GA no. 671657

READEX

Runtime Exploitation of Application Dynamism
for Energy-efficient eXascale computing

Periodic Technical Report
Part B

Period covered by the report	From 01/03/2017 to 31/08/2018
Periodic report	2nd
Version	1.0

Document history:

Version	Date	Author/Editor	Description
1.0	**30.10.2018**	**Robert Schöne, Katja Boettcher**	Fixed typo, update of Sec 5
0.61	10.10.2018	Robert Schöne, Othman Bouzi, Marie-Christine Sawley	Comments, updated figure
0.6	09.10.2018	**Katja Boettcher**	Section 5
0.5	09.10.2018	**Robert Schöne,** Michael Gerndt, Per Gunnar Kjeldsberg, Kai Diethelm	Added comments
0.41	05.10.2018	**Robert Schöne**	Fixed comments
0.4	26.09.2018	**Robert Schöne,** Katja Böttcher	Include input by Katja
0.3	11.09.2018	**Robert Schöne**, Andreas Gocht, Michael Gerndt	Objectives (input from D5.3, D6.6), Q-Learning, Section 2, annotations by MG
0.21	16.08.2018	**Robert Schöne,** Per Gunnar Kjeldsberg, Kai Diethelm	Objectives
0.2	14.08.2018	**Robert Schöne,** Per Gunnar Kjeldsberg, Kai Diethelm	Changes due to first review
0.1	24.07.2018	**Umbreen Mian**	Final version for First Review
0.06	19.07.2018	**Robert Schöne**, Per Gunnar Kjeldsberg, Andreas Gocht, Umbreen Mian, Venkatesh Kannan, Lubomír Říha, Madhura Kumaraswamy, Michael Gerndt	Second batch of submissions
0.05	18.07.2018	**Robert Schöne**, Per Gunnar Kjeldsberg, Andreas Gocht, Umbreen Mian, Venkatesh Kannan	First submissions
0.01	08.06.2018	**Robert Schöne**	Initial version

Table of content

List of figures

List of tables

Glossary

ACP	see Application Configuration Parameter
Application Configuration Parameter	Such a tuning parameter is located in input/configuration files of an application
ATM	see Application Tuning Model
ATP	see Application-level Tuning Parameter
Application Tuning Model	During DTA, the application is automatically analysed for dynamism, optimal configurations are determined for the different code regions. These are classified as scenarios and stored in an Application Tuning Model.
Application-level Tuning Parameter	Parameter that determines an execution decision in the application, e.g., a selectable code-path
Design-time Analysis	The analysis of region and phase characteristics from application runs before the application is run in production
DK	see Domain Knowledge
DKSI	see Domain Knowledge Specification Interface
Domain Knowledge	The knowledge of a developer related to the application structure, the application characteristics, and specific application-level tuning parameters.
Domain-level Knowledge Specification Interface	The software interface that enables users to use Domain Knowledge
DTA	see Design-time Analysis
DVFS	see Dynamic Voltage and Frequency Scaling
Dynamic Voltage and Frequency Scaling	The ability of a set of processor components to change voltage and frequency at runtime
Dynamic Tuning	Usage of different configurations for an application depending on the current phase or region that is executed
Dynamism	Defines whether significant regions within an application have different optimal configurations. The more different the configurations, i.e. the more different the settings of single tuning parameters, the higher the dynamism
HDEEM	A power measurement infrastructure designed and implemented by Bull SAS (Atos technologies) and Technische Universität Dresden
Identifier	A piece of information that is used to classify a Runtime Situation
MPI	A programming interface for message-passing-based process-parallel applications
OpenMP	A programming paradigm for thread-parallel applications
Periscope Tuning Framework	A framework for automatic tuning of large-scale parallel applications
PTF	see Periscope Tuning Framework
RADAR	see READEX Application Dynamism Analysis Report
RAT	see Runtime Application Tuning
Runtime Application Tuning	The improvement of energy efficiency at runtime via switching of tuning parameters
READEX Application Dynamism Analysis Report	A report that describes application dynamism found by manual tuning

READEX Runtime Library	A library that applies tuned configurations at runtime based on the tuning model created in DTA. Also used in DTA to set parameters during the search for optimal configurations.
RRL	see READEX Runtime Library
RTS	see Runtime Situation
Runtime Situation	An instance of a significant region
Score-P	A scalable performance monitoring library that supports various forms of instrumentation, monitoring extensions, and measurement back-ends
Significant region	A region of code within an application whose execution time is high enough to justify switching the configuration
Static Tuning	Applying a configuration that deviates from the default one for the whole application run
Tuning Parameter	A hardware or software switch that can be changed and potentially influences the energy efficiency of a computing system

1. Explanation of the work carried out by the beneficiaries and overview of the progress

1.1. Objectives

The READEX project identified five objectives that are at the core of the project. These are listed in Section 1.1 of the DoA:

- Static Energy Efficiency Tuning (O1)
- Manual Dynamic Energy Efficiency Tuning (O2)
- Integrated Tool Suite for Dynamic Auto-Tuning (O3)
- Novel Programming Paradigm (O4)
- Use of READEX Technologies (O5)

Below, we describe the work executed to achieve these objectives.

1.1.1. O1: Static Energy Efficiency Tuning

This objective aims at exploring the effects of optimizing parameter settings for whole application runs (static tuning). In the READEX project, we implemented a dynamic auto-tuning approach and compared it against the static base line. Static tuning can be done either manually or by using the original approach of the Periscope Tuning Framework (PTF). We used the former for our comparison, which we lay out in detail in D5.3.

For static tuning, we used components that we developed in WP1 (handling of the parameters) and WP5 (manual tuning). We used multiple applications, as described in D5.3, for a comparison. In the proposal, we expected ***typically +10% static savings in comparison to the non-optimized execution***.

Table 1: Manual static tuning results

(Intel Compiler, Intel MPI. Haswell partition of TUD Top500 system Taurus
***gcc measurement, since Intel compilers did not work for this software)**

Software	Static tuning savings
AMG2013	12.5 %
Blasbench	7.4 %
Kripke	11.5 %
Lulesh	17.6 %
NPB3.3	11 %
BEM4I	15.7 %
INDEED	17.6%
ESPRESO*	4.3 %
OpenFOAM*	15.9 %
Average	12.6 %

1.1.2. O2: Manual Dynamic Energy Efficiency Tuning

This objective is an extension of O1: Here, we extended the analysis to dynamic parameter settings, i.e., switching parameters during the application run. The objective of this task is to find optimal system configurations for significant regions of the applications, i.e., regions that are running long enough to justify switching of configuration.

We documented the manual tuning results in D5.3. The measure of success has been ***up to 20 percentage points improvement on top of the state-of-the-art static tuning results***.

Here, the results vary significantly, depending on the dynamism of the application. The application BEM4I reaches the highest dynamic savings with 18 % on top of the 15.7 % static savings.

Table 2: Manual dynamic tuning results

(Intel Compiler, Intel MPI. Haswell partition of TUD Top500 system Taurus;
***gcc measurement, since Intel compilers did not work for this software)**

Software	Manual dynamic tuning savings
AMG2013	0 percentage points
Blasbench	+7.9 percentage points
Kripke	+7.1 percentage points
Lulesh	0 percentage points
NPB3.3	0 percentage points
BEM4I	+18.4 percentage points
INDEED	+1.9 percentage points
ESPRESO*	+3.9 percentage points
OpenFOAM*	+4.2 percentage points

1.1.3. O3: Integrated Tool Suite for Dynamic Auto-Tuning

The design and development of an integrated Tool Suite for dynamic auto-tuning is at the core of the READEX project. With this Tool Suite, users are able to perform automated optimization for energy-efficiency through a workflow tool that combines *Design-time Analysis (DTA*) and runtime tuning (*Runtime Application Tuning, RAT*). In a first step, the READEX Tool Suite is capable of analysing applications for dynamism. If there is no sufficient dynamism, READEX can still make use of static tuning. Then, PTF determines optimal system configurations for different code regions. At the end of DTA, PTF generates a tuning model (*Application Tuning Model, ATM*), which the lightweight *READEX Runtime Library (RRL)* uses to tune parameters during runtime.

In Deliverable D4.1, we created a formalism for automated tuning splitting it in design-time and runtime tuning. We drafted an initial design for the implementation in the same document. Based on this design, we implemented the required extensions to PTF, Score-P, and the RRL. Furthermore, we established a strategy for the integration of the different components, which includes software quality assurance measures and an approach to software management (WP4). We also investigated potential parameters and implemented ways for controlling these parameters (WP1).

We made the results available to the public, starting with a pre-alpha in M12 at our development system Taurus, which is located at TUD. Since M30, we published four beta prototypes at regular intervals, which users could download from our website. In M36, we created a prototype reference implementation that is now available for download. The measures of success for this objective was ***an energy efficiency improvement at least 50% of the manual dynamic tuning results and a 90% reduction in programming effort in comparison to manual tuning***.

Table 3: Manual tuning results vs. READEX automatized tuning results

(Intel Compiler, Intel MPI. Haswell partition of TUD Top500 system Taurus;
***:gcc measurement, since Intel compilers did not work for this software**
****: This benchmark needed domain knowledge to exploit the tuning potential)**

Software	Overall manual tuning savings	READEX automatic savings	READEX compared to manual
AMG2013	12.5 %	7.0 %	56 %
Blasbench	15.3 %	9.9 %	64.7 %
Kripke	18.5 %	10.5 %	56.8 %
Lulesh	18.7 %	18.2 %	97.3 %
NPB3.3**	11.0 %	0.0 %	0.0 %
BEM4I	34.1 %	34.0 %	99.7 %
INDEED	19.5 %	19.1 %	97.9 %
ESPRESO*	8.2 %	7.1 %	86.6 %
OpenFOAM*	20.1 %	9.8 %	48.8 %
Average	17.5 %	12.8 %	73.2 %

In Table 3, we show the energy savings achieved with READEX and compare it to the manual approaches. Here, READEX achieves **more than 70 % of the manually achieved savings in comparison to the targeted 50 %**.

As we discuss in Deliverable D5.3, we had to implement supportive routines to be able to tune the applications manually, i.e. for accessing the energy measurements and hardware parameters. Without these supportive routines, it would not have been possible to tune the applications manually. While applying READEX took between two and eight days for the applications, the implementation of the supportive routines took significantly longer than ten weeks. This does not even count the performance monitoring to detect significant regions and the instrumentation itself. Therefore, we **reduced the programming effort by more than 90%** in comparison to the manual approach.

1.1.4. O4: Novel Programming Paradigm

The programming paradigm developed in the READEX project allows users to provide application *Domain Knowledge* (DK) to the dynamic auto-tuning process in order to improve the dynamism detection and to facilitate the exploitation of *Application-level Tuning Parameters* (ATPs).

In the second reporting period, we implemented the concepts for DK, including user-level code regions (i.e., region, phase, and input identifiers) and ATPs. We described the latter in more detail in Deliverable D1.2.

We furthermore used Score-P interfaces to annotate phases, regions, and parameters within the program. We did this for the reason of standardization. The annotations can now also be used for profiling and tracing the application or to use the data for other mechanisms besides of READEX.

The measures of success for this objective were ***an energy efficiency improvement of up to 50% of the automatically achieved dynamic tuning on top of the automatic dynamic tuning results and a programming effort reduction of 60% in comparison to manual tuning*.**

With the usage of the Domain-level Knowledge Specification Interface (DKSI) on the NPB application, we were able to push the average savings for hardware and system software parameters to **more than 80 % of the manual savings (in average over the whole Test Suite), as compared to the 75 %, which we targeted**. The instrumentation needed 10 minutes by an experienced developer.

Not all applications benefit from an optimization with hardware and runtime parameters. Therefore, we also tested selected applications for ATPs. This complies with the task given to us in the review for the first project period. Here, the reviewers wished for ***emphasis […] on applications with significant tuneable application parameters. These have greater potential to deliver significant energy performance increases than the system parameters, which have been the primary focus so far.*** Therefore, we extended the list of applications. With the implementation of ATPs and Application Configuration Parameters (ACPs), we were able to **save 33.25 % energy** in average in comparison to a reasonable default.

1.1.5. O5: Use of READEX Technologies

The READEX project aimed at achieving a high user uptake of the methodology and the installation of the Tool Suite on at least 4% of the European systems in the Top500 list from June 2014. However, some of the systems are not available anymore and others are still restricting important software interfaces. We therefore also supported the installation and usage of READEX software on clusters of seven sites with European Top500 installations from June 2018. These represent 9.7% of the European Top500 sites from June 2018. We furthermore supported the installation of READEX technology on smaller scale systems and the usage of READEX technology (for example, the support of alternative measurement interfaces).

As said before, providing access to hardware power saving mechanisms is not as common as predicted. To broaden the applicability of READEX, the Tool Suite now provides support for multiple common interfaces for hardware parameters[1] and energy measurement[2]. Furthermore, we described how alternative energy measurement interfaces can be used for Score-P and READEX [1]. For this purpose, we implemented significant portions of READEX in Score-P that are now part of every Score-P installation. Therefore, users can easily enable mechanisms like the Runtime Calibration by just downloading and installing the respective plugin from github and registering them as a new plugin. The interface that we provide makes it also possible to implement alternative optimization or measurement plugins [1].

The measure of success for this objective was that ***the READEX Tool Suite and Programming Paradigm are used at up to 4% of the European supercomputers on the Top500 list from June 2014*.**

[1] https://github.com/readex-eu/libfreqgen/blob/master/README.md
[2] https://github.com/score-p/scorep_plugin_x86_energy

Currently, READEX is installed and used at four June 2014 Top500 installations: Viljie (NTNU), Taurus (TUD), SuperMUC (LRZ), and Salomon (IT4I). We also supported the installation and usage at systems that were listed in newer releases of the Top500 list, since some of the older ones were not available anymore. This includes partitions on JURECA (JSC), Mistral (DKRZ), and Hazel Hen (HLRS). **Based on the number of 117 European Top500 systems in June 2014, this represents 5.9%**. We furthermore supported other sites with installing and using READEX (e.g., KTH, Hartree, BSC) and plan to do so in the future. Figure 1 shows an overview of usage of READEX software components. We provide more details in Deliverable D6.6.

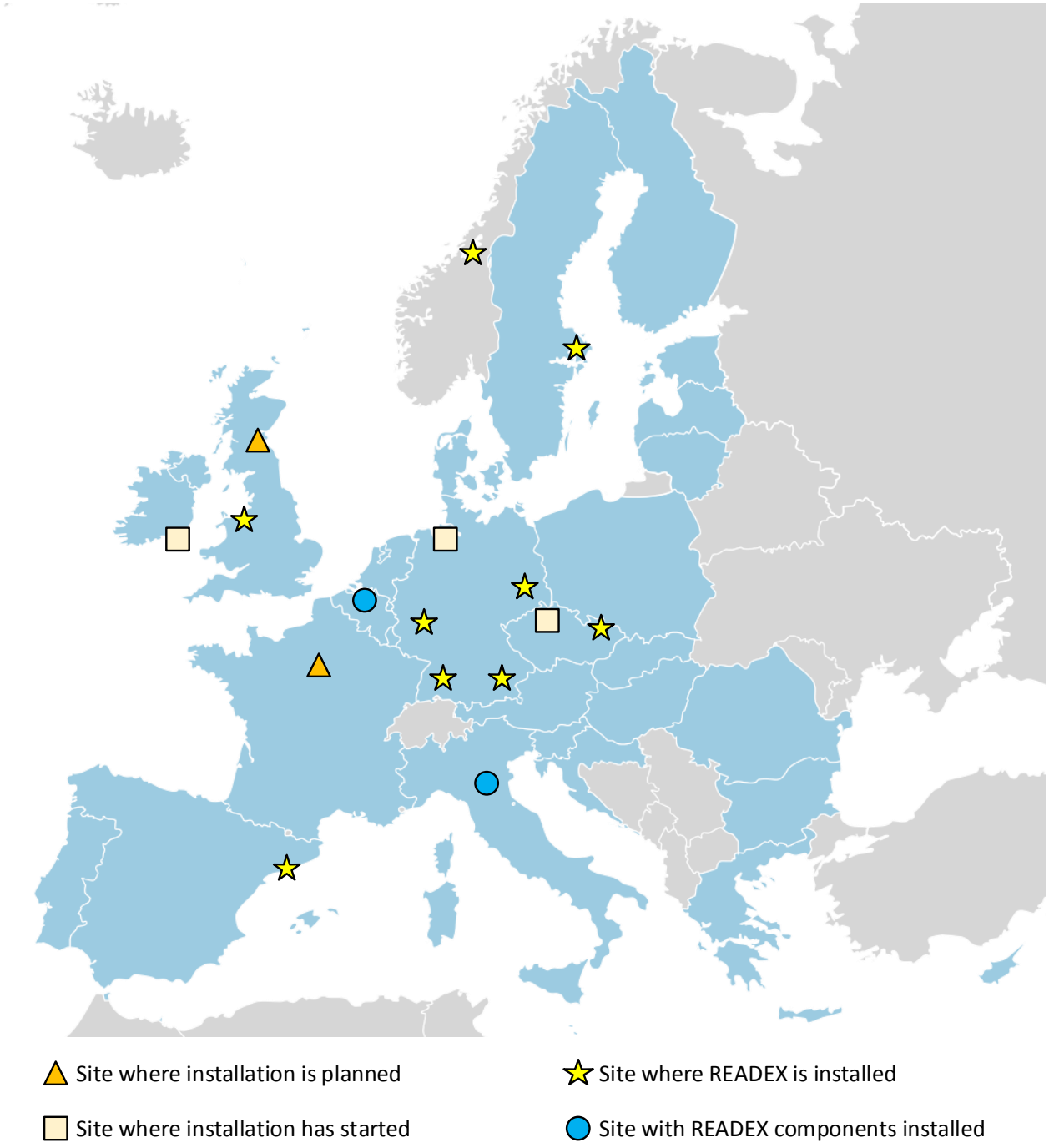


Figure 1: Installation status of READEX software in Europe

1.2. Explanation of the work carried out per WP

1.2.1. WP1 Tuning Parameters

The focus of this work package was the investigation of potential tuning parameters and the creation of tuning plugins for PTF and Score-P. On the PTF side, these plugins provide a set of reasonable parameter values to the tuning system to find the optimal setting. The plugins have access to profiling data and thus can employ expert knowledge to choose from the set of all values of a specific parameter. On the Score-P side, the plugins are responsible for controlling the actual parameter setting process. Therefore, they are called *Parameter Control Plugins (PCPs)*. As an example, the plugin that controls the processor frequency is responsible for issuing system calls to set the frequency of all (or a subset of) CPUs to a requested value. The plugin is controlled by the RRL in Score-P, which receives so-called tuning commands from PTF through Score-P's Online Access interface. These commands specify the value of the parameter to set while the application is in a specific state, i.e., based on the call-path. Moreover, the project partners conducted the integration of various energy measurement interfaces in Score-P in this work package. TUD led the work package with support from Intel, NUIG, and IT4I.

Main objectives of the work packages for the second reporting period were:

- ✓ The finalisation of application parameter tuning
- ✓ Investigation on future tuning parameters

Major results of the work package during the second reporting period are:

- The finalisation of an ATP interface
- The investigation of possible future parameters that are worth evaluating
- The provision of libraries for hardware parameters that support a broader variety of interfaces

*Task 1.1 Investigate hardware parameters (**TUD**, INTEL, M01 - M12)*

This task already finished in the first reporting period. We described the findings and implementations of this task in Deliverables D1.1 and D1.2. However, to broaden the applicability of READEX, TUD implemented a library that allows users to use the most common interfaces for core and uncore frequency tuning.

*Task 1.2 Investigate application parameters (**Intel**, IT4I, NUIG, M01 – M24)*

Most of this task finished in the first reporting period. We described the findings and implementation of this task in Deliverable D1.2. In addition to reported tasks, we fully completed the ATP library and ATP server and tested ATPs with PTF and RRL.

*Task 1.3 Investigate runtime system parameters (**TUD**, IT4I, Intel, M01 – M12)*

This task finished in the first reporting period. We described the findings and implementations of this task in Deliverables D1.1 and D1.2.

*Task 1.4 Future Parameter Investigation (**NUIG**, TUD, INTEL, M25 - M36)*

The objective of this task was to investigate and identify potential tuning parameters that were not targeted in the READEX project but may be available on future and emerging extreme scale systems. NUIG, TUD and Intel investigated possible tuning parameters that may be available in the network hardware, future (co-)processor hardware, memory subsystems, operating systems and run-time system hardware.

We investigated the network hardware related parameters that are currently available in the Intel OmniPath and Mellanox Infiniband infrastructures. Our criterion for these parameters was that they influence the performance and/or energy consumption of the network fabric and therefore implicitly also for the application. At the time of writing, these parameters were only available for static setting, where network administrators can configure them depending on application characteristics. These characteristics could be, for example, single-node or multi-node application, single-threaded or multi-threaded applications, communication workload imparted by the application, sizes of buffers exchanged by processes in an application, and configuring the use of network interface cards by processor sockets for communication. We determined the different possible values for the parameters and identified their influence on applications. On the network software side, we investigated the Intel MPI library to determine runtime parameters that could influence the applications performance. The main parameters that can control the behaviour of the MPI runtime are the I_MPI_ADJUST parameters, which control algorithms for collective operations. On top of that, we investigated parameters that control the behaviour of the fabric and parameters (I_MPI_HBW_POLICY, I_MPI_BIND) for memory placement.

For processor hardware, we developed a SLURM plugin for changing NVIDIA GPGPU hardware parameters. With this infrastructure available, we were able to measure the influences of GPU core and memory frequencies on different benchmarks. Furthermore, we analysed a newer processor architecture of Intel to verify whether our approaches will still work for future architectures. We also analysed the availability of new power measurement interfaces for newer operating systems and architectures and extended the power measurement plugins accordingly[3]. Furthermore, we investigated the usage of AVX instructions and their influence on execution time and power consumption of the application. Here, users can use the ATP library to use the most energy efficient instruction set for their applications.

We also investigated High Bandwidth Memory (HBM), which is used on Intel Xeon Phi processors and FPGA boards. Users can either use HBM as a cache for the DRAM or place data explicitly in HBM, bypassing DRAM. Optane memory can also be used to extend the memory pool of the application or as a cache between storage and DRAM. The tuning parameters that could be used in READEX for the memory subsystems could be the placement data structures in the memories.

We provided a more detailed analysis in Deliverable D1.3.

1.2.2. WP2: Design-time analysis

The focus of this work package was the semi-automatic analysis of the application characteristics to determine optimized system configurations for the application's *Runtime Situations (RTS)*. This analysis is carried out at design-time, i.e., before the application runs in production, and leads to the creation of an Application Tuning Model (ATM). This ATM is used later by the READEX Runtime Library (RRL, WP3), which dynamically adapts the system configuration in order to increase the application's energy efficiency.

This work package was led by TUM and involves contributions from TUD, NTNU, IT4I and NUIG.

Main objectives for the second reporting period were:

- ✓ The finalisation of the Periscope Tuning Framework (PTF) for READEX purposes
- ✓ The finalisation of intra-phase dynamism handling
- ✓ The finalisation of visualization activities for dynamism and optimization
- ✓ The extension of the analysis with domain knowledge

[3] https://github.com/tud-zih-energy/x86_energy/tree/v_2

- ✓ The provision of PTF versions for regression testing
- ✓ The support of inter-phase dynamism

Major results of the work package during the second reporting period are:

- The extension of the readex_intraphase tuning plugin. It now analyses the application for dynamism inside a phase. During this period, we added support for ATPs, and input identifiers.
- The finalization of the tuning model visualization and runtime switching visualization with Vampir
- The implementation of domain knowledge support and its integration into the intra-phase analysis.
- The continuous provision of updated DTA tools
- The implementation of the readex_interphase tuning plugin and its integration with RAT
- The implementation of a new tuning plugin called readex_configuration, which allows application experts to tune Application Configuration Parameters that manifest in the input files of the application.

*Task 2.1: Scenario identification (**NTNU**, IT4I, TUM, M1-M30)*

This task focused on generating the ATM based on the analysis results from the pre-computation described in Task 2.2. In the First Periodic Report, we described the work on developing a formal mathematical description and common understanding of the READEX concepts as reported in Deliverable D4.1. We have also described the work on the first scenario identification mechanism based on clustering of runtime situations with identical configurations in Deliverable D4.2.

In the second half of the project, we extended the scenario identification mechanism to perform grouping of runtime situations based on configuration similarity. We also enabled users to perform the pre-computation of configurations multiple times with different application input, which are described through input identifiers. In such cases, the scenario identification includes a merging of tuning models. Finally, we added support for inter-phase tuning during the second half of the project. We described all of this work in Deliverable D2.3, with additional information in Deliverables D4.3 and D4.4.

*Task 2.2: Pre-Computation of configurations (**TUM**, NTNU, M1-M30)*

The pre-computation of configurations determines the best settings for tuning parameters, which are then stored in an ATM by the scenario identification described above. This pre-computation of configurations is carried out by tuning plugins of PTF, exploiting intra-phase and inter-phase dynamism via efficient search methods. The analysis carried out by the provided plugins focuses on assessing configurations during individual program phases and offers different search algorithms so that exhaustive search with full application runs for each configuration are avoided. This is a pre-condition for handling long running real world applications.

In the reporting period, we extended the readex_intraphase plugin to support the READEX *Domain-level Knowledge Specification Interface (DKSI),* which provides means to specify the application structure, application characteristics, and ATPs, as documented in Deliverable D4.5. We extended the supported objectives with normalized versions, which enables users to tune applications where only the execution time of significant regions changes but not their characteristics. We also implemented support for input parameters, which enables users to generate different tuning models for different inputs. The READEX Tool Suite is now able to merge these tuning models into a single tuning model, as described in Task 2.1. We created a new PTF plugin for handling inter-phase dynamism. This plugin uses a novel tuning approach

for applications with phase characteristics that change over the sequence of phases. It applies experiments with a random search strategy and collects the objective values and the phase features for each phase. The phase features capture the characteristics of the phases and allow a clustering of these based on similarity. The used mechanism determines the best configurations for the phase region and for the individual RTSs based on the clusters. We documented both plugins in Deliverable D2.3.

On request of the application experts in the READEX project, we created an additional tuning plugin, which supports static tuning of *Application Configuration Parameters (ACP)*. ACPs are tuning parameters in the input files of the application. Therefore, they can only be tuned statically. The readex_configuration tuning plugin provides a flexible configuration file for the specification of ACPs and their value ranges. The plugin identifies input file templates in which the ACP names are replaced by the value chosen during DTA. DTA then searches the design space to create configurations and assesses each configuration by copying the input template files. Afterwards, the application is restarted and either a single phase or, if specified by the user, multiple phases are executed, and the objective values are measured. Finally, DTA writes the optimal configuration into the input files and into a result file. After this static optimization, either the intra-phase or the inter-phase plugin is executed. We document the usage of the readex_configuration tuning plugin in the user's guide, which is part of Deliverable D4.4.

*Task 2.3: PTF analysis of tuning potential (**TUM**, NTNU, NUIG, M7-M12)*

This task already finished in M12. We described it in Deliverables D2.1 and D7.2.

*Task 2.4: Visualization of application dynamism (**TUD**, TUM, M13-M24)*

This task focused on providing ways to visualize application dynamism. Users can do that at different stages of the READEX tuning process. First, the `readex_dyn_detect` tool, which we developed under Task 2.3, shows the application dynamism in terms of execution time dynamism and compute intensity dynamism as text. Here, we implemented another way to visualize the dynamism at this stage. A script enhances an existing trace of an application by the dynamism information given to enable a user to examine the behaviour more graphically, as shown in Figure 2.

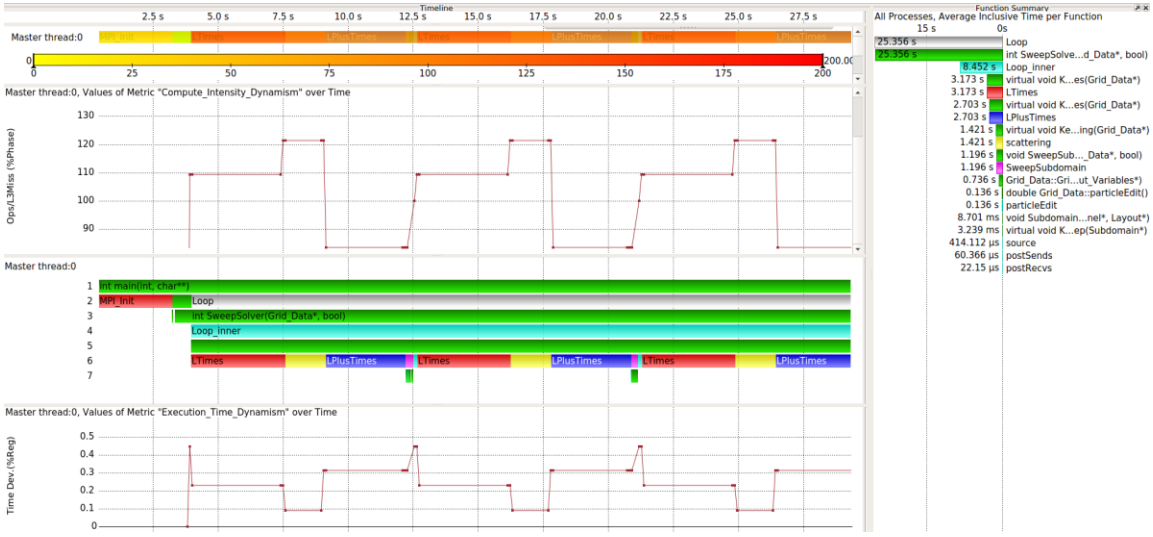


Figure 2: Vampir trace showing "Compute_Intensity_Dynamism" and "Execution_Time_Dynamism " metrics. The phase region is named "Loop" and granularity threshold is 500ms

To enable the user of the READEX Tool Suite to investigate the ATM generated during DTA, we designed different ways to visualise the results of the READEX methodology. One tool is based on the JavaScript library D3.js. It enables users to compare scenarios in the ATM with respect to their similarity and weight. We have updated the visualization tool for new versions of the tuning model format. We added minor improvements to increase its usability. These cover the layout of the force layout used for visualizing similarities among the scenarios and the textual output when inspecting individual scenarios or RTSs. We also support visualization for the application of the tuning model. Here, we examine the actual tuning process. To do so, we added Score-P metric support to the RRL. This enables users to visualize each of the configuration switches, which apply during the application tuning process. Users can also verify the application by reading hardware performance monitoring counters.

We described the components in more detail in Deliverable D2.2.

1.2.3. WP3: Run-time detection and switching

The focus of this work package is on the implementation of the READEX Runtime Library and its use at runtime. To achieve this objective, we implemented a scalable RRL architecture. It makes use of the ATM, which is provided by the DTA (WP2) and enhances it by a Runtime Calibration mechanism. With this approach, the READEX Tool Suite enables dynamic runtime adaptation to changing application characteristics in order to increase the application's energy efficiency. In addition, PTF uses the RRL for controlling parameter settings during DTA.

Main objectives of the work package for the second reporting period were:

- ✓ A final definition and creation of a scalable runtime library
- ✓ A final definition and creation of efficient scenario detection and switching mechanisms
- ✓ A final definition of an efficient runtime scenario calibration mechanism

Major results of the work package during the second reporting period are:

- The finalisation of a scalable runtime library and its testing and validation
- A runtime scenario detection that extends the traditional DTA/RAT difference
- A runtime scenario detection mechanism that include all identifier types determined in the project and signals need for calibration
- A runtime calibration mechanism that is able to calibrate the tuning model and tune applications during runtime
- The provision of interfaces as part of a commonly used performance monitoring infrastructure
- A scenario switching mechanism that enables configuration of all tuning parameters determined in the project

*Task 3.1: Scalable runtime library architecture (**TUD**, TUM, M1-M24)*

This task focused on the development of a scalable architecture for the READEX Runtime Library (RRL), which performs the detection of the system scenarios at runtime, applies the ATM generated at design time by adjusting tuning parameters, and enables an efficient run-time scenario calibration. The design goal of the RRL was to create an easy to modify and easy to extend library that provides low-overhead switching capabilities in order to keep the impact of the library on application performance as low as possible. The library was implemented in C++14. For the communication between PTF and RRL as well as the ATM, we used the JSON standard.

The READEX project implemented an extension to Score-P called *Substrate Plugin Interface*. This interface provides a plugin infrastructure for consumers of events generated from the instrumentation. We implemented the RRL as a substrate plugin to avoid direct integration of the tuning functionality into the Score-P code base. This is necessary to minimize the impact on the regular Score-P development process and to reduce maintenance efforts for both the Score-P and the READEX developers by using a stable and abstract interface between the two projects. The *Substrate Plugin Interface* is now part of the Score-P release 4.0.

The RRL performs the detection of runtime situations, the classification of runtime situations into scenarios, the selection of configurations and the application of these configurations through parameter control plugins. Moreover, it includes the interaction between the tuning model, the Score-P substrate plugin interface (used for the detection of runtime situations), the Online Access interface (used at design-time to control parameter settings), and the calibration mechanism. It also includes the support for applying ATP tuning both at design-time and at runtime.

*Task 3.2 Runtime scenario detection (**NTNU**, TUM, M1-M30)*

The work of this task in the second half of the project is a continuation of the runtime scenario detection work described in in Deliverable D4.2 and D7.2. We extended the RRL to handle the new version of the ATM, which includes input identifiers, user parameters, and inter-phase tuning. If the RRL cannot identify a current scenario, it is now able to detect the need for scenario calibration, as well as include new scenarios for detection after calibration. We described our work in more detail in Deliverables D3.2 and D4.4.

*Task 3.3: Runtime calibration mechanism (**TUD**, NTNU, M13-M30)*

For runtime calibration, we provide two different Machine Learning based approaches. The first approach is a Q-Learning based mechanism. The algorithm starts at the maximal core and uncore frequency. With a probability of ϵ, the algorithm selects a different setting from the next direct neighbours. Then it measures the energy at this point. Based on that, it calculates a so-called Q-Value. Afterwards, the algorithm chooses the next optimal status according to the Q-Value and starts from the beginning. Figure 3 shows how the mechanism searches for an energy efficient optimum (core frequency = 2.3 GHz, uncore frequency = 2.2 GHz) starting from a selected initial setting (core frequency = 1.9 GHz, uncore frequency = 2.2 GHz).

In a second approach, we used performance counters to predict the optimal setting using Neural Networks. First, the approach identifies the most relevant performance counters. In a second step, it trains a shallow neural network in order to predict a good frequency. This model is then loaded during runtime. Once the RRL encounters an unseen RTS, it measures the relevant performance counters and uses the neural network to predict a good setting. In opposite to the Q-Learning approach, this approach does require a pre-training, which has to be performed once per system. However, once this is done, the network needs just to be evaluated, which requires less tuning and runtime overhead compared to the Q-Learning approach. We described the mechanisms in Deliverable D3.2.

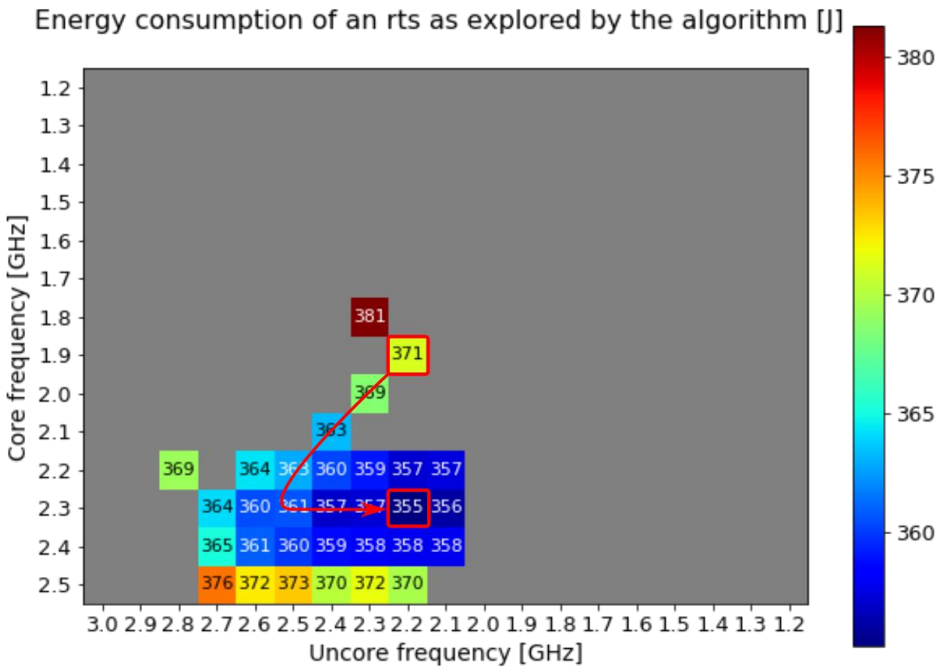


Figure 3: Heatmap of energy consumption of a specific region for different core and uncore frequencies as explored by a Q-Learning algorithm applied at runtime. The algorithm starts at 1.9/2.2 GHz and finds a more suitable setting (2.3/2.2 GHz).

*Task 3.4 Efficient switching decision making (**NTNU**, TUD, TUM, M1-M30)*

The work of this task in the second half of the project was a continuation of the switching decision work described in Deliverables D4.2 and D7.2. The switching decision making takes place after the scenario identification is finished. If the configuration of the upcoming scenario differs from that of the current scenario, a switch takes place where the relevant platform parameters are changed.

We extended the mechanism to take into account additional parameters, as well as the possibility of adding scenario configuration settings after runtime calibration. Furthermore, we implemented a mechanism where the user can decide whether the previous configuration should be restored after the exit of a region or if the new configuration shall be kept. We described our work in Deliverables D3.2 and D4.4.

1.2.4. WP4: READEX Tool Suite Development

The main objective of WP4 was to integrate the developed techniques of WP1, WP2, and WP3 into the overall READEX Tool Suite. As part of this objective, we defined the interfaces between the design and run-time components and overall tuning steps, based on the terminology specified in Task 2.1. We also defined and developed a generic specification for providing domain knowledge about application dynamism. As part of our integration efforts, we exploited the Pathway tool to integrate the overall READEX tuning process into tuning workflows. In this work package, we also coordinated the development of the READEX Tool Suite prototypes and the final version of the overall Tool Suite, including the final software release.

NUIG lead this work package and received contributions from TUM, TUD, NTNU and IT4I. WP4 has depended on input from work packages WP1-WP3, with the output of WP4 feeding into WP5 and WP6.

Main objectives of the work packages for the second reporting period were:

- ✓ The finalisation of the objectives named above, excluding the definition of interfaces
- ✓ The integration of further prototypes
- ✓ The delivery of a final software

Major results of the work package during the second reporting period are:

- The finalisation of an interface to specify domain-level knowledge about dynamism in an application, called Domain Knowledge Specification Interface.
- Extensions to the Pathway tool with additional components and features to support applying and engineering the READEX workflow on applications.
- Regular versions of the READEX Tool Suite prepared through continuous integration and regression testing using the READEX Test Suite, in the form of beta and release candidate versions, supported by installation and user guides.
- The provision of beta-releases and the final software release candidate on the project website

*Task 4.1: Design-time/runtime interface (**TUM**, TUD, NTNU, NUIG, M01-M12)*

This task finished in the first reporting period. We documented the progress in Deliverables D4.1, D4.2, and D7.2.

*Task 4.2: Domain-level dynamics specification (**TUM**, TUD, NTNU, IT4I, NUIG, M7-M24)*

During the second reporting period, we developed the READEX DKSI, which provides means to express the expert's domain knowledge related to the application structure, the application characteristics, and specific application-level tuning parameters. The specification of the application structure and the application characteristics leverage features available in Score-P, such as user regions and user parameters. An additional specification file, which provides input characteristics to the DTA, supports different input identifiers capturing characteristics of a given input. The ATP library provides an API, which publishes the specified application-level tuning parameters. We described the DKSI specification in more detail in Deliverable D4.5.

*Task 4.3: Performance engineering workflows (**NUIG**, TUM, M13 - M30)*

The objective of this task was to use and extend the existing performance engineering workflow tool called Pathway in order to integrate the steps of DTA and RAT of the READEX Tool Suite.

Following the initial extensions listed in the first periodic report D7.2, we added more features and components to Pathway for the extensions in the beta prototype by M30.

The extensions for the beta prototype include:

- Using the ATP library in the workflow during DTA and RAT,
- A structured report of the Tool Suite outputs (tabulated tuning potential reported by `readex-dyn-detect` and the tuning model generated by PTF),
- A feature to merge multiple tuning models generated by PTF,
- A feature to allow PTF to use a dedicated node during DTA,
- Generic tasks that allow the end-user to provide customised application build, and
- Execution scripts and specification of choice and value ranges for different hardware and system software tuning parameters.

At the end of the task in M30, the extended Pathway tool was tested using the benchmark applications and is now available via the Pathway repository at https://periscope.in.tum.de/git/pathway.git as well as on the READEX github page.

*Task 4.4: Framework integration (**NUIG**, TUD, NTNU, IT4I, TUM, M7-M36)*

The objective of this task was to provide mechanisms to integrate the different components developed for the READEX Tool Suite by ensuring software quality, regression testing, bug tracking, and reporting to the owners for debugging and fixing.

For this, we used the Jenkins continuous integration system at TUM to set up projects that automatically check the repositories of the READEX components for updates. The system triggers builds of the relevant tools, performs regression tests using the benchmark applications in the READEX Test Suite, and reports any errors in these steps to the tool owners along with the error summary.

The application owners at TUD, TUM, IT4I, GNS and NUIG prepared scripts for regression tests for their applications in the READEX Test Suite, which we have described in Deliverable D5.3. The continuous integration process uses these scripts to automatically run the tests and report results/errors to the developers and evaluators.

For bug reporting, bug tracking, discussions of debugging and features, we used a GitLab setup to categorise and facilitate issues related to each component (Score-P, PTF, RRL, ATM, PCPs, ATP), continuous integration setup, documentation and scripts to automate applying the Tool Suite on HPC applications.

This continuous integration system also automatically creates modules for the different tools and the READEX Tool Suite as a whole on the cluster at TUD. Additionally, we have periodically prepared regular versions of the READEX Tool Suite (alpha, beta and updated beta versions) along with corresponding installation and user guides. We made the beta versions available to both consortium members and external users via the READEX website (www.readex.eu) starting in March 2018. The installation and user guides contain information to support deploying the Tool Suite on any external HPC system, while also providing concrete examples and scripts based on the cluster at TU Dresden.

This task culminated in the preparation of the final release of the READEX Tool Suite along with the installation and user guides, and a Deliverable D4.4 that briefly summarises the design features of the Tool Suite components in M36.

1.2.5. WP5: Applications and validation

This work package mainly focused on the evaluation of the dynamism of real world applications. However, in order to be able to test complex applications efficiently, we needed to develop support routines for manual instrumentation and manual energy efficiency evaluation. We evaluated saving potentials of applications as part of Task 5.1 and investigated application behaviour in both, Task 5.1 and Task 5.2. Furthermore, we evaluated the READEX Tool Suite in Task 5.3.

Main objectives of the work packages for the second reporting period were:

- ✓ A finalisation of the application set for evaluation
- ✓ The evaluation of the READEX Tool Suite

The major results of the work package during the second reporting period are:

- The definition of a set of applications used for tests and evaluation

- The finalisation of the manual tuning analysis
- An evaluation of beta prototypes of the READEX Tool Suite and provision of hints for the Tool Suite developers
- The general evaluation of the READEX Tool Suite in comparison to other approaches

*Task 5.1: Evaluating Dynamism in HPC applications (**NUIG**, NTNU, IT4I, GNS, M01-M18)*

This task ended in the previous reporting period and was described in detail in Deliverables D5.1 and D7.2.

*Task 5.2: Manually exploiting application dynamism (**IT4I**, GNS, M07 - M30)*

The work of this task in the second half of the project was to extend the portfolio of the applications that forms the final READEX Test Suite. We omitted some of the applications that we evaluated earlier for a lack of dynamism or unsuitable programming techniques. This includes the following applications: Blender graphic Tool Suite, SMURFF, Betweenness, Probabilistic Time-Dependent Routing kernel (applications from ANTAREX project). We have presented the final list of selected application in Deliverable D5.2.

For the final set of applications, we developed scripts for manual tuning and integrated these scripts into READEX Test Suite, where these provides results for manual static and manual dynamic tuning as presented in D5.3. For the manual effort (Objective O4), we also recorded our effort in order to compare it with fully automatic tuning provided by READEX Tool Suite.

*Task T5.3: Evaluation of the READEX Tool Suite (**IT4I**, TUD, NUIG, GNS, M19 – M36)*

This task was active for the entire second half of the project duration. In this task, all partners validated the applications specified in Deliverable D5.2 to enable the application of the READEX Tool Suite. A significant amount of effort was devoted to debugging of the READEX Tool Suite and pointing out limitations that could only be identified when we used the READEX Tool Suite with complex applications such as ESPRESO or OpenFOAM. We also added some complex scientific applications to the READEX Test Suite. We ported the Test Suite to multiple batch systems (SLURM and PBS) and machines. We used the Top500 listed clusters Taurus (TUD) and Salomon (IT4I) for evaluation. We presented the results of this task in Deliverable D5.3. We also presented ATP tuning for complex applications using both a dynamic but in most cases a static approach using ACPs.

1.2.6. WP6: Dissemination and communication

The objective of this work package was to implement dissemination and communication strategies to inform project stakeholders, the general public, the technology community, and interested parties about ongoing developments and project results.

The objectives of this work package stretched from M01 to M36 and include:

- ✓ Dissemination, including papers at conferences, journals, and trade journals
- ✓ Organization of workshops
- ✓ Exploitation of results
- ✓ Communication through website, mailing list, social media, and press releases
- ✓ Communication of project results

Major results of the work package are:

- The utilization of public communication channels to reach out to public, scientific, and industry communities
- The dissemination of scientific results in presentations, and publications
- The (co-)organization of four workshops
- The maintenance of an external advisory board
- The collaboration with other projects and research groups

*Task 6.1: Project website and awareness raising (**TUD**, M01-M36)*

The project partners started disseminating project ideas and results early on in the project. These activities included the creation of dissemination material such as flyers and presentation templates, electronic dissemination through a website (https://www.readex.eu), a twitter account (@readex_eu), and a Research Gate project (https://www.researchgate.net/project/READEX). During the second reporting period, we organized four workshops to provide a platform for an exchange of ideas on the topic of energy efficiency auto-tuning.

Through these activities, the READEX project reached a broad audience. The READEX website provides both an overview and more detailed information on the project to policy makers, the general public, and interested users. We publish updates on the project through the website, twitter feed, and Research Gate. More information is available in Deliverable D6.6.

*Task 6.2: Dissemination of project results (**TUD**, NTNU, IT4I, NUIG, INTEL, TUM, M01-M36)*

All project partners actively participated in the dissemination of project results. This includes scientific publications, presentations at workshops and conferences, and non-scientific (popularised) publications. We presented more details in Deliverable D6.6.

*Task 6.3: Exploitation of project results (**INTEL**, IT4I, GNS, M01-M36)*

During the second phase of the project, the project partners IT4I and GNS tested and applied the READEX Tool Suite with their applications. The project partner Intel used the parameter analyses to determine energy efficiency knobs in preparation to a collaboration with the Open Source tool GEOPM. Furthermore, Intel used READEX for the optimization of the application Alya in collaboration with EoCoE and BSC. We will also use READEX technology in future projects (e.g., the EU project *PRACE-5IP,* and the German BMBF IKT-Forschungsvorhaben *ProVerB*[4]). Furthermore, we will exploit the READEX Tool Suite with partners associated with the European Exascale Labs, the IPCC centres, and IXPUG community. We present more details in Deliverable D6.6 and Section 2 of this document.

*Task 6.4 Synchronisation with External Advisory Board (**TUD**, INTEL, TUM, M01 - M36)*

In the second reporting period, we held two EAB meetings in project months 23 and 29, respectively. We described the status of the software to the EAB members, laid out our development plans, and discussed the project and the software. The general feedback of the external advisory board during these meetings was positive. Feedback from Bronis di Suspinski (LLNL) regarded the merging of regions, which is not possible with the READEX approach. We have published a way to track region order, which is necessary to achieve this [1]. However, such an approach would significantly increase measurement overhead. We therefore decided not to include it. Furthermore, the External Advisory Board criticised the manual marking of a phase in READEX. We therefore developed a patch that enables users to use compiler instrumentation alternatively. However, since none of the codes we analysed was structured in a way that

[4] https://www.softwaresysteme.pt-dlr.de/media/content/Projektblatt_ProVerB.pdf

such an approach was usable, we lowered the priority for such a fix and left its final integration as future work. The next concern regarded the overhead and granularity of energy measurements in the first prototype. To overcome this hurdle, we integrated lower-overhead energy measurement interfaces. This enabled us to re-define the granularity of regions from hundreds to tens of milliseconds. Even though the interest of IMEC is currently not about energy efficiency optimizations, the EAB member Francky Catthoor was and is interested in collaboration regarding energy efficiency analysis. We held two telephone conferences for discussion and provided them with access to the project test cluster and software that IMEC actively uses at their site.

We furthermore sent out a survey to the EAB and other sites, where we collected information about potential software that we could target, used processor architectures, and available software interfaces. We used the responses to extend our Tool Suite to be applicable on additional platforms.

*Task 6.5 Contribution to external working groups (**TUD**, NTNU, INTEL, M01 - M36)*

NTNU actively participated in the "Special interest group on Scenario Driven Design for Embedded Systems", as discussed in the interim report (PM9), and continues to do so. Currently the group is finalizing a book on system scenario based design, where the results from READEX is included. TUD contributed to the Score-P project for the 4.0 release. The project partners provided feedback and improvement suggestions to the HDEEM project, based on the experience on the Taurus test system. Research cooperation with the DFG SFB HAEC[5] resulted in multiple publications. Furthermore, it resulted in the development of a new performance and energy efficiency monitoring tool and a proposal for a patch for the Linux kernel, whose final form is included in Linux kernel 4.17 and can tens of millions € annually for European data centres. We described more details in Deliverable D6.6. TUD collaborated with the EEHPCWG in organizing a workshop at ISC'18 and sparking the interest for READEX at the SC'17 conference.

1.2.7. WP7: Coordination and Management

The objective of WP7 is to ensure an effective management of the project, including day-to-day administration, project co-ordination and monitoring of the work in progress.

The objectives of this work packages stretch from M01 to M36:

- ✓ Ensure highest quality of research activities, streamline the research and development activities carried out by all partners;
- ✓ Ensure the project's implementation within the targets of time, budget and quality and the achievement of objectives;
- ✓ Provide a management structure and platform for efficient communication and decision-making between the partners while establishing and maintaining a high level of team spirit;
- ✓ Co-ordinate the overall project
- ✓ Provide the interface between the European Commission and other stakeholders for communication ensuring visibility of the project
- ✓ Provide the interface for the External Advisory Board in order to involve it in major decisions within the project and ensure efficient communication of relevant issues and information

[5] http://gepris.dfg.de/gepris/projekt/164481002

Major results of the work package are:

- The coordination of the project, for example via face-to-face meetings, online meetings, and collaboration tools, to ensure collaboration between project partners and dissemination quality;
- The establishment and hosting of internal collaboration tools;
- The communication and synchronization with the External Advisory Boards;
- Regular reporting activities to the European Union.

We carried out the activities in WP7 Project Management according to plan. TUD coordinated the activities within this work package and carried out all tasks in cooperation and with support from all project partners.

*Task 7.1 Project coordination (**TUD**, M01 - M36)*

The focus of Task 7.1 was the supervision of the research progress considering the scientific objectives targeted by the project. This included several aspects such as

- Ensuring accomplishment of the project's objectives;
- Quality management and monitoring compliance by the consortium participants with their obligations, verifiable assessment and reviewing of the project against the deliverables and milestones;
- Ensuring knowledge transfer among work packages;
- Management of risks and conflicts, internal reporting, internal dissemination of information;
- Preparing and chairing of meetings, preparing minutes of meetings and monitoring implementation of decisions taken at meetings;
- Collecting, reviewing, and submitting technical reports and other deliverables (including financial statements and related certification) to the European Commission

Project management structure

The READEX project coordination ensured a smooth implementation and execution of the project. We used a highly efficient management structure. This management structure included three control levels:

- At the strategic level, the General Assembly (GA), in which each partner was represented, decided the overall strategic orientation of the project, agreed on plans, monitored milestones and approved results.
- At technical and operational level, the Board of Work Package Leaders (WPL) steered the technical activities of the project and ensured the technical quality of the deliverables. The External Advisory Board (EAB) acted as gateways between READEX and its stakeholders.
- During day-to-day operation, the Coordination Office at TUD conducted the daily affairs at technical and scientific level.

Meetings

Since the beginning of the project, we exchanged information internally on a regular basis. To do so, we used online collaboration tools and audio conferencing tools whenever possible. The READEX project meetings during the second project period were scheduled and held as follows:

- Monthly GA phone meetings,

- Bi-weekly technical online meetings,
- Bi-annual face-to-face plenary workshops (at least two days of technical discussions)
    - Review preparation meeting: May 11, 2017 @Munich/DE
    - Rehearsal and review meeting: May 22-23, 2017 @Brussels/BE
    - Plenary meeting 4: September 12-13, 2017 @NTNU in Trondheim/NO
    - Plenary meeting 5: March 5-7, 2018 @NUIG in Dublin/IE
    - Review preparation meeting: August 28, 2018 @Dusseldorf/DE
- External Advisory Board meetings
    - 2nd EAB meeting: June 19, 2017, @Frankfurt/DE
    - 3rd EAB meeting: January 10, 2018, online

We refer to the Sharepoint for more information on the agenda and the meeting minutes of the past face-to-face meetings.

*Task 7.2 Legal, financial and administrative coordination (**TUD**, M01 - M36)*

This task supervised all legal, financial and administrative issues including

- monitoring of work flows and scheduling of work, communication between partners and to EC;
- monitoring and collecting deliverables according to DoA, monitoring of milestones, external reporting to EC;
- collecting, reviewing, and submitting technical and financial reports and other deliverables (including financial statements and related certification) to EC, budgeting and distribution of money to the partners.

Progress follow-up internal reporting (technical and financial)

On a quarterly basis, all partners and WP Leaders produced a short internal technical status report describing the progress for the past period, while considering any deviations from the plan and proposing corrective measures. This included an overview about resources (spent person months and costs), which eased timely corrective measures in case of substantial deviations. These financial reports allowed the Administrative Manager at TUD to monitor actual costs and spent effort, but also detect and prevent possible financial errors. All reports were prepared based on structured templates and uploaded in Sharepoint for all partners and the Project Coordinator.

*Task 7.3 Internal website and communication tools (**TUD**, M01 - M36)*

At the beginning of the READEX project several tools we set-up tools to coordinate and manage the development and research activities. All partners used these tools for the internal communication.

The main mechanism for project communication was through email and by means of two mailing lists:

- one for development and active research discussions (mailto:readex-dev@fusionforge.zih.tu-dresden.de), and
- one for general project organisation and management (mailto:readex-project@fusionforge.zih.tu-dresden.de).

For managing documents, deadlines, and tasks, we used a Sharepoint project, which is accessible via https://sharepoint.tu-dresden.de/projects/readex/.

For the distributed development and managing of the source code of the project, we used two instances of the version management system GIT, located at TU Munich and TU Chemnitz, respectively. The first repository mainly holds documents like deliverables (http://periscope.in.tum.de/git), and publications. For development purposes, we mostly relied on the second repository. In addition to the mere version control, the TU Chemnitz gitlab server (https://gitlab.hrz.tu-chemnitz.de/READEX) provided us with issue trackers, continuous integration hooks and other features for distributed software development.

Last but not least, we used an Adobe Connect video room for the regular monthly status calls, bi-weekly technical calls, and any remote extraordinary meeting *(*https://webconf.vc.dfn.de/readex*).*

1.3. Impact

Topic item 1: Contribution to the realisation of the ETP4HPC Strategic Research Agenda

There is no change in the expected impact, described in Section 2.1 of the DoA. We defined a programming paradigm that enables users to provide application domain knowledge. We introduced metrics to find application dynamism and implemented the Tool Suite.

Topic item 2: Covering important segments of the broader and/or emerging HPC markets

There is no change in the expected impact, described in Section 2.1 of the DoA. We demonstrated that the READEX Tool Suite can be used to enhance energy efficiency at different hardware and software stacks in Deliverable D5.3. We worked together with the EAB and other users across Europe to ensure a stable deployment process and a precise documentation.

Topic item 3: Impacts on standard bodies and other relevant international research programmes and frameworks

There is no change in the expected impact, described in Section 2.1 of the DoA. The project members contributed to the external working groups as described in Deliverable D6.6.

Topic item 4: European excellence in mathematics and algorithms for extreme parallelism and extreme data applications to boost research and innovation in scientific areas such as physics, chemistry, biology, life sciences, materials, climate, geosciences, etc.

The impact described in Section 2.1 of the DoA is still valid. The project partners showed the applicability of the Tool Suite on a variety of applications.

2. Exploitation and dissemination of results

The project partners published 34 articles and papers, including accepted ones. The partners organized and co-organized five workshops and presented READEX posters at 20 conferences and workshops. We also communicated results through the project web page, a Twitter account, and a ResearchGate project. As suggested in the interim review, we used the final phase of the project for exploitation and collaboration, where we collaborated with external partners on tuning their codes and supporting installation at European HPC sites. We also collaborated with other European and national research projects to advertise our tools and combine expertise for related questions. We implemented numerous software components, which are available with non-restrictive Open Source licenses at the READEX github repository[6]. We show an overview of dissemination activities in Figure 4: General Overview on Dissemination Activity. Here each dot represents one or multiple (up to five) activities of the three dissemination domains during a project month. We present more information in Deliverable D6.6 and Part A of the final report.

The industry partner GNS plans to use READEX technology for the optimization of its existing software products. Furthermore, GNS will use it in development of a new software package related to the ultra long-term simulation of behaviour of concrete under mechanical load (BMBF funded project ProVerB). In both cases, the company believes that the use of the READEX technology will lead to improvements of these software systems that lead to a better performance and reduced energy requirements that will, in turn, provide a better position on the market. In addition to these actions, GNS will also develop a marketing strategy for introducing energy-related consulting services for commercial HPC centre operators. In the long run, it is hoped that this will lead to an additional market that GNS can serve, thus diversifying its

[6] https://github.com/readex-eu

product and service portfolio and reducing its dependence on the currently served markets. We will use the READEX Tool Suite in future projects and present it during user trainings. Furthermore, the interfaces we created will spark research activities in other areas.

The industry partner Intel opened a discussion with the GEOPM[7] team to further use READEX technology. Among the novel ideas that have shown to be efficient, is the automated identification of interesting regions: the automated detection of the runtime situations. The promising idea is to take into account the various peripherals that play a role in the energy savings, like the network fabric and memory management. Intel is committed to use the findings of the academic partners in the READEX project for the open source project GEOPM.

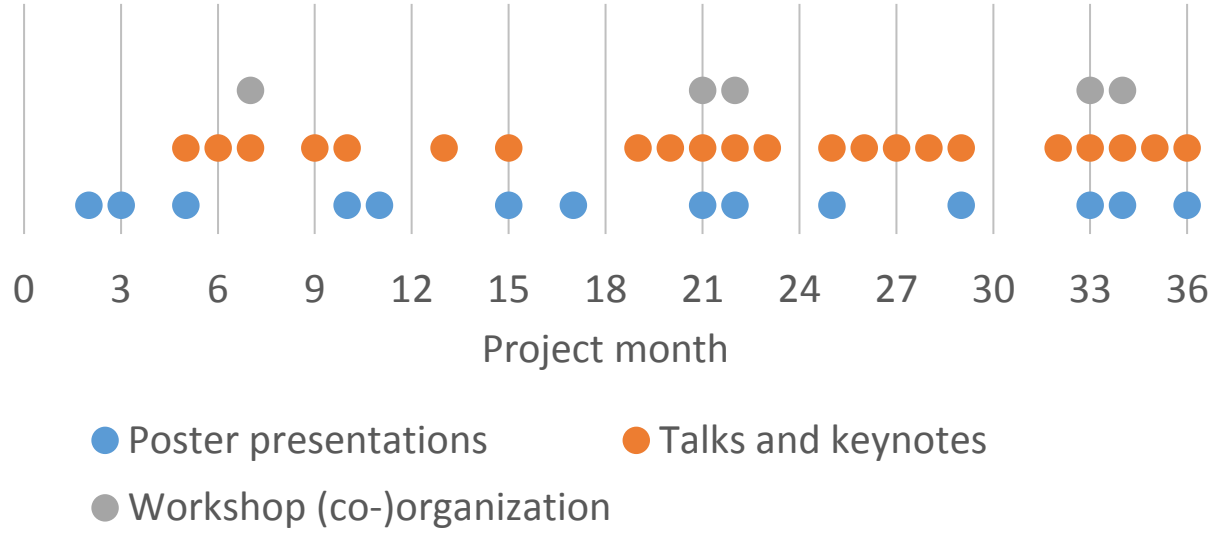


Figure 4: General Overview on Dissemination Activity

3. Update of the data management plan

As described in the proposal, we published all of our software components using the most liberal open source licenses at https://github.com/readex-eu. We used the BSD license whenever possible. Furthermore, we also provide the test applications and test scripts at this location to enable other users and researchers to test and validate the READEX Tool Suite.

[7] https://geopm.github.io/

4. Follow-up of recommendations and comments from previous review(s)

In addition to continuing to implement the work plan, which remains a good and relevant guide in this project, the key consideration of the consortium should be to maximise the uptake of the technology, which they are developing.

The idea of site visits to potential user institutions, including those represented in the external advisory board, is a good one. However, there are also other issues, which should be addressed, in particular to convince application developers that the READEX technology is worth investing in.

Recommendation 1:
The website should become focussed on the product rather than the project, with an emphasis on high quality documentation and working, responsive support routes.

How it was addressed by READEX:
We re-structured the website with a focus on the software, added regular updated versions of our software, added a user support mailing list and opened a github page.

Recommendation 2:
The software needs to be easily and robustly deployable to workstations as well as supercomputers so that application developers can integrate it into their test and development workflows.

How it was addressed by READEX:
We clearly defined the dependencies for our software and tested it with common freely available software versions. For the workflow integration, we provide Pathway with READEX specific extensions.

Recommendation 3:
In selecting the remaining test applications, emphasis should be placed on applications with significant tuneable application parameters. These have greater potential to deliver significant energy performance increases than the system parameters, which have been the primary focus thus far. Demonstrating success in this area may induce application developers to invest time in employing READEX.

How it was addressed by READEX:
To support more application tuning parameters, we put additional effort in implementing Application Configuration Parameters. These can be used without instrumenting the existing source code and can be optimized with the READEX Tool Suite.

Recommendation 4:
The project is not participating in the open data pilot, which is perfectly fine under current rules. However good scientific practice demands that a best effort approach to traceability and reproducibility of results be taken. In particular, this should mean that the code employed, the raw result data, and the processing scripts used to produce the publications of the project should be properly recorded and archived, using a service such as Zenodo or Figshare.

How it was addressed by READEX:

The READEX Test Suite, i.e. the set of applications used for the evaluation of the READEX Tool Suite, is stored in a public repository at https://github.com/readex-eu/readex-apps. We instrumented the applications manually for the evaluation of the manual tuning. In addition, we prepared scripts for the READEX Tool Suite, which use automatic instrumentation. This includes compilation and run scripts for SAF, RDD, PTF and RRL on several HPC clusters (Taurus Haswell partition and Taurus Broadwell partition at TUD and Salomon machine at IT4I). Therefore, researchers can reproduce the results reported in D5.3 by cloning the repository and using the prepared scripts. Moreover, the scripts can be easily adapted for new applications and for new READEX users. The Test Suite's Zenodo link is https://zenodo.org/record/1451438.

5. Deviations from Annex 1 and Annex 2 (if applicable)

5.1. Tasks

There have been no deviations at task level. All deliverables have been delivered in time and the milestones achieved according to the plan.

5.2. Use of resources

5.2.1. Effort spent in the period

Figure 5 summarises the effort spent per partner spent during the second reporting period (March 2017 - August 2018). In contrast, the effort spent per partner and per work package from the beginning of the project (September 2015) until its end (August 2018) is shown in figures 6 and 7.
Figure 8 illustrates the planned total effort vs. reported effort in M1-M36 per work package, whereas Figure 9 focuses on the distribution at partner level.

	WP1		WP2		WP3		WP4		WP5		WP6		WP7		TOTAL	
	Plan M19-36	Spent M19-36	Plan M19-36	Spent M19-36	Plan M19-36	Spent M19-36	Plan M19-36	Spent M19-36	Plan M19-36	Spent M19-36	Plan M19-36	Spent M19-36	Plan M19-36	Spent M19-36	Plan M19-36	Spent M19-36
TUD	4,00	5,14	3,00	1,82	8,30	13,60	5,40	3,27	9,00	8,68	5,50	5,63	9,00	6,82	44,20	44,96
NTNU	0,00	0,00	7,60	9,76	10,10	8,99	4,80	5,85	0,00	0,88	2,50	7,10	0,00	0,00	25,00	32,58
IT4I	2,00	6,59	2,40	4,41	0,00	0,00	2,80	4,24	21,50	32,91	3,00	4,75	0,00	0,00	31,70	52,90
NUIG	4,70	6,53	0,00	0,00	0,00	0,00	13,80	19,07	6,00	7,15	2,00	2,42	0,00	0,00	26,50	35,17
INTEL	7,00	18,47	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	4,50	8,86	0,00	0,00	11,50	27,33
TUM	0,00	0,00	15,80	17,50	4,30	5,50	5,50	7,50	0,00	0,00	2,50	3,50	0,00	0,00	28,10	34,00
GNS	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	12,00	15,80	1,00	2,10	0,00	0,00	13,00	17,90
Total	17,70	36,73	28,80	33,49	22,70	28,09	32,30	39,93	48,50	65,42	21,00	34,36	9,00	6,82	180,00	244,84

Figure 5: Effort spent during reporting period (M19-M36).

	WP1		WP2		WP3		WP4		WP5		WP6		WP7		TOTAL	
	Plan M1-36	Spent M1-36	Plan M1-36	Spent M1-36	Plan M1-36	Spent M1-36	Plan M1-36	Spent M1-36	Plan M1-36	Spent M1-36	Plan M1-36	Spent M1-36	Plan M1-36	Spent M1-36	Plan M1-36	Spent M1-36
TUD	20,00	23,22	6,00	5,37	17,00	22,47	9,00	9,18	9,00	8,68	11,00	9,78	18,00	11,42	90,00	90,12
NTNU	0,00	0,00	21,00	18,16	24,00	19,99	8,00	10,65	2,00	2,58	5,00	8,10	0,00	0,00	60,00	59,48
IT4I	11,00	10,99	6,00	6,01	0,00	0,00	6,00	6,04	43,00	48,01	6,00	5,95	0,00	0,00	72,00	77,00
NUIG	7,00	7,53	6,00	4,00	0,00	0,00	29,00	30,07	12,00	10,15	4,00	2,72	0,00	0,00	58,00	54,47
INTEL	30,00	28,27	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	9,00	8,86	0,00	0,00	39,00	37,13
TUM	0,00	0,00	42,00	41,50	12,00	11,80	13,00	13,50	0,00	0,00	5,00	5,20	0,00	0,00	72,00	72,00
GNS	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	18,00	22,20	2,00	2,60	0,00	0,00	20,00	24,80
Total	68,00	70,01	81,00	75,04	53,00	54,26	65,00	69,44	84,00	91,62	42,00	43,21	18,00	11,42	411,00	415,00

Figure 6: Effort spent in total since project start (M1-M36).

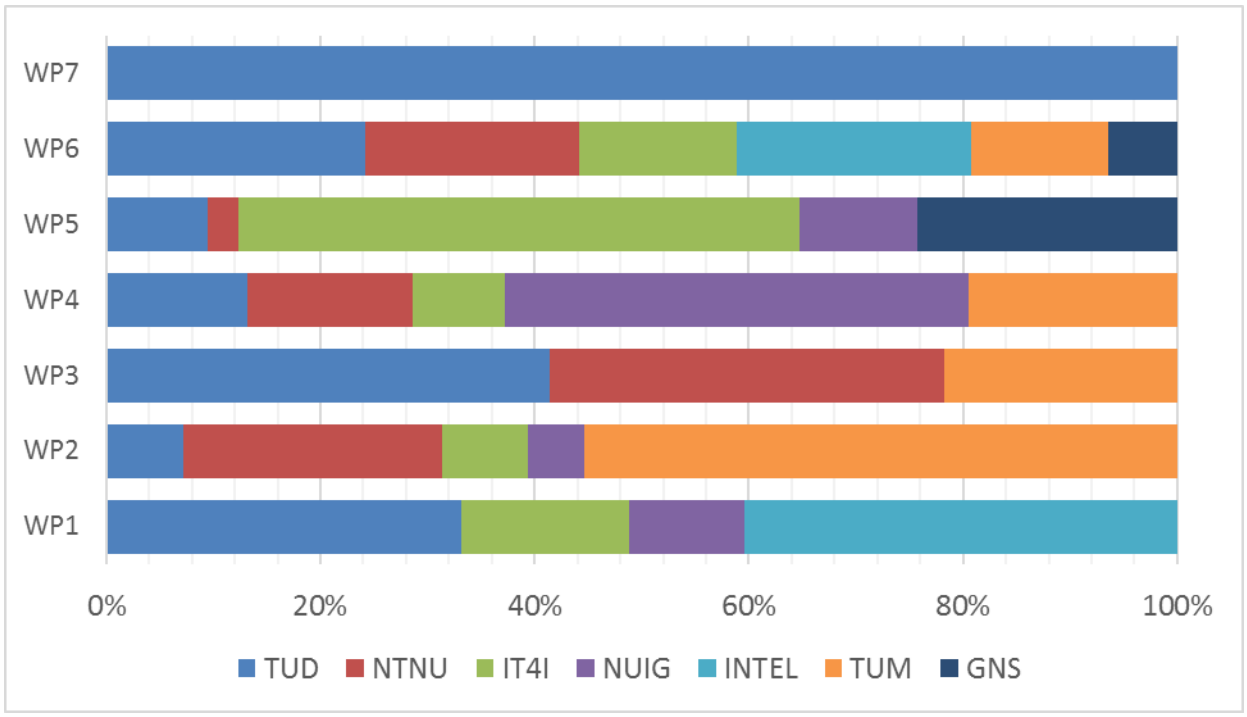
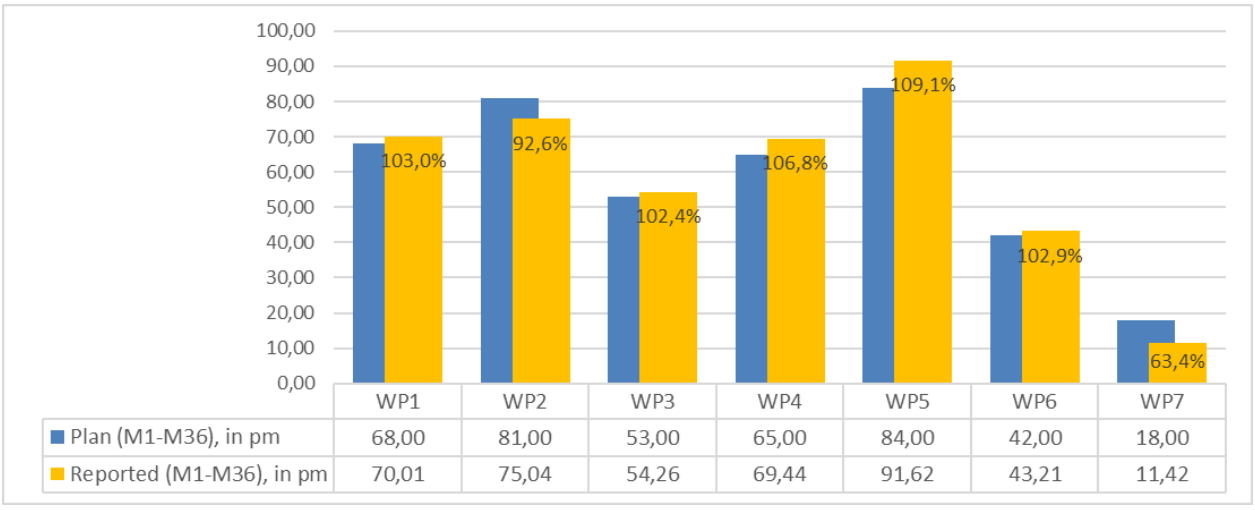
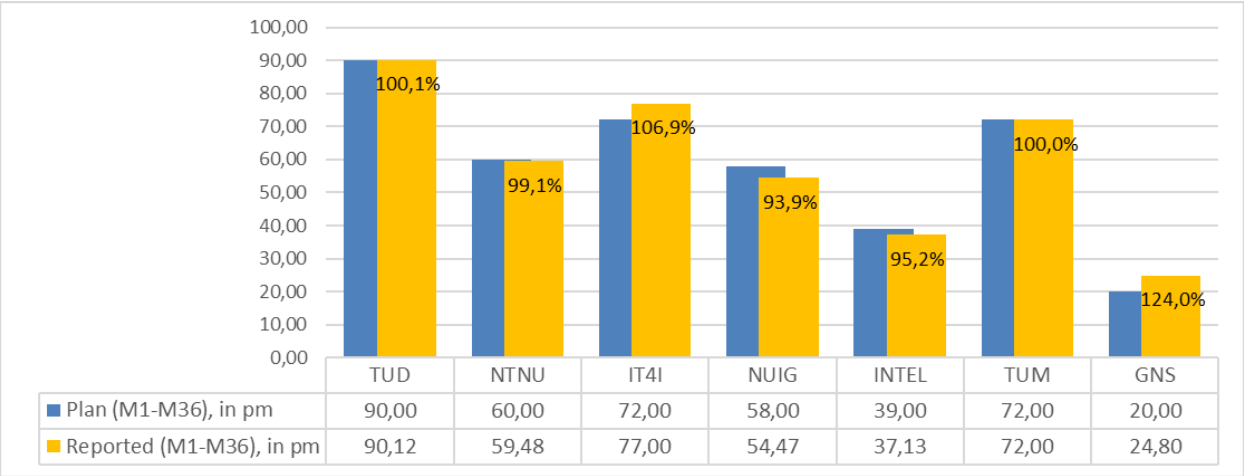


Figure 7: Partner involvement per WP since project start (in %).



	WP1	WP2	WP3	WP4	WP5	WP6	WP7
Plan (M1-M36), in pm	68,00	81,00	53,00	65,00	84,00	42,00	18,00
Reported (M1-M36), in pm	70,01	75,04	54,26	69,44	91,62	43,21	11,42

Figure 8: Spent effort per work package from project start until its end.



	TUD	NTNU	IT4I	NUIG	INTEL	TUM	GNS
Plan (M1-M36), in pm	90,00	60,00	72,00	58,00	39,00	72,00	20,00
Reported (M1-M36), in pm	90,12	59,48	77,00	54,47	37,13	72,00	24,80

Figure 9: Spent effort per partner from project start until its end.

5.2.2. Deviation from the plan

Over the period M1 – M36, partner GNS spent more effort than initially planned (see figure 9). The effort required becoming familiar with the concepts and techniques from the Embedded Systems world and the effort for the Application Parameter Tuning were higher than expected. Moreover, the addition of new staff members to the project during the run time caused a higher time effort because of the introductory training required by these employees. This effect is compensated by the fact that the salaries of the employees involved in the project were lower than expected at the time of writing the proposal. Therefore, there was no noteworthy overspending for GNS from the financial point of view.

At WP level, there are slight deviations of planned versus spent effort in WP5 and WP7 (see figure 8).

The overspending in WP5 was related to evaluation of additional applications (NTNU, IT4I) and evaluation of ACP (Application Configuration Parameters) (IT4I, GNS) which was introduced during the project runtime. In case of NTNU the overspending was only 0.6 person-months. In case of IT4I and GNS more person-months have been used, but due to lower average salaries than expected there was no overspending from a budgetary point of view.

Finally, the underspending in WP7 resulted from the fact that the coordination effort from TUD could not be charged completely to the project as the source of funds for the personnel cost is not READEX itself.

5.2.3 Adjustments of previous reporting period

Partners TUD, NUIG, TUM, and INTEL submitted adjustment of the previous reporting period due to the following reasons:

- TUD: Adjustment of personnel costs and of annual special payment taking into account the Horizon 2020 calculation scheme and required double ceiling.
- TUM: The adjustment became necessary as the costs of two travels were wrongly charged to the project in the first period and some travels during the end of the first period that were not charged

to the first reporting period. Additionally, the total of personnel costs had to be corrected due to Horizon 2020 calculation scheme.

- NUIG: Correction of personnel costs due to Horizon 2020 calculation scheme.
- INTEL: The adjustment for the first reporting period comes from the small error in calculations which was found during the check while preparing second financial statement. Not all the incurred eligible personnel costs was taken into calculation in total.

5.2.3. Unforeseen subcontracting

not applicable

5.2.4. Unforeseen use of in kind contribution from third party against payment or free of charges

not applicable

6. References

[1] R. T. T. I. J. S. D. H. W. E. N. Robert Schöne, "Extending the Functionality of Score-P through Plugins: Interfaces and Use Cases," *Tools for High Performance Computing 2016,* pp. 59-82, 2017.

[2] R. S. M. B. A. G. D. H. Thomas Ilsche, "lo2s—Multi-core System and Application Performance Analysis for Linux," *IEEE International Conference on Cluster Computing (CLUSTER), ,* 2017.

[3] M. H. R. S. M. B. D. H. Thomas Ilsche, "Powernightmares: The challenge of efficiently using sleep states on multi-core systems," *European Conference on Parallel Processing,* pp. 623-635, 2017.