GA no. 671657

D5.2
Extended READEX Test-Suite with Manually Tuned Applications

Document type:	Report
Dissemination level:	Other
Work package:	WP5
Editor:	Lubomír Říha (IT4I-VSB)
Contributing partners:	IT4I-VSB, ICHEC-NUIG
Reviewer:	Venkatesh Kannan (ICHEC-NUIG)
Version:	1.0

Document history

Version	Date	Author/Editor	Description
0.1	27/01/18	Lubomír Říha, Jan Zapletal, Ondřej Vysocký (IT4I-VSB)	1st Draft
0.2	08/02/18	Lubomír Říha, Jan Zapletal, Ondřej Vysocký (IT4I-VSB)	2nd Draft
1.0	26/02/18	Lubomír Říha, Jan Zapletal, Ondřej Vysocký (IT4I-VSB)	Final version

Executive Summary

The objective of WP5 is to evaluate READEX in comparison to a default execution and a manual tuning of applications. To do so, READEX defines a test-suite of proto applications.

In the previous deliverable D5.1, we demonstrated the tuning potential of the applications. In D5.2, we extended the list of benchmarks and provide scripts for manual tuning as well as an initial version for the automated tuning with READEX. This document accompanies D5.2 and describes the current state of the test suite.

It is assumed that the reader of this document has already a good understanding of the READEX concepts from reading previous deliverables such as D4.1 [2] and D4.2 [1].

Contents

1 Introduction

This documents accompanies Deliverable D5.2 *Extended READEX test-suite with manually tuned applications.* The actual deliverable, the extended READEX test suite, is available in a git repository and can be downloaded and used. This document is structured as follows. Section 2 describes the structure of the repository and the usage of the benchmarks and scripts that are part of it. Section 3 briefly describes each analysed application. Section 4 shows an example analysis of AMG2013 application from the apps repository. Section 5 closes the document.

2 The `readex-apps` Repository

The repository of test applications is located at `git@acratus.ichec.ie:readex-apps.git`. Currently, the repository holds several benchmark and production applications listed in Table 1.

Apart from the source files, each application's directory contains bash scripts to compile the application in several configurations. Namely, this includes the compilation of the uninstrumented (plain) version used as a reference and compilation for every tool in the READEX tool suite (scorep-autofilter, readex-dyn-detect, PTF, and RRL). The compilation scripts have been prepared both for manual and automatic instrumentation by Score-P (All the applications are manually instrumented, which is inserted in to the code on demand during compilation.). While the former approach serves for the evaluation of possible savings, the latter can be used to evaluate the automated READEX tool suite in comparison to the manual effort. In addition to the compilation scripts the repository contains scripts to launch the individual READEX tools and a brief `READEX_README.txt` help file that describes the usage.

application	**path**	**maintained by**
AMG2013	benchmark_apps/amg2013	IT4I
Blasbench	benchmark_apps/blasbench	IT4I
Kripke	benchmark_apps/kripke	IT4I
Lulesh	benchmark_apps/lulesh	ICHEC,TUM
MCBenchmark	benchmark_apps/mcbenchmark	IT4I
NPB3.3	benchmark_apps/NPB3.3-MZ-MPI	TUD
BEM4I	production_apps/BEM4I	IT4I
ESPRESO	production_apps/ESPRESO/espreso_readex_new	IT4I
INDEED	production_apps/Indeed_for_READEX	GNS
OpenFOAM	production_apps/OPENFOAM	IT4I

Table 1: List of applications in the readex-apps repository.

To set up the environment source script files from the `readex_env` directory via `source`. On the READEX test system taurus, `readex_env` is a symbolic link to a directory in `readex-apps/readex-repository/env` and allows for easy switching among different compilation environments (Intel, GNU). This approach ensures that the application uses the current state of the continuously integrated READEX tool suite and in addition provides a rather easy way for porting the test suite for other clusters equipped with the tool suite: User only have to adapt the sourced files, update the symbolic link to a preferred location, and use the provided compilation and run scripts without significant changes (the run scripts expect to be launched by the SLURM scheduler).

To summarize, the repository contains the files listed in Tables 2, 3.

file	**description**
`set_env_cxx.source`	general environment for the compiler
`set_env_plain.source`	environment for the uninstrumented version
`set_env_saf.source`	environment for scorep-autofilter
`set_env_rdd.source`	environment for readex-dyn-detect
`set_env_ptf_hdeem.source`	environment for PTF with HDEEM
`set_env_ptf_rapl.source`	environment for PTF with RAPL
`set_env_rrl.source`	environment for RRL

Table 2: Summary of environment files available in `readex-apps/readex-repository/env`.

3 Benchmarks description

3.1 AMG2013

AMG2013 is a parallel algebraic multigrid solver for linear systems arising from problems on unstructured grids. It has been derived directly from the BoomerAMG solver in the hypre library, a large linear solver library that is being developed in the Center for Applied Scientific Computing (CASC) at LLNL. The driver provided in the benchmark can build various test problems. FOr our tests the Laplace type problem on an unstructured domain with various jumps and an anisotropy in one part is used.

3.2 Blasbench

Blasbench is a simple artificial benchmark designed to simulate different workloads including compute, memory, I/O, or communication bound regions. The users can define their own mixture of such regions to obtain a highly dynamic code. The benchmark thus can serve as a best scenario for the tuning by the READEX tool suite.

file	**description**
`READEX_README.txt`	a help file
`set_env_XXX.source`	application specific environment
`compile_for_plain.sh`	compilation of the uninstrumented version
`compile_for_saf.sh`	compilation for scorep-autofilter
`compile_for_rdd.sh`	compilation for readex-dyn-detect, automatic instr.
`compile_for_rdd_manual.sh`	compilation for readex-dyn-detect, manual instr.
`compile_for_ptf.sh`	compilation for PTF and RRL, automatic instr.
`compile_for_ptf_manual.sh`	compilation for PTF and RRL, manual instr.
`run_plain.sh`	runs the uninstrumented version
`run_saf.sh`	runs scorep-autofilter
`run_rdd.sh`	runs readex-dyn-detect
`extend_readex_config.sh`	extends readex_config.xml produced by readex-dyn-detect
`run_ptf.sh`	runs PTF
`run_rrl.sh`	runs the application with RRL

Table 3: Summary of files specific for each application.

3.3 Kripke

Kripke is a simple, scalable, 3D Sn deterministic particle transport code. Its primary purpose is to research how data layout, programming paradigms and architectures effect the implementation and performance of Sn transport. A main goal of Kripke is investigating how different data-layouts affects instruction, thread and task level parallelism, and what the implications are on overall solver performance.

3.4 Lulesh

Livermore Unstructured Lagrangian Explicit Shock Hydrodynamics (LULESH) is part of the ALE3D Software Infrastructure. According to the documentation, it is "one of five challenge problems in the DARPA UHPC program and has since become a widely studied proxy application in DOE co-design efforts for exascale." It is implemented in C++ and uses OpenMP and MPI for parallelization.

3.5 MCBenchmark

The Monte Carlo Benchmark (MCB) is intended for use in exploring the computational performance of Monte Carlo algorithms on parallel architectures. It models the solution of a simple heuristic transport equation using a Monte Carlo technique.

3.6 NPB3.3

The NAS Parallel Benchmarks are a set of micro benchmarks that mimic real applications used in HPC. In our test suite, we use the hybrid parallel benchmarks written in FORTRAN and parallelized with MPI and OpenMP.

3.7 BEM4I

BEM4I is a solver for partial differential equations based on the boundary element method and is under development at IT4Innovations. Contrary to alternative finite element solvers, BEM4I produces dense matrices and due to the nature of boundary integral equations the assembly of system matrices is more or less compute bound. This is in contrast to the iterative solver used for the solution of the resulting system of linear equations which is usually memory bound due to matrix vector multiplications. BEM4I is thus a suitable library to be involved in the test apps repository.

3.8 ESPRESO

The ESPRESO library is a combination of Finite Element (FEM) and Boundary Element (BEM) tools and TFETI/HTFETI solvers. It supports FEM and BEM (uses BEM4I library) discretization for Advection-diffusion equation, Stokes flow and Structural mechanics. The ESPRESO solver is a parallel linear solver, which includes a highly efficient MPI communication layer designed for massively parallel machines with thousands of compute nodes. The parallelization inside a node is done using OpenMP.

3.9 INDEED

INDEED is a sheet metal forming simulation software that combines ease of use with high quality simulation. Since it is a commercial tool, we do not provide the sources in the repository, but only the diff that has to be applied for manual instrumentation.

3.10 OpenFOAM

OpenFOAM is an open source C++ toolbox for computational fluid dynamics (CFD). OpenFOAM does not have a generic solver applicable to all cases, but there is a long list of solvers each for specific class of problems e.g. compressible and incompressible flow, multiphase flow, combustion, particle-tracking flows heat transfer and many more.

4 Example Analysis for AMG2013

In this section, we describe how one benchmark from the repository is tuned with READEX by using the script described in the previous section.

The AMG2013 benchmark is located at

```
readex-apps/readex-repository/benchmark_apps/amg2013
```

To apply the READEX tool suite on AMG2013 follow the steps listed below.

1. Update the symbolic link `readex_env` to point to the environment of your choice. The current options on Taurus are
   - `ln -sfnv ../../env/bullxmpi1.2.8.4_gcc6.3.0/ readex_env`
   - `ln -sfnv ../../env/intelmpi2017.2.174_intel2017.2.174/ readex_env`
2. Compile the uninstrumented version and perform a testing run by
   - `./compile_for_plain.sh` (the executable created is `./test/amg2013_plain`),
   - `sbatch ./run_plain.sh`
3. Compile the code for `scorep-autofilter` and run the filtering by
   - `./compile_for_saf.sh` (the executable created is `./test/amg2013_saf`),
   - `sbatch ./run_saf.sh` (output in `scorep.filt`).

   Note that this step is only relevant for automatic instrumentation. In case of manual instrumentation only the manually inserted regions are taken into account and no filtering is necessary.
4. Compile the code for `readex-dyn-detect` and run the dynamism detection tool by
   - `./compile_for_rdd.sh` for automatic instrumentation by Score-P taking into account the filter file produced above or `./compile_for_rdd_manual.sh` for a manually instrumented version (the executable created is `./test/amg2013_rdd`),
   - `sbatch ./run_rdd.sh` (output in `readex_config.xml`).
5. The output `readex_config.xml` contains the description of significant regions for PTF. It is up to the user to specify the tuning parameters, the tuning strategy and further details described in `how_to_use_readex_toolsuite.pdf`. A script for automatic extension of the configuration file is provided for each app in the test suite and can be called by
   - `./extend_readex_config.sh` (output in `readex_config_ptf.xml`).

6. Compile the code for `PTF` and run the analysis by
   - `./compile_for_ptf.sh` for automatic instrumentation by Score-P taking into account the filter file produced above or `./compile_for_ptf_manual.sh` for a manually instrumented version (the executable created is `./test/amg2013_ptf`),
   - `sbatch ./run_ptf.sh` (output in `tuning_model.json`).
7. Since the compilation for RRL is the same as for PTF, the binary `./test/amg2013_ptf` can be reused to test dynamic switching according to `tuning_model.json` by RRL. To run the tuned application and compare the energy consumption to the uninstrumented version (`./test/amg2013_plain`) use
   - `sbatch ./run_rrl.sh` (output of uninstrumented and RRL tuned versions measured by HDEEM in `amg2013_rrl_plain_hdeem.out` and `amg2013_rrl_rrl_hdeem.out`, respectively).

amg2013_rrl	1	4	8	2	12	109429	22456.414
amg2013_rrl	2	4	8	2	12	107916	22354.112
amg2013_rrl	3	4	8	2	12	107463	22471.416

Table 4: Energy measurements stored in amg2013_rrl_plain_hdeem.out.

amg2013_rrl	1	4	8	2	12	126788	20952.191
amg2013_rrl	2	4	8	2	12	126162	20926.660
amg2013_rrl	3	4	8	2	12	126297	20907.157

Table 5: Energy measurements stored in amg2013_rrl_rrl_hdeem.out.

The format of the files `amg2013_rrl_XXX_hdeem.out` follow Tables 4, 5. The respective columns contain the name of the test, test number, number of nodes employed, total number of MPI processes, number of MPI processes per node, number of OpenMP threads per MPI process, runtime of the app in milliseconds, and energy consumption in Joules. It can be seen from the two tables that while the runtime increases for the tuned version (second-last column), the energy consumed decreases (last column).

5 Summary

This document shortly described how the READEX test application repository is structured and how it can be used. The actual deliverable is available for download in a git repository.

References

[1] Andreas Gocht, Umbreen Sabir Mian, Michael Lysaght, Venkatesh Kannan, Michael Gerndt, Anamika Chowdhury, Madhura Kumaraswamy, Per Gunnar Kjeldsberg, Mohammed Sourouri, and Nico Reissmann. D4.2: Prototype READEX tool suite. Technical report, ICHEC, TUD, TUM, NTNU, IT4I, Intel, GNS, 2017.

[2] Michael Lysaght, Kashif Iqbal, Joseph Schuchart, Andreas Gocht, Michael Gerndt, Anamika Chowdhury, Madhura Kumaraswamy, Per Gunnar Kjeldsberg, Magnus Jahre, Mohammed Sourouri, David Horak, Lubomir Riha, Radim Sojka, Jakub Kruzik, Kai Diethelm, and Othman Bouizi. D4.1: Concepts for the READEX tool suite. Technical report, ICHEC, TUD, TUM, NTNU, IT4I, Intel, gns, 2016.