for this deliverable: Bernd Mohr (PRACE-JSC)
Name of reviewers for this deliverable: Christian Feld, Alexandre Strube (PRACE-JSC), Rosa Badia (PRACE-BSC)

Abstract: Report on the maturity evaluation of three software components and on potential centre model: This report documents the evaluation of the level of maturity of three selected Exascale software components following the methodology defined in EESI2 D6.1 and discusses issues and provides feedback regarding the evaluation methodology. Finally, it suggests a structure for a potential European Extreme-scale Software Centre (EESC).

Revision 1.0

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

Table of Contents

Glossary

Abbreviation / acronym	Description
EESC	European Extreme-scale Software Center
EESI	European Exascale Software Initiative
HPC	High Performance Computing
OMM	Open Maturity Model
OSS	Open source software
TRL	Technology Readiness Level

1. Executive Summary

One of the important recommendations of the first phase of the European Exascale Software Initiative (EESI1) was to establish a European Exascale Software Centre to coordinate research, development, testing, and validation of EU HPC Exascale software ecosystem components and modules. As a first step in this direction, the main charter of work package 6 of EESI2 was to investigate the feasibility and prepare for the operation of such a centre. To accomplish this, the objectives of WP 6 were:

- Develop and document a methodology for estimating the level of maturity of Exascale software components
- Identify 3 software stack components from existing and near future European Exascale projects and apply the defined methodology
- Examine existing “equivalent" centers and propose a structure adapted to Exascale software

This deliverable (D6.2) reports on the results on the latter two objectives as defined by WP6 Task 6.2 (“Perform evaluation on 3 components”). First, it documents the evaluation of the level of maturity of three selected Exascale software components following the methodology defined in EESI2 D6.1 and discuss issues and provides feedback regarding the evaluation methodology. Secondly, it recommends three actions including reinforcing the recommendation for the creation of a European Extreme-scale Software Centre (EESC). Finally, it suggests a structure and funding model for such a center.

2. Evaluation of Three EU Exascale Open-Source Components

To test the maturity evaluation procedure developed and documented in EESI2 Deliverable D6.1, three EU Exascale program components were selected. Then, each of the selected components was evaluated regarding the criteria defined in D6.1 by an HPC expert.

2.1 Rationale on the Selection of the Components

The criteria used for the selection of the components were

1. Used in EU FP7 (and H2020) Exascale projects
2. Mature established software packages
3. Usage beyond the software authors

In the end, the following three software components were selected:

- For Programming model implementations (PM), we selected OmpSs, the programming model developed and maintained by BSC. OmpSs is used in two of the three EU FP7 Exascale projects DEEP and Mont-Blanc (as well as in the follow-up projects DEEP-ER, Mont-Blanc-2 and Mont-Blanc 3) and is also proposed to be used in the OpenFET Flagship project Human Brain, and proposals recently accepted such as Exanode and Intertwine. It was also subject of the EU FP7 project TEXT.
- For Development Tools (DT), we evaluated the community performance instrumentation and measurement package Score-P developed by a German consortium led by Technical University Dresden, GRS Aachen, RWTH Aachen University, Technical University Munich, and Jülich Supercomputing Centre. Score-P is used in two of the three EU FP7 Exascale projects DEEP and Mont-Blanc, in the follow-up project Mont-Blanc-2, and is also proposed to be used in the OpenFET Flagship project Human brain. It was also subject of the EU FP7 project HOPSA, the EU ITEA2 project H4H and the German BMBF projects SILC, LMAC, and Score-E, and the DOE project PRIMA.
- As a third package, we used the communication library GASPI/GPI-2 developed by Fraunhofer which is part of the EU FP7 Exascale project EPIGRAM.

2.2 Basic Evaluation Procedure

Each of the selected components was evaluated regarding the criteria defined in D6.1 by an HPC expert. In the evaluation (sections 2.3, 2.4, and 2.5), the original criteria from D6.1 are shown in “black” and the answers / evaluation in “blue”. The evaluations of Score-P (2.3) was done by the developers of the components (“self-evaluation”), OmpSs (3.4) was externally evaluated but cross-checked for accuracy by the developers, and finally GASPI (2.5) was evaluated by a HPC expert of Jülich Supercomputing Centre with no connection to the project solely on publicly available sources (“external evaluation”). The evaluation was done in May 2015 and the answers reflect the status of the projects known at this point. As much as possible, the evaluation was based on publicly available information on the internet, and respective pointers to websites or documents used in the evaluation are documented in the evaluation.

2.3 Component 1: Performance Instrumentation and Measurement Package Score-P

Short description of the component: According to the project's website (http://www.score-p.org), Score-P is a highly scalable and easy-to-use tool suite for profiling, event tracing, and online analysis of HPC applications. It is jointly developed by the Jülich Supercomputing Centre, the German Research School for Simulation Sciences, Technische Universität Dresden, Technische Universität

München, University of Oregon, RWTH Aachen University, GNS (Gesellschaft für numerische Simulation mbH), and GWT-TUD GmbH (Gesellschaft für Wissens- und Technologie-Transfer), and it is funded by BMBF/Germany and DOE/USA.

Besides, Score-P sits under the umbrella of the Virtual Institute – High Productivity Supercomputing (VI-HPS). Its initial funding came from the Helmholtz Association of German Research Centers, and its partners are: Allinea Software Ltd., Barcelona Supercomputing Center, Forschungszentrum Jülich, German Research School for Simulation Sciences, Lawrence Livermore National Laboratory, RWTH Aachen University, TU Dresden, TU München, Université de Versailles St-Quentin-en-Yvelines, University of Oregon, Universität Stuttgart, and the University of Tennessee Knoxville

Score-P is the measurement infrastructure of choice for the analysis tools Periscope, Scalasca, Vampir and Tau. Its open-source nature allows one to develop further tools. Score-P integrates the Open Trace Format Version 2, the Cube profiling framework and the Opari2 source-to-source instrumenter.

2.3.1 Documentation and Support

Availability of Documentation

- Up-to-date installation guide: Score-P's website http://www.score-p.org/ provides a quick start guide, as well as documentation on how to instrument and measure applications. It further provides documentation on how to perform performance analysis with it.
- User guide: Can be found at http://www.score-p.org – in HTML or as a 163-page PDF. On the same page, HTML and PDF versions of user guides for sub-components OTF2 (920 Pages), OPARI2 (37 pages), and Cube (77 pages).
- FAQ: No
- Quick introduction (getting started): A quick start guide is available as HTML at https://silc.zih.tu-dresden.de/scorep-current/html/quickstart.html , and as a chapter of the PDF (6 pages long). Sub-components have their own "getting started" files: OTF2 https://silc.zih.tu-dresden.de/otf2-current/html/, OPARI2 https://silc.zih.tu-dresden.de/opari2-current/html/ and Cube http://www.scalasca.org/software/cube-4.x/documentation.html (PDF, 7 pages)
- Release notes: The "ChangeLog" file inside the distribution package contains release notes.
- Up-to-date training classes or courses : Tuning Workshops are actively promoted by the developers and were organized for several years now. There are three upcoming workshops at the time of this writing (May 5th), for the next two months:
  - The 18th VI-HPS Tuning Workshop, in Grenoble, France: http://www.vi-hps.org/training/tws/tw18.html
  - PATC-EPCC/DiRAC Performance Workshop (25-26 June 2015, Durham, England): https://events.prace-ri.eu/event/386/overview
  - ISC-HPC'15 Tutorial 06: Hands-on Practical Hybrid Parallel Application Performance Engineering (12 July 2015, Frankfurt, Germany) http://www.isc-events.com/isc15_ap/sessiondetails.htm?t=session&o=136&a=select&ra=index
- Up-to-date training materials, manuals, guidelines: All the material required for the courses (slides, Live-ISO disk image with the tools pre-installed) is available on Score-P's website. Videos can be found at http://supercomputing.cyi.ac.cy/index.php/improvement/performance-analysis
- Availability of professional (commercial, contracted) training: ParaTools (http://www.paratools.com/) and GWT (http://gwtonline.de/) cover Score-P in their commercial training for TAU and Vampir.
- Published books by persons not affiliated with the component. No
- Documentation available in multiple languages: The documentation is all in English. There is training material available in Chinese. Support can receive requests by e-mail in English and German. In each system, the responsible system administrators write their own documentation about using Score-P. For example, TSUBAME computing services of Tokyo Tech has instructions on Score-P in Japanese at http://tsubame.gsic.titech.ac.jp/en/node/1245
- Published procedure for user (feature) requests: Score-P's website directs the user to send an e-mail to support@score-p.org, and to sign-up for the Score-P's news mailing list at https://mailman.zih.tu-dresden.de/groups/listinfo/scorep-news
- Product examples or demos: At the page http://www.vi-hps.org/training/live-iso, it is possible

to download an ISO DVD image of a Linux distribution with Score-P pre-installed and with some examples of real runs.

- Published roadmap (future platforms and features): The roadmap is described at the paper “Scalasca v2: Back To The Future”, by Ilya Zhukov, Christian Feld, Markus Geimer, Michael Knobloch, Bernd Mohr and Pavel Saviankou, in: Proceedings of the 8th International Workshop on Parallel Tools for High Performance, Springer, 2015.

Support (bug fixing, help)

- Website, support mailing list, user forum, phone service: support is given in German and English by sending an e-mail directly to support@score-p.org. There is no user forum, and the mailing list is for news only.
- 24/7 support or best effort: Best effort. The e-mails are usually answered by the responsible of that section of the software.
- Usage of bug tracking and/or ticket systems: Score-P uses TRAC (http://trac.edgewall.org/) for its bug tracking and ticket system. It's only available internally for its own developments. The user is directed to send an e-mail to the aforementioned support address, and one developer will create the ticket, if that is the case.
- Availability of professional (commercial, contracted) support: No
- Average bug time solving: Bugs are added on the ticket system as soon as they are received, and being such a complex system that runs on many platforms, bug fixing times are can vary. Some bugs, for example, might only be considered “solved” when a fix on some related supercomputer is performed (say, a broken compiler, for instance). If the ticket happens to become a feature request, it might be held for the next release (see next item).
- Frequency of releases and patches: Score-P's main releases tend to happen twice a year, around February and August. This gives developers time to hone the software and fix bugs (by releasing minor versions, containing mostly bug fixes) before the two most important supercomputing conferences: ISC High-Performance, in Europe, around July, and the SC conference, in the United States, around the end of November.

Long-term commitment to further development, maintenance, support and active maintainer organization / sponsor

Score-P has been created in the German BMBF project SILC and the US DOE project PRIMA and has so far been maintained and enhanced in a number of follow-up projects such as LMAC and HOPSA, among others (already finished). Further funding for development, maintenance and support is given by projects like Score-E, Prima-X, CATWALK and others, and the partners actively work in order to extend funding beyond the current round of project proposals. Besides, Forschungszentrum Jülich and TU-Dresden have their own internal funding for developers that ensure the project's continuity.

2.3.2 Availability and Coverage

Licensing

- Score-P is developed under a BSD 3-Clause License.

Distribution media

- Online, CD/DVD : Both the source packages of the recent versions of Score-P, OTF2, Cube and OPARI2 (the software packages/libraries developed concomitantly with Score-P), as well as a demo version of all the software in a LIVE DVD ISO image can be downloaded from Score-P's website. During Tuning Workshops, DVD copies of the most recent software bundle are handed to the participants, in order to test the tools on their own machines. Another possible form of installation is by using EasyBuild, (http://hpcugent.github.io/easybuild/), a software build and installation framework created to manage scientific software on High Performance Computing (HPC) systems in an efficient way.
- Source code or binary: The release package is source-code only, and must be manually compiled by the user. The LIVE DVD contains both the sources and compiled binaries for the target platform (usually, X86-64).

- Binary Packages (e.g. RPM): No
- Source-code based package installers (e.g. EasyBuild): Yes
- Access to project repository (read-only or completely open, e.g. GitHub) and/or ticket system: Access to project's repository is closed to the developers only.

Organizational form of (distributed) developer consortium:

- Open-source; Well-defined governance model: Score-P is governed by a meritocratic governance model. See http://www.vi-hps.org/upload/packages/scorep/scorep_gov.pdf

Coverage

- Languages: Score-P is developed mostly in C. The visualization tool Cube is developed in C++ with Qt. A Java version of the Cube writer library was developed for integration with TAU. Score-P is able to instrument codes in C, C++ and Fortran.
- Compilers: Score-P is regularly tested with GCC, Intel, PGI, Oracle Solaris Studio, and IBM's XL.
- Score-P supports the following programming paradigms:
  - Multi-process paradigms:
    - MPI
    - SHMEM
  - Thread-parallel paradigms:
    - OpenMP
    - Pthreads
  - Accelerator-based paradigms:
    - CUDA
    - OpenCL
- Platforms beyond PRACE: According to the configure file, Score-P can be run on SGI Altix, the IBM's Blue/Gene L, P, and Q series, IBM's Power 6 and 7, Cray's XT, XE, XK, and XC series, ARM processors, the K computer, Fujitsu's FX10 and FX100 Series, Solaris, Aix, and Linux systems, as well as Intel MIC. Windows is partially supported.
- Accelerator support: Support for instrumenting CUDA and OpenGL codes is present. Intel MIC is supported.

Number of users / installations / component downloads

Since this is an open-source software, it is hard to measure the number of users and installations. Several supercomputers on the TOP500 list have Score-P (see next section). The “scorep-news” mailing list contains around 120 users, several of them being system administrators that use the list to receive the news and update their systems accordingly.

2.3.3 Portability and Scalability

Installation test on major PRACE platforms

Score-P is currently installed, maintained and tested on the following platforms:

- JUQUEEN, a 5 PFLOPS IBM BlueGene/Q system, at the Jülich Supercomputing Centre of Forschungszentrum Jülich, in Jülich, Germany. http://www.fz-juelich.de/ias/jsc/EN/Expertise/Supercomputers/JUQUEEN/Configuration/Configuration_node.html
- JUROPA, a 207 TFLOPS Intel Nehalem-EP system, also located at the Jülich Supercomputing Centre: http://www.fz-juelich.de/ias/jsc/EN/Expertise/Supercomputers/JUROPA/Configuration/Configuration_node.html
- Blizzard, a 158 TFLOPS IBM POWER 6 used by German climate researchers at the Deutsches Klimarechenzentrum, in Hamburg, Germany: https://www.dkrz.de/Klimarechner-en/hpc
- Hornet, a 3.8 PFLOPS Cray XC40 system, located at the High Performance Computing Center Stuttgart (HLRS) of the University of Stuttgart:

https://www.hlrs.de/news/systems/systems/article/supercomputer-hornet-up-and-running/
- K computer, a 10.5 PFLOP Fujitsu system, located at the RIKEN Advanced Institute for Computational Science in Kobe, Japan: http://www.aics.riken.jp/en/k-computer/about/
- PRIMEHPC FX10 90 TFLOPS Fujitsu System, also at RIKEN
- Piz Daint, a 7.7 PFLOPS Cray CX30 system located at the Swiss National Supercomputing Centre, in Lugano, Switzerland: http://www.cscs.ch/computers/piz_daint/index.html
- Vilje, a 467TFLOPS SGI Altix 8600 system, at the Norwegian Metacenter for Computational Science, in Trondheim, Norway: https://www.notur.no/hardware/vilje
- TIANHE-2, a 33.86 PFLOPS an Intel Xeon and Xeon Phi system at the China's National University of Defense Technology, in Changsha, at the Hunan province of China: http://spectrum.ieee.org/tech-talk/computing/hardware/tianhe2-caps-top-10-supercomputers
- Among others.

Execution test on platforms using a reasonable number of nodes/cores (1024 nodes?)

Benchmarking was done on JuQueen at Jülich Supoercomputing Centre, an IBM BlueGene/Q which is part of the PRACE Tier-0 systems, with the Unified European Application Benchmark Suite (see http://www.prace-ri.eu/ueabs). Tests were performed with and without measurement filtering, up to 65536 MPI processes. The benchmark suite consists of 12 applications (ALYA, Code_Saturne, CP2K, GADGET, GENE, GPAW, GROMACS, NAMD, NEMO, QCD, QuantumEspresso, SPECFEM3D).

In summary, it was possible to build 11 applications (ALYA, Code_Saturne without GUI, CP2K, GADGET, GENE, GROMACS, NAMD, NEMO, QCD, QuantumEspresso, SpecFEM3D), all of them ran and produced reasonable results. So far, GPAW could not be built on BlueGene/Q. 10 of those 11 applications could be instrumented and ran under control of Score-P. ALYA, Code_Saturne, GROMACS, GADGET, GENE, NAMD (without compiler instrumentation), NEMO, QCD, QuantumEspresso and SpecFEM3D produced reasonable profiles and trace files whereas CP2K has a segmentation fault due to lack of available memory on compute node.

Comparison of the results with results provided in PRACE-3IP (http://www.prace-ri.eu/IMG/pdf/d7.3.2_3ip.pdf are difficult due to following issues:

- The runtime configuration is not always provided (amount of compute nodes, processes, threads, amount of iterations or time steps);
- The version of application is not always provided,
- It was not always provided how execution time was measured.

For each of the benchmarks, the following steps were performed:

1. Build application
2. Run application with provided test cases
3. Instrument with Score-P 1.4
4. Run instrumented application, produce profile.
5. Prepare filter file based on profile in step 4
6. Collect trace files based on SIONlib-1.5.5
7. Analyze trace files with Scalasca-2.2

The results in detail:

- ALYA-1.1: Successful. Tests were done with Test Case A on 256 nodes. Measurements on larger node counts were no reasonable to the necessary sequential pre-processing part of ALYA and the lack of enough memory for this on the compute nodes of JuQueen.
- Code_Saturne: Successful. Tests were done with Test Case A (51M cells, 10 time steps) up to 8192 nodes and Test Case B (110M cells, 10 time steps) up to 65538 nodes. Traces were only collected for Test Case A, as it was unpractical for B due to a huge amount of MPI_Irecv calls (over 1 million per rank).
- CP2K-2.4.0: FAILED. Instrumented applications fails with segmentation fault during runtime due to lack of sufficient memory on compute node.

- Gadget-3: Successful for smaller test case. Large test case did not work due to an MPI_Allgatherv issue in un-instrumented application.
- GENE: Successful. Tests were done with version provided by developers; code and test cases provided on PRACE website did not work. Tests were done on up to 512 nodes.
- GROMACS-4.6.7: Successful. Tests were done with Test case B (3.3M atom model of cellulose and lignocellulosic biomass in aqueous solution, 100 000 time steps, pure MPI) up to 32768 nodes. Larger tests failed due to the domain decomposition in the provided data set. Trace analysis with more than 16384 failed due to memory constraints in the Scalasca MPI communicator handling.
- NAMD-2.10: Successful. Tests were done with up to 65538 nodes, however without function instrumentation (MPI only measurement).
- NEMO-3.4: Successful. Tests with up to 8192 nodes.
- QCD-1.1: Successful. Tests were done with kernels A, B, and C up to 65536 nodes, with kernels D and E up to 16384 nodes. Trace analysis was not possible in all cases due to memory limitations.
- QuantumEspresso-5.1.2: Successful. Tests were done with Test case A (cp) and B (pw) up to 2048 nodes.
- SpecFEM3D-6.0.0: Successful. Tests were done with default and Test Case A on up to 864 nodes.

Scalability

Score-P 1.4.1 was successfully tested on JUQUEEN with the NAS BT-MZ benchmarks with 1,048,576 threads for profiling and with 655,360 threads for tracing. The lower number for tracing is due to an internal size limit in definition records for the OTF2 trace format which will be removed in future versions.

2.3.4 Performance and Quality

Code quality

- Existence of developer documentation: The developers' internal website has a developer's corner. It contains instructions on how to download and build the sources, how to start the creation of branches for new feature development, directory structure, and others. The code itself uses the automatic documentation creator Doxygen.
- Coherent usage of coding guidelines/standard: Each new feature must be created and tested in its own branch, thus not interfering with other developments. Files, directories, functions and variables have a consistent naming scheme. The coding style is found in the “Naming Conventions” page, at the end of the developers' corner.
- Source code quality: Code is must pass some syntax/indentation checks before it can be committed. Passing this test, it will be peer-reviewed, and then automatically tested.
- Use of (automated) standardized (regression) tests: Edgewall's BITTEN is used as a regression test suite. A total of 301 Bitten “Slaves” are regularly executed.

Ease-of-use

- Detailed feature description and user manual: Score-P comes with extensive documentation. Besides the user manual, multiple presentations from previous tutorials are available at the URL http://www.score-p.org
- Existence of examples and demos: The website has the LIVE ISO image, which contains examples of Score-P runs on some supercomputers.
- Ease of installation, configuration, and usage: Score-P is a standard “configure; make; make install” program. It tries to auto-detect settings for several systems, compilers, MPI libraries and others. Being a development tool, it provides many other configurable options. Another way of installing and using Score-P is through the aforementioned EasyBuild. It automatically brings, compiles and install all dependencies required by Score-P besides of Score-P itself.

Testability

- Built-in distribution and installation checks: Yes

- Availability of test suites: The Cube Library, one of Score-P's components, contain tests to ensure correctness,
- Availability of benchmark suites, benchmark data sets or micro-benchmarks: No

Fault-tolerant/resilience support

- Checkpoint/restart: No

Re-usability

- Two components of Score-P, the OTF2 (Open Trace Format), and CUBE4 file format are used by other applications, such as Vampir (https://www.vampir.eu/), a commercial performance optimization visualizer developed by GWT-TUD GmbH located in Dresden, Germany and by TAU, the “Tuning and Analysis Utilities”, developed by the University of Oregon and by the Los Alamos National Laboratory, both in the United States of America: http://www.cs.uoregon.edu/research/tau/home.php
- Score-P is used as the basic instrumentation and measurement tool for the Scalasca performance analysis toolset, developed by the Jülich Supercomputing Centre, and uses Score-P component Cube as its primary visualization tool: http://www.scalasca.org.

Efficiency

Score-P is used to instrument code and provide measurements at runtime. In order to avoid noise in the measurements by the measurement system itself, Score-P avoids flushes to disk while tries to keep memory usage at minimum.

This measurement system can do light profiles, that counts visits to regions in the user's code, or full traces of an application execution, which can skew measurement data heavily. In order to keep efficiency, the profiling as well as the tracing measurements can be filtered, and the memory usage of Score-P can be defined by the user, according to memory availability on the execution's machine, memory usage of the application itself, etc. Detailed information can be found at the “measurement” webpage: https://silc.zih.tu-dresden.de/scorep-current/html/measurement.html.

User satisfaction

- List of organizations, testimonials, and other projects using this software: Besides the institutions mentioned within the “Portability and Scalability” section, the following projects use Score-P:
  - DEEP Project (Dynamical Exascale Entry Platform) http://www.deep-project.eu
  - DEEP's follow-up project DEEP-ER (Dynamical Exascale Entry Platform – Extended Reach), http://www.deep-er.eu
  - Mont Blanc Project: http://www.montblanc-project.eu
  - Human Brain Project: https://www.humanbrainproject.eu
  - BMBF Project GASPI: http://www.gaspi.de/en/
  - SPPEXA project DASH: http://www.dash-project.org
- Case studies, usage histories: In the article Implementation and scaling of the fully coupled Terrestrial Systems Modeling Platform (TerrSysMP v1.0) in a massively parallel supercomputing environment - a case study on JUQUEEN, the authors report a speedup of up to 19% after analysing the application with Score-P and Scalasca. See http://www.geosci-model-dev.net/7/2531/2014/gmd-7-2531-2014.pdf
- Reports from forums, blogs, mailing lists, newsgroups, magazines, scientific articles: There has been 8 VI-HPS tuning workshops, 6 additional hands-on workshops and several other HPC school workshops. The average attendance is around 18 participants per workshop. For each workshop, questionnaires asking for quality of the training and the tools were collected. Typically, the training participants rated the tools 4 (out of 5) points.
- Organized user community (e.g. annual user group meetings or BoFs): The very own specificity of a HPC tool hardly limits the score of an organized user community. Given the special case posed by the project's scope, a “community” per se makes little sense. However, Score-P is managed by a Meritocratic Governance Model, and the admission is open to any individual or organization interested in the development of the project. See http://www.vi-hps.org/upload/packages/scorep/scorep_gov.pdf . Multiple institutions applied for

participation, making this a real community development, although not in the classic sense of open-source projects.

- Quality of support services:
  - Average response time: Score-P is supported by its developers in a best-effort sense, and most e-mails are answered on the same day.
  - Average number of contacts with customer to resolve issue: Most user questions are answered in a round of e-mails. The exception is usually fault of the user (e.g. not providing enough information in order to solve the issue).

2.4 Component 2: Programming Model OmpSs

Short description of the component: OmpSs (https://pm.bsc.es/ompss) is a programming model whose goal is to extend OpenMP with new directives to support asynchronous parallelism and heterogeneity (devices like GPUs). However, it can also be understood as new directives extending other accelerator based APIs like CUDA or OpenCL.

OmpSs is one of the implementations of the StarSs programming model idea, and had previous instances for specific architectures and computing platforms (GRIDSs, SMPSs, CellSs, …). StarSs is a task based programming model based on the idea of a sequential program with task annotations and indication of the directionality of the arguments of the tasks. From this sequential code, a data-dependence graph is built at execution time, where the nodes represent tasks and edges data-dependences between them. Then, this graph defines a partial execution order of the tasks and potential concurrency. The StarSs runtime is then able to schedule the tasks in the computing resources and perform other operations like data transfer, exploitation of data locality, etc.

OmpSs was initially thought as an integration of the StarSs ideas with OpenMP. Developed by BSC, OmpSs has been used to demonstrate the StarSs ideas and push them into the OpenMP standard. Concepts already integrated in the standard are the task model and the data dependence concept.

2.4.1 Documentation and Support

Availability of Documentation

- Up-to-date installation guide: http://pm.bsc.es/ompss-docs/user-guide/installation.html – and on the PDF, found at http://pm.bsc.es/ompss-docs/user-guide/OmpSsUserGuide.pdf
- User guide: http://pm.bsc.es/ompss-docs/user-guide/ and the PDF mentioned above.
- FAQ: http://pm.bsc.es/ompss-docs/user-guide/faq.html and can be found at the user guide PDF.
- Quick introduction (getting started): http://pm.bsc.es/ompss-docs/user-guide/run-programs.html
- Release notes: No.
- Up-to-date training classes or courses: The “Programming Models @ BSC website” (https://pm.bsc.es/) lists courses and tutorials about OmpSs since 2012.
  - The first mention of a tutorial is from Supercomputing 2012.
  - In 2013, a tutorial (in Spanish) was given at the “Red de Computación de Altas Prestaciones sobre Arquitecturas Paralelas Heterogéneas (CAPAP-H)”, see http://capap-h.uji.es/?q=node/142.
  - In July 2013, at the Programming and Tuning Massively Parallel Systems Summer School (PUMPS), there was an introduction to OmpSs and for the Paraver analysis tool. See http://bcw.ac.upc.edu/PUMPS2013/start.
  - In 2013, a course on heterogeneous programming on GPUs with MPI + OmpSs was given at SBAC-PAD, in Porto de Galinhas, Brazil. See http://www.cin.ufpe.br/~sbac2013/sbac/overall_program_new.php.
  - In May 2013, a tutorial at XSEDE project was given at the University of New York. See https://pm.bsc.es/blog/xteruel/ompss-tutorial-xsede-project_1375274402.
  - In June, 2014, a course on programming models using OmpSs was given in Bucaramanga, Colombia. See https://pm.bsc.es/blog/xteruel/course-programming-models-using-ompss_1401460142 .
  - In May, 2014 there was a course on Heterogeneous Programming on GPUs with MPI +

    - OmpSs, at the PRACE Advanced Training Centre (PATC) – See http://www.bsc.es/marenostrum-support-services/hpc-events-trainings/prace-trainings/clone-patc-course-23-24-may12.
    - The same course will happen in may, 2015. See http://www.bsc.es/patc-programming-2015.
    - An OmpSs tutorial will be given at the Programming and Tuning Massively Parallel Systems summer school (PUMPS) in July, 2015. See http://bcw.ac.upc.edu/PUMPS2015/start.
- Up-to-date training materials, manuals, guidelines: The most recent training material is from October 2014, at the SBAC-PAD. See http://sbac.lip6.fr/2014/session%204/5-OmpSs.pdf.
- Availability of professional (commercial, contracted) training: No
- Published books by persons not affiliated with the component: No
- Documentation available in multiple languages: OmpSs' documentation is in English.
- Published procedure for user (feature) requests: The website https://pm.bsc.es/ directs the users to send an e-mail to pm-tools@bsc.es. One can also join the low-traffic pm-tools-users mailing list by sending an e-mail to pm-tools-users-join@bsc.es.
- Product examples or demos: http://pm.bsc.es/ompss-docs/examples/OmpSsExamples.pdf Also an application repository is publicly available under https://pm.bsc.es/projects/bar/
- Published roadmap (future platforms and features): No

Support (bug fixing, help)

- Website, support mailing list, user forum, phone service: OmpSs's website (https://pm.bsc.es/) lists the e-mail pm-tools@bsc.es for questions or suggestions. The pm-tools-users-join@bsc.es is the subscription address for the user mailing list. No telephone number is mentioned.
- 24/7 support or best effort: best-effort.
- Usage of bug tracking and/or ticket systems: The OmpSs project uses two TRAC systems for its tickets: https://pm.bsc.es/projects/nanox and https://pm.bsc.es/projects/mcxx. GIT is used internally. http://pm.bsc.es/git
- Availability of professional (commercial, contracted) support: No
- Average bug time solving: different modules have different response times. Researchers working on their theses might consider more important to have a working prototype for their work, and fix things later. Nanos++ has 177 open bugs as can bee seen here: https://pm.bsc.es/projects/nanox/report/1 and mercurium has 669 bugs, as seen here: https://pm.bsc.es/projects/mcxx/report/1
- Frequency of releases and patches: It's possible to find 8 versions of OmpSs on their download page, from April, 2014, until April, 2015 – See https://pm.bsc.es/ompss-downloads. An automatic system tries to create automatic daily snapshots of the nanox and mcxx components.

Long-term commitment to

- Further development, maintenance, support and active maintainer organisation / sponsor: OmpSs is maintained by the Barcelona Supercomputing Center. Training, development, maintenance and support are partially funded by different initiatives, like Nvidia via the CUDA research center. (http://ccoe.ac.upc.edu/bscupcccoe), by the Partnership for Advanced Computing in Europe (PRACE) http://www.prace-ri.eu/, and by the Spanish National Supercomputing Network, and a large number of EC funded projects (TEXT, Montblanc, DEEP, Intertwine, Exanode…).

2.4.2 Availability and Coverage

Licensing

- Gnu General Public License version 2.

Distribution media

- Online, CD/DVD: Online only: https://pm.bsc.es/ompss-downloads

- Source code or binary: One can download the source code directly, or use the binary packages repositories provided at the downloads page.
- Binary Packages (e.g. RPM): Yes, a RPM repository is available for CentOS 6.5, one for OpenSuse11.4 and later, and there is one DEB repository for Debian 7.6 and recent versions of Ubuntu.
- Source-code based package installers (e.g. EasyBuild): No
- Access to project repository (read-only or completely open, e.g. GitHub) and/or ticket system: the two OmpSs components, Nano++ and Mercury, have each their own Trac: https://pm.bsc.es/projects/nanox and https://pm.bsc.es/projects/mcxx, public Git repository: http://pm.bsc.es/git/nanox.git and http://pm.bsc.es/git/mcxx.git, and private Git repositories for developers, at https://pm.bsc.es/git-dev/nanox.git and https://pm.bsc.es/git-dev/mcxx.git.

Organizational form of (distributed) developer consortium:

- Open-source; mainly developed by BSC developers team with occasional contribution from external developers.

Multiple sources (PM, LF only)

- Multiple implementations available: OmpSs itself is a specialization of OpenMP. The first question at their FAQ is about the difference among each other: http://pm.bsc.es/ompss-docs/user-guide/faq-openmp-vs-ompss.html#initial-team-and-creation – however, other OpenMP implementations do not follow OmpSs' specification. We are not aware of another implementation available.
- Standards, especially multiple versions of a standard: OmpSs standard is implemented directly by the Mercurium compiler and Nanos++ runtime system.

Coverage (LF, DT, PM only)

- Languages: C, C++ and Fortran
- Compilers: Nanos++ supports most C/C++ and Fortran compiler (GNU, Intel, IBM, …) , and can use CUDA and OpenCL for GPU support. Mercurium, their C/C++ source-to-source compiler, requires a backend C/C++ or Fortran compilers, among other tools from the GNU toolchain: http://pm.bsc.es/ompss-docs/user-guide/installation.html#installation-of-nanos
- Platforms beyond PRACE: It currently supports Linux on i386, x86-64, Xeon Phi, ARM, PowerPC and IA64.
- Accelerator support: It supports CUDA and OpenCL, and Xeon Phi as well.

Coverage (AC, PM only)

Application areas/use cases: OmpSs was created with accelerators in mind. According to the project description, “In particular, our objective is to extend OpenMP with new directives to support asynchronous parallelism and heterogeneity (devices like GPUs). However, it can also be understood as new directives extending other accelerator based APIs like CUDA or OpenCL.”

Number of users / installations / component downloads

OmpSs’ developers distribute both source code and compiled distributions for the most common Linux distributions (CENTOS, OpenSUSE, DEBIAN and UBUNTU). It can be found at http://pm.bsc.es/ompss-downloads

There have been more than 1000 downloads of the distribution during last year (raw statistics of downloads for the last 52 weeks can be find at http://pm.bsc.es/~rferrer/logs/ompss.html）.

2.4.3 Portability and Scalability

Installation test on major PRACE platforms

- BlueGene, Cray, Linux cluster, other: In Forschungszentrum Jülich, the Jülich Dedicated GPU Environment (JuDGE) is the reference platform for OmpSs. In the MareNostrum

supercomputer and MinoTauro cluster (Barcelona Supercomputing Center): https://pm.bsc.es/ompss-bsc

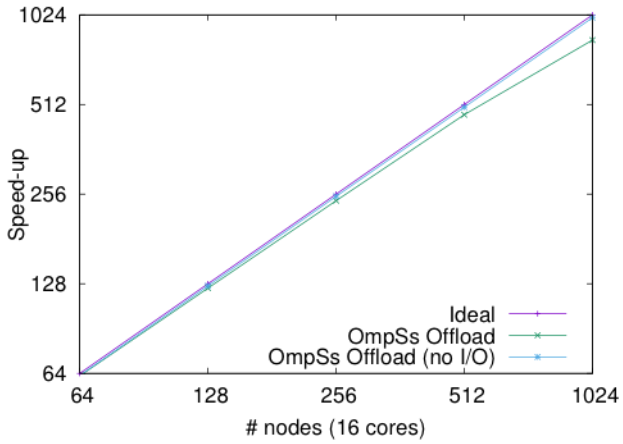
Execution test on platforms using a reasonable number of nodes/cores (1024 nodes?)

- Being OmpSs a node-level programming model, the port of the PARSEC benchmarks is an example which covers a wide range of scientific and non-scientific applications and have been run on reasonable size multicore nodes, improving performance versus pthreads or OpenMP 3.0 versions of the same codes. Other benchmarks such as Cholesky or linear algebra, such as the one reported in the Scalability section, have been run in large NUMA nodes. In its hybrid MPI/OmpSs usage example reference cases includes Hydro, CGPOP, SPECFEM3D mini-applications.

Scalability

- OmpSs in combination with MPI has been compared in distributed platforms against pure MPI or MPI+OpenMP showing better scalability. Also, it has been compared in NUMA platforms against other programming models or libraries.

- Next chart shows results of OmpSs offload of MPI applications. It compares the speedup of the FWI application with OmpSs offload running in MareNostrum with up to 1024 nodes (each of them 16 cores).

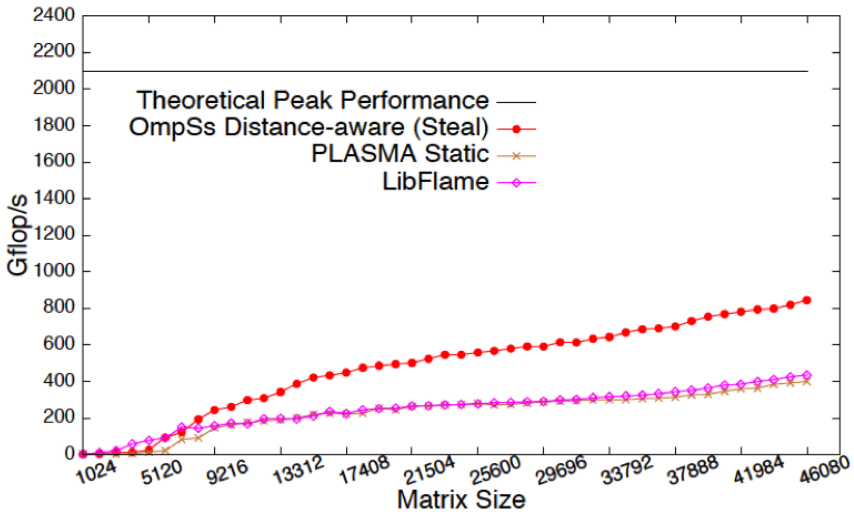


More results of the OmpSs offload will be available in the following DEEP deliverables:

D5.5) Optimised and malleable runtime: Including the minimisation of the data transfers and optimised BN to BN communications. The runtime will also interface to the dynamic BNs allocation mechanisms, being able to demand or release BNs as the application parallelism requires.

D8.3) Final pilot applications report: Description of porting effort, use of communication between BNs and CNs, and benchmarking results obtained with the DEEP System. Report on best practices and recommendations for improvements.

With regard experiments in NUMA aware architectures, the following results are for a Cholesky factorization in OmpSs, using the OmpSs NUMA-aware scheduler, run in a SGI Altix 1000 UltraViolet shared-memory machine based on Intel Nehalem EX processors featuring 128 sockets, eight cores each, running at 2:0GHz with one NUMA node per socket, comparing with PLASMA and LibFlame. These results were obtained using 256 threads.



These and other results on NUMA platforms will be published in:

R. Al-Omairy, G. Miranda, H. Ltaief, R. M. Badia, X. Martorell, J. Labarta and D. Keyes, Dense Matrix Computations on NUMA Architectures with Distance-Aware Work Stealing, to be published in Journal of Supercomputing Frontiers and Innovations.

2.4.4 Performance and Quality

Code quality

- Coherent usage of coding guidelines/standard: Coding guidelines can be found at https://pm.bsc.es/projects/nanox/wiki/BestPractices/CodingGuidelines
- Source code quality: OmpSs is a research project. Different sections will have widely different quality levels, according to the researcher or student developing the prototype for their research. That being said, everyone writing code for the project has to abide by the aforementioned development guidelines. Besides, OmpSs is developed in the form of plugins. That assures that the work of one researcher will not interfere with the works of others (see more at the “Efficiency” section).
- Existence of developer documentation: A developer's guide for Nanos++ can be found at https://pm.bsc.es/projects/nanox/wiki , and a developer's page for Mercurium can be found at https://pm.bsc.es/projects/mcxx/wiki/Developers
- Programming language uniformity: unclear
- Use of (automated) standardized (regression) tests: The project uses Jenkins (https://jenkins-ci.org/), but its access is for developers only.

Ease-of-use

- Detailed feature description and user manual: There is ample documentation. Section 2 of the documentation shows most of the features of OmpSs' programming model: http://pm.bsc.es/ompss-docs/specs/02_programming_model.html . More can be read at the “Language Description”, section 3 of the user's manual: http://pm.bsc.es/ompss-docs/specs/03_language_description.html
- Existence of examples and demos: There are code examples at http://pm.bsc.es/ompss-docs/specs/examples.html and solutions to standard scientific problems (Matrix multiplication, parallel sort, common kernels) at http://pm.bsc.es/ompss-docs/examples/README.html .
- Ease of installation, configuration, and usage: Nanos++ and the Mercurium C/C++ compiler are a standard “./configure; make; make install” project. There are plenty of options documented on the user guide at http://pm.bsc.es/ompss-docs/user-guide/installation.html – alternatively, one can just use the pre-compiled packages, whose repositories are described here: https://pm.bsc.es/ompss-downloads . Detailed usage can be seen at the “Compiling

OmpSs programs" and "Running OmpSs programs", available here: http://pm.bsc.es/ompss-docs/user-guide/compile-programs.html and http://pm.bsc.es/ompss-docs/user-guide/run-programs.html .

Testability

- Built-in distribution and installation checks: There is a simple "make check".
- Availability of test suites: There are test suites for the supported Job Schedulers, examples for beginners, GPU device and MPI+OmpSs exercises: http://pm.bsc.es/ompss-docs/examples/README.html
- Availability of benchmark suites, benchmark data sets or micro-benchmarks: There are three benchmarks that are supported by the OmpSs' development team: Cholesky kernel, the Stream Benchmark, and the Array Sum: http://pm.bsc.es/ompss-docs/examples/ompss-ee/01-examples/README.html

Fault-tolerant/resilience support

- Checkpoint/restart: There is current research on adding checkpoint/restart capabilities, as well as redundancy. See http://www.gsd.inesc-id.pt/~mcouceiro/eurotm/dmtm2014/subasi.pdf and http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7092706. Furthermore, Checkpoint/restart in OmpSs are a proposed feature for the DEEP-ER project; see http://www.deep-er.eu/software/resiliency

Re-usability

The concepts developed in OmpSs are making their way into the official OpenMP standard: tasks, dependencies and support for compute devices are part of OpenMP 4 Specification, as seen at http://openmp.org/wp/2013/07/openmp-40/

Efficiency

According to https://pm.bsc.es/projects/nanox/doxygen/, when relating to OmpSs runtime library Nanos++, "The main purpose of Nanos++ is to be used in research of parallel programming environments. Our aim has been to enable easy development of different parts of the runtime so researchers have a platform that allows them to try different mechanisms. As such it is designed to be extensible by means of plugins: the scheduling policy, the throttling policy, the dependence approach, the barrier implementations, the slicers implementation, the instrumentation layer and the architectural level. This extensibility does not come for free. The runtime overheads are slightly increased, but there should be low enough for results to be meaningful except for cases of extreme-fine grain applications."

User satisfaction

- List of organizations, testimonials, and other projects using this software:
  - The Project Mont Blanc, at http://www.montblanc-project.eu/;
  - Dynamical Exascale Entry Platform (DEEP), at http://www.deep-project.eu/;
  - And its follow-up project Dynamical Exascale Entry Platform - Extended Reach (DEEP-ER) http://www.deep-er.eu
- Case studies, usage histories:
  - Using OmpSs for analysing seismic images: http://www.hpcwire.com/2014/06/03/seismic-imaging-deep-end
- Reports from forums, blogs, mailing lists, newsgroups, magazines, scientific articles
  - Scoring in PATC trainings were always very positive
- Organized user community (e.g. annual user group meetings or BoFs):
  - http://www.montblanc-project.eu/industrial-user-group
- Quality of support services
  - Average response time: the requests send to the pm-tools@bsc.es list have multiple recipients (developers, researchers, even team leaders) and although there is no a formal procedure stablished they are usually answered very fast (in a few hours, most of times in the same day).
  - Average number of contacts with customer to resolve issue: usually after a round of at most two or three emails the issues are fixed.

2.5 Component 3: Communication Library GASPI/GPI-2

Short description of the component: GPI-2 (http://www.gpi-site.com/gpi2/) is the Fraunhofer ITWM's implementation of the Global Address Space Programming Interface (GASPI), a Partitioned Global Address Space (PGAS) API. GPI-2 is an API for asynchronous communication. It provides a flexible, scalable and fault tolerant interface for parallel applications.

2.5.1 Documentation and Support

Availability of Documentation

- Up-to-date installation guide: Basic instructions in the README file that comes with the source code: https://github.com/cc-hpc-itwm/GPI-2/blob/next/README
- User guide: The API specification is at http://www.gaspi.de/fileadmin/GASPI/pdf/GASPI-1.0.1.pdf (pdf, 120 pages)
- FAQ: There are two FAQs, that roughly coincide: http://www.gpi-site.com/gpi2/faq and http://www.gaspi.de/en/faq.html
- Quick introduction (getting started): The documentation website has a quick introduction, at http://www.gpi-site.com/gpi2/docs/
- Release notes: Short announcements of new releases are published at the “News” section of GPI-2's website. This is the most recent release announcement available at release time: http://www.gpi-site.com/gpi2/new-gpi-2-release-v1-1-1
- Up-to-date training classes or courses:
  - The same “News” section of its website lists the most recent tutorials. In January, 2014, one was realized at the HLRS in Stuttgart, Germany: http://www.gpi-site.com/gpi2/gaspi-tutorial-at-hrls
  - In June, 2014, a tutorial was held at the Leibniz Supercomputing Centre: http://www.gpi-site.com/gpi2/2014/05
  - At the same HLRS, a tutorial was held last April, 2015: http://www.hlrs.de/organization/sos/par/services/training/2015/GASPI
  - In May, 2015, a tutorial will be held at the University of Bristol: https://events.prace-ri.eu/event/382
- Up-to-date training materials, manuals, guidelines: The most recent training material was published in June, 2014, when GPI-2 version 1.0.1 was available. At the time of writing, versions 1.1.0 and 1.1.0 were published. http://www.gpi-site.com/gpi2/tutorial
- Availability of professional (commercial, contracted) training: Yes, for commercial users. No further information is available; The website redirects to the “Contact” page: http://www.gpi-site.com/gpi2/contact.
- Published books by persons not affiliated with the component: No
- Documentation available in multiple languages: The documentation is completely in English
- Published procedure for user (feature) requests: The help webpage (http://www.gpi-site.com/gpi2/help) directs and shows the last entries of the gpi2-users mailing list.
- Product examples or demos: Some benchmark runs, and products that use GPI-2, and simulation results (Such as a 3D render) are at http://www.gpi-site.com/gpi2/benchmarks
- Published roadmap (future platforms and features): The README file of the distribution available at GitHub has a brief section on upcoming features: https://github.com/cc-hpc-itwm/GPI-2/blob/next/README (section 9)

Support (bug fixing, help)

- Website, support mailing list, user forum, phone service: The website directs you to read the documentation first, and then consult the gpi2-users mailing list, readable from the “Help” section of GPI-2's website: http://www.gpi-site.com/gpi2/help
- 24/7 support or best effort: The aforementioned users' mailing list is on a best-effort basis.
- Usage of bug tracking and/or ticket systems: The GitHub accepts the submission of issues,

but it does not seem actively used by the developers. There are only 7 bugs ever reported: https://github.com/cc-hpc-itwm/GPI-2/issues?q=is%3Aissue+is%3Aclosed. Mentions of an internal Git repository can be seen through the code, such as here: https://github.com/cc-hpc-itwm/GPI-2/commit/342677dc8015628337901eae0a22f1b6e3ad8aa5

- Availability of professional (commercial, contracted) support: Yes. Details are not provided.
- Average bug time solving: Unknown. The few ones present at GitHub were ignored for a while, then closed.
- Frequency of releases and patches: The news of the 3 available versions of GPI-2 report releases happened in July, 2013, June, 2014 (in time for ISC'2014), and November, 2014 (In time for SC'2014).

Long-term commitment to

- Further development, maintenance, support, active maintainer organisation / sponsor: Fraunhofer ITWM (http://www.itwm.fraunhofer.de/en.html) and its commercial spinoff, SCAPOS (http://www.scapos.com) have been providing fixes, licensing and support for the former version of the software - GPI – for a number of years. GPI-2's development, maintenance, and support are expected to continue on the same fashion. Other partners on the specification project can be seen at: http://www.gaspi.de/en/partners.html. Besides, the EU's project EPIGRAM states that their EXASCALE PGAS goal is to use PGI (no mention to PGI2, but it can understood as such) as their reference implementation: http://www.epigram-project.eu/project/.

2.5.2 Availability and Coverage

Licensing

GPI-2 uses the Gnu Public License, version 3. Commercial licenses are not mentioned in the source, but are hinted at in the Contact session of the Website: “If you have any question or suggestion and for more information on commercial GPI licenses and support, you can use the following form or the contacts at the end of this page.” (http://www.gpi-site.com/gpi2/contact/)

Distribution media

- Online, CD/DVD: Online
- Source code or binary: Source code
- Binary Packages (e.g. RPM): No
- Source-code based package installers (e.g. EasyBuild): No
- Access to project repository (read-only or completely open, e.g. GitHub) and/or ticket system: Uses GitHub (https://github.com/cc-hpc-itwm/GPI-2). Some form of integration between their intranet's development server and GitHub can be hinted at the commits. It seems that the development was opened after that.

Organizational form of (distributed) developer consortium:

GPI-1 used to have a closed model development. After the creation of the standard specification GASPI and the version bump to GPI-2, development moved to Github with a GNU Public License version 3; however, development is performed internally by Fraunhofer. Only 3 users had committed to the GPI-2 source code base on GitHub in the past 18 months. 99% of the development is made by dr. Rui Mario da Silva Machado. http://www.itwm.fraunhofer.de/abteilungen/hpc/employees/dr-rui-mario-da-silva-machado.html

Multiple sources (PM, LF only)

- Multiple implementations available: GPI-2 is the open-source reference implementation of GASPI (http://www.gpi-site.com/gpi2/gaspi/ and http://www.gaspi.de/en/project.html). The open specification and open source implementation were created after the commercial version, GPI
- Standards, especially multiple versions of a standard: Only one version of the GASPI standard

(1.0), with minor corrections at 1.0.1: http://www.gaspi.de/fileadmin/GASPI/pdf/GASPI-1.0.1.pdf

Coverage (LF, DT, PM only)

- Languages: C, C++ and Fortran
- Compilers: No compilers are mentioned in the documentation. The Xeon Phi modules need the Intel compiler, and GFORTRAN is needed for Fortran bindings. It depends on the GNU toolchain, but uses whatever the "make" command understands as the $CC environment variable as the standard C compiler.
- Platforms beyond PRACE: None known
- Accelerator support: Xeon Phi, CUDA Support

Coverage (AC, PM only)

Application areas / use cases: Some common use cases are that of Computational Fluid Dynamics, Stencil codes, and several graph algorithms.
http://www.gpisite.com/cms/sites/default/files/gpi_flyer_apps.pdf

Number of users / installations / component downloads

GPI has been used as the parallel programming model for the following industrial applications:

- Pre-stack Pro: A tool for visualization, interactive processing, and quantitative amplitude analysis of pre-stack seismic data. http://www.envision.no/products/2
- 3D-GRT: Fraunhofer ITWM and Statoil have developed an angle domain migration based on the theory of generalized Radon transform (GRT) for massively parallel commodity clusters. http://www.seismic-grt.com/index.php/history
- Aerospace simulations for the German Aerospace Center: https://www.fraunhofer.de/en/press/research-news/2013/june/programming-model-for-supercomputers-of-the-future.html

2.5.3 Portability and Scalability

Installation test on major PRACE platforms

As can be seen at http://www.epigram-project.eu/wp-content/uploads/2013/11/gpi2.pdf, it is installed on

- Hornet, at the High Performance Computing Center Stuttgart (HLRS): https://wickie.hlrs.de/platforms/index.php/GPI-2
- TU Dresden
- Jülich
- GPI-2 was adapted to run at the Leibniz Supercomputing Centre's SUPERMUC: http://inside.hlrs.de/#advanced-one-sided-communication-patterns-with-gpi-2-anisotropic-diffusion-filtering-of-seismic-data
- Purdue
- REPSOL
- STATOIL

Execution test on platforms using a reasonable number of nodes/cores

GPI-2 is the base of the application "Parallel Edge- and Coherence-Enhancing Anisotropic Diffusion Filter (Parallel ECED)". This filter is used to reduce the noise of seismic data while preserving its layer structure at the same time. These are results on up to 32768 cores. This filter is used in the aforementioned application Pre-stack Pro. See http://www.gpi-site.com/gpi2/benchmarks/

Scalability

Most of the published benchmarks focus on throughput instead of scalability, and comparing GPI-2 with MPI. In their tests, it is consistently faster than MPI for small-scale experiments or small

messages. When the scale or message size increases, MPI approaches GPI's performance. See http://www.gpi-site.com/gpi2/benchmarks/.

2.5.4 Performance and Quality

Code quality

- Coherent usage of coding guidelines/standard: None can be found online
- Source code quality: The code is available publicly on GitHub: (https://github.com/cc-hpc-itwm/GPI-2/tree/next/src). The code itself is poorly documented, if at all. The API is documented at http://www.gpi-site.com/gpi2/gpi2-docs/annotated.html. A full audit would be needed to assess code quality.
- Existence of developer documentation: None could be found online
- Programming language uniformity: All the code, except for the FORTRAN bindings, is in C.
- Use of (automated) standardized (regression) tests: None could be found online. One of the "make" targets is "make tests", as can be seen at http://www.gpi-site.com/gpi2/gpi2-docs/index.html, section "Building GPI-2". These are however tests of the runtime, and there is no indication of an automated regression test procedure.

Ease-of-use

- Detailed feature description and user manual: GPI-2's website documentation describes the workings and provides a user manual: http://www.gpi-site.com/gpi2/docs
- Existence of examples and demos: Three simple examples of GPI-2's main features can be found at http://www.gpi-site.com/gpi2/examples
- Ease of installation, configuration, and usage: There is a quick install guide at http://www.gpi-site.com/gpi2/gpi2-docs/index.html

Testability

- Built-in distribution and installation checks: No
- Availability of test suites: Yes. "make tests" builds them, as explained in the aforementioned quick install guide.
- Availability of benchmark suites, benchmark data sets or micro-benchmarks: The "benchmarks" page shows results when compared with tools that perform similar functions, such as MPI: http://www.gpi-site.com/gpi2/benchmarks/

Fault-tolerant/resilience support

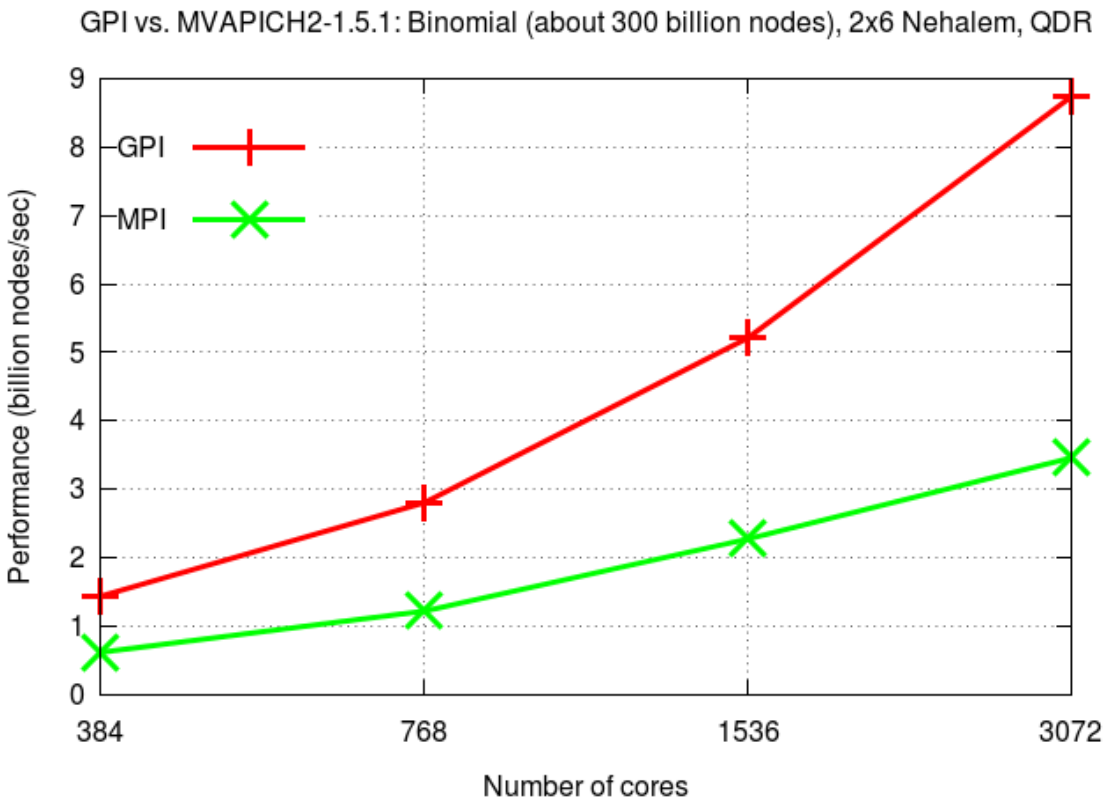
According to their FAQ (http://www.gpi-site.com/gpi2/faq/), GASPI – and its implementation GPI-2 - is failure tolerant in the sense that all non-local operations feature a timeout with a defined exit status. GPI-2 maintains a status vector with the current node status. If an error occurs and a node is lost, it is possible to either reduce the GASPI node set or to request a new node, e.g. from a pool of spare nodes. Application-level steps are nevertheless necessary, e.g. actual initialization from a previously written checkpoint. GPI-2 does not feature a fully automatic handling of failure tolerance via, for example, checkpoint/restart; GPI-2 does provide a minimal set of the required low-level functions.

Re-usability

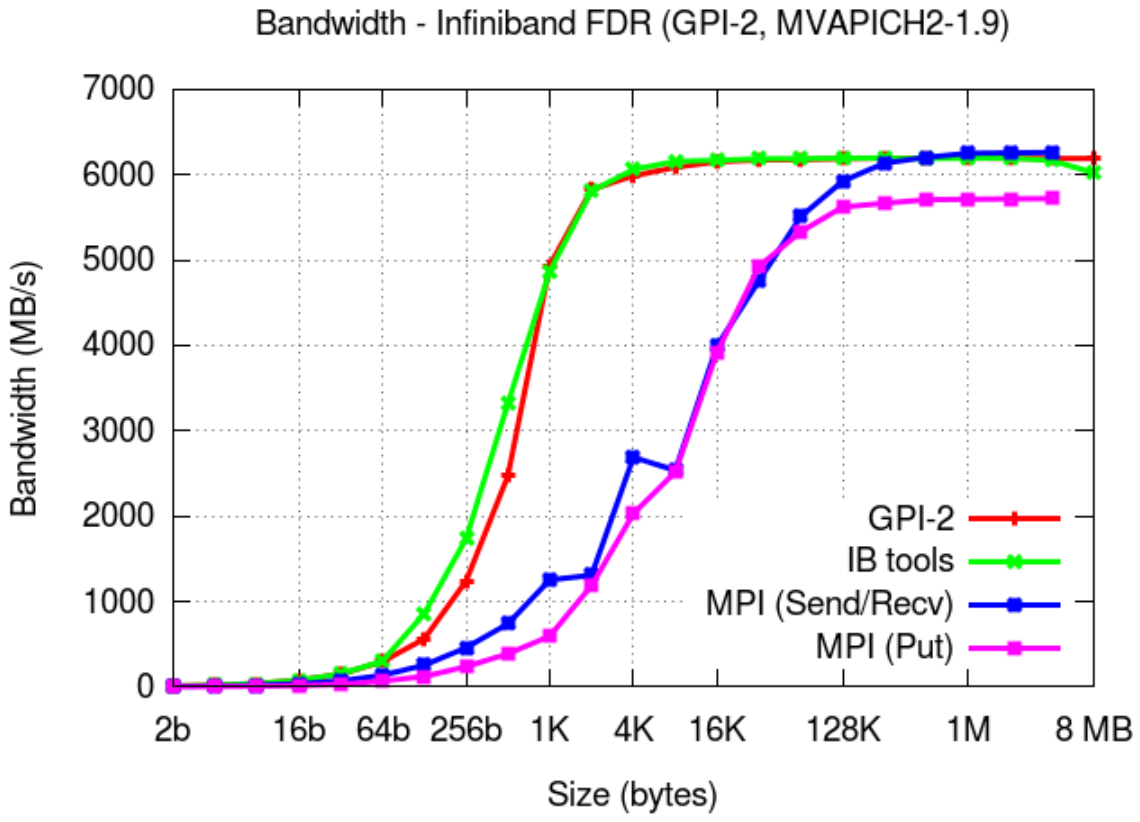
Migration from MPI to GPI-2 can be done incrementally, which allows one to re-use older MPI codes while providing a smooth migration path. Besides, GPI-2 is being used in a number of projects, as mentioned at the "Number of users/installations/component downloads" section.

Efficiency

In almost all benchmarks shown at http://www.gpi-site.com/gpi2/benchmarks, it has shown better throughput than MPI in similar conditions. In the Unbalanced Tree Search benchmark, for example, it performs almost 3 times more nodes than MPI:



However, when the message sizes start to increase, MPI's bandwidth seems to catch up, as in the following example:



User satisfaction

- List of organizations, testimonials, and other projects using this software: The predecessor to GPI-2, GPI, was created to close a gap in visualization applications. It is used, for example, at LBL, REPSOL, STATOIL, HLRS, ZIH, KIT, DLR, Fraunhofer, T-Systems, TU Dresden, Deutscher Wetterdienst, Jülich and Purdue (as stated at http://www.epigram-project.eu/wp-content/uploads/2013/11/gpi2.pdf).
- Case studies, usage histories:
  - Stencil-based industry applications, with REPSOL: http://www.itwm.fraunhofer.de/fileadmin/ITWM-Media/Abteilungen/CC_HPC/Pdf/hpc_flyer_GPI-Space-GPI-booster_for_stencil_based_ind.applications-EN.pdf
  - Other application examples: http://www.gpi-site.com/cms/sites/default/files/gpi_flyer_apps.pdf
  - The aforementioned pdf has several usage stories, such as BQCD codes, UTS, and 3D acoustic wave equation for reflection seismology intended to predict the fluid content of potential oil and gas reservoirs.
- Reports from forums, blogs, mailing lists, newsgroups, magazines, scientific articles : A Forum meeting happened in September, 2014: http://www.gpi-site.com/gpi2/gaspi-forum-meeting-2014
- Organized user community (e.g. annual user group meetings or BoFs): The GASPI specification and GPI-2 implementation are kept by a number of partners: http://www.gaspi.de/en/partners.html. Furthermore, GPI-2 has been chosen as the implementation of an “Exascale PGAS” for the EpiGRAM project: http://www.epigram-project.eu/project/.
- Quality of support services
  - Average response time: unknown
  - Average number of contacts with customer to resolve issue: unknown

2.6 Discussion of the Proposed Methodology

The goal of WP6 to “Develop and document a methodology for estimating the level of maturity of Exascale software components” was reached. In compliance with the international standard for the evaluation of software quality, ISO/IEC 25010:2011, a quality model for Exascale software components, consisting of a structured set of characteristics and sub-characteristics, was defined and documented in deliverable D6.1. The characteristics and sub-characteristics were developed in discussions with experts from EESI2 and constantly refined and improved during the lifetime of the project. In the last year of the project, the set of characteristics stabilized and no further improvements or changes were proposed by HPC experts asked for comments on the methodology.

As discussed in deliverable D6.1, the actual collection of the quality model factors, and their evaluation to determine the corresponding TRL could have been done in various ways:

- The simplest form would be that interested projects (“self-evaluation”) or external evaluators (“external evaluation”) document the quality model factor values in a simple document which follows the structure of the defined quality model and documents the values and answers for each item in text form.

  Then, based on the collected information and answers, the TRL is estimated for each of the major group of factors (Documentation and Support, Availability and Coverage, Portability and Scalability, and Performance and Quality). The overall TRL could then be determined by averaging the four sub values.
- Ideally, the input for the quality criteria factors would be collected with the help of a web-based online form, standardizing the input as much as possible, making it possibly to automatically evaluate the collected input (e.g. by scoring the answers) and the Exascale TRL could be automatically determined by creating a map between score ranges and TRLs.

Due to the limited amount of person months available for this work package, we chose the simpler form. Overall, the following observations became clear during the evaluation of the chosen three EU Exascale software components:

- Performing an actual evaluation by an HPC expert took about two to three days of work.
- In order to get experience on differences between a self-evaluation and external evaluations, we evaluated the three components in three different ways:
    1. Score-P was done as a self-evaluation.
    2. OmpSs was done as an external evaluation, but the developers of the component were given the evaluation with the request to fix errors and add missing information.
    3. GASPI was solely evaluated externally.

  Interestingly, there was not a big difference in time performing the evaluation and amount of content found when comparing the results for self-evaluations and external evaluations. The quality was also not a big issue, as in the second case, the developers did only correct a few minor items in the external evaluation (mostly wrong spellings) and also only added a few extra information pieces, all of them not easily available from public sources. This makes us believe that given an reasonable well experienced HPC expert, she or she can create a high-quality evaluation.
- There were a few characteristics which were hard to determine, for example
    - The “Number of users / installations / component downloads” are hard to determine for open-source projects. Most (if not all) of them to not track downloads in a way that the number of downloads by different persons or organisations can be determined. Even if a rough number would be known, it is unclear how many of them actually succeeded in the installation of the component, and if successful, how many different users actually used this installation. For example, most of the considered components are installed at large HPC centres like the PRACE Tier-0 centers with hundreds of users. However, in this situation, it is unknown how many of them actually used the components.
    - It is unclear (and was a point of intense discussions with the EESI2 experts) how to judge the quality (and not just the quantity) of documentation or of the support service. While it is clear, that if more documentation is available (user guide, installing guide, FAQs, etc) it can be sign of a more mature project, but it is also clear that the quality cannot derived from the length (e.g. number of pages). More research in that area is needed.
    - Originally unexpected, the evaluation of the criteria “Execution test on platforms using a reasonable number of nodes/cores (1024 nodes?)” turned out much more difficult than expected. By the end of the project, we still had not found a reasonable way how to define suitable test cases for Application codes (AC), Libraries and Frameworks (LF) and programming Model implementations (PM). They can be tested if test cases are provided with the component package, but of course this makes them hard to compare between components, and leaves it unclear what to do if the developers do not provide test cases.

      Interestingly, even the evaluation of Development Tools (DT) turned out much more complicated than expected. Here, the experts agreed very early that applying the tools to a well-defined benchmark set would provide a suitable test and that the Unified European Application Benchmark Suite (aka PRACE benchmarks) would be the ideal candidate. It turned out that the benchmark codes available at the PRACE website were outdated and especially the installation and usage instructions were incomplete and sometimes even wrong. It took an experienced HPC software engineer who had done such evaluations before with other benchmark suites over a full month to evaluate Score-P with the benchmarks. Most of the time was spent on

figuring out on how to install and use the benchmarks, not so much on the actual evaluation of the development tools.[1]

- Also, no agreement could be reached among the involved EESI2 experts on how to "translate" the evaluation results into the Technology Readiness Level (TRL) for Exascale Software as defined by deliverable D6.1. Although, we strongly believe that the evaluation reports as documented in this deliverable are still useful to the developers and the HPC community.

[1] We contacted the original person of contact for the PRACE benchmark set and got told, that because the current PRACE implementation project has no funding maintaining the benchmarks, they do not see a way of keeping the benchmarks up-to-date with current HPC platforms, not even for the PRACE Tier-0 architectures. This again is a prime example, why an institution like the proposed EESC is necessary or an alternative way of making the EU enormous HPC software investments sustainable.

3. Proposed Structure of European Extreme-Scale Software Centre (EESC)

The European Union will spent over 700 million € on research in high performance computing in the current decade. As the EU in these programs does not support the acquisition and installation of HPC hardware, the largest part of this investment is in system and application software developed as part of research and innovation and other funded actions. Many projects publish the resulting software as Open Source in some form. Although desired, commercial exploitation of these software products rarely happens. A big problem in this context is the long-term maintenance of the software components: once, the project ends, the developers might not have enough resources to maintain the software and adapt them to new platforms and architectures. So, as the hardware and software platforms in HPC change and improve much more quickly than standard IT platforms, the software components whose development was paid for with typically quite large project funds, gets outdated and therefore useless very quickly.

As one possible solution for this dilemma, the first phase of the European Exascale Software Initiative (EESI1) recommended to establish a European Exascale Software Centre to coordinate research, development, testing, and validation of EU HPC Exascale software ecosystem components and modules. As there would be never enough funds to maintain all software components developed in HPC EU projects, an independent (external to the project) evaluation of quality of project software is needed to select possible candidates and the results of this project documented in the deliverables 6.1 and 6.2 could be used as the basis for such an evaluation. Finally, a public available catalog of externally certified HPC software components would simplify and enhance the trust in and reuse of these components in the HPC community and industry.

3.1 Recommendations of EESI2 WP6

The experts in work package 6 of EESI2 make the following recommendations:

- R6.1: An independent EESC (European Extreme-scale Software Center) should be established (as already recommended by the first phase of the EESI project).
- R6.2: To simplify the funding of the EESC, we recommend to allow H2020 projects (and ideally other types of EU and National projects) to pay for EESC services out of their projects funds, if desired by the project partners.
- R6.3: The creation, original funding, and monitoring of EESC should be the responsibility of the HPC Private-Public Partnership (i.e., by ETP4HPC).

3.2 European Extreme-scale Software Centre

In the following, the envisioned goals, tasks, overall structure, possible funding and access model of such an EESC is further detailed.

3.2.1 EESC Goals and Tasks

The main goal of the EESC is to facilitate and ensure the technology transfer of the HPC system and application software developed in EU H2020 projects into industrial and HPC community use. For this, the EESC should fund, organize and monitor the technological transfer of EU HPC projects into maintained and trustable code for scientists and industry.

In detail, we envision the following tasks for the EESC:

- To coordinate the research, development, testing, assessment, and validation of EU Open Source HPC Extreme-scale software ecosystem components and modules
- To collect, catalog, and maintain software for the HPC community (especially if the productization or another commercial exploitation is impossible or undesirable).

- To specify, maintain, and promote APIs developed inside EU projects shared at the international level, and, if possible work for and with the projects to standardize them in international standardization bodies.
- To promote and organize links with vendors, ISVs and service providers, for example, by providing support for software startups.

3.2.2 EESC Structure / Governance

We strongly recommend to structure, manage, and staff the EESC as a professional service, not just a research activity. Parts of its tasks might be research activities, but they only should be done within the context of supporting the service provision. The actual implementation could be by either research institutions or professional (non-profit or commercial) organizations or a combination of the two. Experience (e.g. from the operation of the UK SSI and HECToR CSE services) indicates that the best model will be a professional managing coordinator which then collaborates with research institutions via subcontracts to deliver its services.

3.2.3 EESC Funding

It is clear that such an EESC will not be successful without proper **and** sustainable funding, that is, enough funding to do a proper job. As the main goal is to maintain HPC community codes beyond the lifetime of typical research projects (two to three years), the challenge will be to ensure a sustainable funding as otherwise it will be difficult to establish the necessary trust of the potential customers into the EESC.

Initially, to kick-start the efforts, the EESC could be funded as a H2020 horizontal Centre of Excellence (CoE). Of course, proposals for such a call would need to present an initial business plan for the sustainable operation of the center and implement it at the end of project. One base for a sustainable funding could be via the selling of services, where companies and HPC Centers (with their own sustainable budget) pay for the use of the services, while EU and national projects code teams could pay for the services with their project funding (→ see recommendation R6.2 above) if desired and they cannot organize another way to ensure long-term maintenance of the software project outcomes. Initial target customers would be the current H2020 vertical CoEs and FETHPC projects. We expect that costs on the order of tens of millions of Euros per year for the operation of EESC.

3.2.4 EESC Customer Access Model

The primary way potential customers for the EESC would be a customer-led interaction, i.e., the customers approach the center for assessment or certification services. However, if funding and resources allow, the EESC could also proactively approach new customers based on input from a HPC community steering board. For example, work on key application codes that have a high utilization on publicly-funded HPC systems seem to be good candidates.

3.2.5 EESC: Other Thoughts

- EESC should have a clear mission to provide support to academia and industry alike.
- The center should not only have an ambition to assess and certify HPC software components, but also
    - software development entities (e.g. research groups) and their processes,
    - developers,
    - and training courses.

- EESC should synchronize with the currently established H2020 CoE POP[2] for providing performance assessment services. If POP, as planned, develops into a self-sustained service, it could be potentially merged with the EESC once established.
- The center could be the focus point for small distributed software development communities and should help coordinate their efforts.
- EESC should provide training and support service on proper software engineering techniques and tools, e.g., by producing best practice guides and training material.
- EESC should provide legal support for EU projects helping them select the right Open Source license model.
- EESC should create and maintain an open accessible catalog of EU HPC software components (which also includes the maturity and performance assessments). Experts suggest that EESC could in addition provide an EU central Open Source Project Hosting (a la GitHub?) so it is not necessary to rely on an U.S. owned organization to host the sources of important EU software projects.
- EESC should promote common EU open source components and coordinate their developments in cooperation with ETP4HPC.
- Finally, the center should have the ambition that production-scale access to the EU's largest publicly-funded supercomputers will require the codes to have been assessed by EESC and/or POP. For this task, and also to independently be able to assess code maturity, it should coordinate with but be organizationally independent from PRACE.

[2] http://www.pop-coe.eu

4. Conclusions

This deliverable reports the results of the work of the project partners on the objectives of EESI2 Task 6.2 (“Perform evaluation on 3 components”) to (a) identify 3 software stack components from existing and near future European Exascale projects and apply the defined methodology and (b) examine existing “equivalent" centers and propose a structure adapted to Exascale software as defined by deliverable D6.1.

First, after describing the three selected Exascale software components and the criteria used in the selection, it documents the results of a pilot evaluation of the level of maturity of these three components following the methodology defined in EESI2 deliverable D6.1. To gain experience and being able to access the quality and efficiency of the evaluation, the process was performed in three different ways:

1. A self-evaluation of an component done by the developers of the software themselves
2. An external evaluation of an component where the results and accuracy of the evaluation was cross-checked by the component developers
3. A pure external evaluation

The simple pilot study did not find any major differences in the outcomes of the evaluation resulting from using the different evaluation forms. Further issues and feedback regarding the evaluation methodology are also documented.

Secondly, it recommends three actions including strongly reinforcing the recommendation for the creation of a European Extreme-scale Software Centre (EESC). Finally, it suggests first ideas on a structure and funding model for such a center.

Of course, the pilot evaluation study of three EU software components and the initial suggestions on structuring and funding an European Extreme-scale Software Centre are just first steps. More work is needed to finalize the HPC software maturity evaluation procedure, for example by finding solutions for the open issues documented in Section 2.6. Especially, it is still unclear how to map the results of the evaluation into an EU HPC Software Maturity TRL as defined in deliverable 6.1.

