READEX Tool Suite Installation Guide

Contents

1. About different compilers **5**

2. Score-P **6**
- 2.1. Requirements . . . 6
- 2.2. Download . . . 6
- 2.3. Preparing the Score-P directory . . . 6
- 2.4. Configuring and installing Score-P . . . 6

3. PTF **8**
- 3.1. Requirements . . . 8
- 3.2. Download . . . 8
- 3.3. Preparing the PTF directory . . . 9
- 3.4. Configuring and installing PTF . . . 9
  - 3.4.1. Managing starter plugins . . . 10

4. RRL **11**
- 4.1. Requirements . . . 11
- 4.2. Download . . . 11
- 4.3. Preparing the RRL directory . . . 11
- 4.4. Configuring and installing the RRL . . . 11

5. PCPs **12**
- 5.1. Requirements . . . 12
- 5.2. Download . . . 12
- 5.3. Configuring and installing the PCP's . . . 12

6. ATP **13**
- 6.1. Requirements . . . 13
- 6.2. Download . . . 13
- 6.3. Preparing the ATP library directory . . . 13
- 6.4. Configuring and installing the ATP library . . . 14

7. Cluster Prediction **15**
- 7.1. Requirements . . . 15
- 7.2. Download . . . 15
- 7.3. Preparing the Cluster Prediction library directory . . . 15
- 7.4. Configuring and installing the Cluster Prediction library . . . 15

8. Score-P Metric Plugins **16**
- 8.1. Processor Energy Event Plugin . . . 16
  - 8.1.1. Installation . . . 16
  - 8.1.2. Usage in READEX . . . 17
- 8.2. HDEEM Energy Measurement Plugin . . . 18
  - 8.2.1. Installation . . . 18

8.2.2. Usage in READEX . . . 19
8.3. EXAMON Energy Event Plugin . . . 19
8.3.1. Installation . . . 19
8.3.2. Usage in READEX . . . 20

A. READEX tool suite installation on Taurus cluster (TU Dresden) 21
A.1. The module environment . . . 21
A.2. Allocating a node . . . 21
A.3. Score-P . . . 21
A.3.1. Modules . . . 21
A.3.2. Download . . . 21
A.3.3. Preparing the Score-P directory . . . 22
A.3.4. Configuring and installing Score-P . . . 22
A.3.5. Creating the related Score-P module file . . . 22
A.4. Using Score-P . . . 23
A.5. PTF . . . 23
A.5.1. Modules . . . 23
A.5.2. Download . . . 24
A.5.3. Preparing the PTF directory . . . 24
A.5.4. Configuring and installing PTF . . . 25
A.5.5. Creating the related PTF module file . . . 25
A.5.6. Using PTF . . . 26
A.6. RRL . . . 26
A.6.1. Modules . . . 26
A.6.2. Download . . . 27
A.6.3. Preparing the RRL directory . . . 27
A.6.4. Configuring and installing the RRL . . . 27
A.6.5. Creating the related RRL module file . . . 28
A.6.6. Using RRL . . . 29
A.6.7. Building doc . . . 29
A.7. PCP's . . . 29
A.7.1. Modules . . . 29
A.7.2. Download . . . 29
A.7.3. Configuring and installing the PCP's . . . 30
A.7.4. Creating the related PCP's module file . . . 30
A.7.5. Using PCP's . . . 31
A.8. ATP . . . 31
A.8.1. Modules . . . 31
A.8.2. Download . . . 31
A.8.3. Preparing the ATP directory . . . 31
A.8.4. Configuring and installing the ATP . . . 32
A.8.5. Creating the related ATP module file . . . 32
A.8.6. Using ATP . . . 33
A.9. Cluster Prediction . . . 33
A.9.1. Modules . . . 33

A.9.2. Download 33
A.9.3. Preparing the Cluster Prediction directory 33
A.9.4. Configuring and installing the Cluster Prediction library 34
A.9.5. Creating the related Cluster Prediction module file 34
A.9.6. Using Cluster Prediction 35

B. READEX Docker Image **36**

C. READEX Integrated Installation Script **37**
C.1. Requirements 37
C.2. Usage 37

1. About different compilers

During the development of the READEX tool suite it turned out that it is not obvious to use the same compiler for everything. Please be aware that there are some incompatibilities between different compilers. Specially for C++ this is a problem, as different compilers may have different ABI versions, which will lead to linking errors. Moreover, a Score-P version compiled for GCC will not work with the Intel compiler and vice versa.

Therefore, it is really important that you are using the same compiler for the entire READEX tool suite as well as for the applications on which you are going to use the tool suite. Again, please use the same compiler for Score-P, PTF, the RRL, the PCPs, the ATP library and your application.

2. Score-P

This section outlines how to build Score-P for READEX.

2.1. Requirements

The build procedure for the READEX version of Score-P requires the following tools to be already installed:

- Intel compiler version $\geq$ 2017.2.174/2018.1.163 or GCC (G++ and GFortran) version $\geq$ 6.3.0/7.1.0. Other Intel or GCC compiler versions can also be used, but have not been explicitly tested by the READEX developers.
- PAPI version $\geq$ 5.5.1 (http://icl.utk.edu/papi/software/).
- Bison version $\geq$ 3.0.4 (https://www.gnu.org/software/bison/).

2.2. Download

Please download the version of Score-P for READEX from the following location and unpack it:

```
http://www.readex.eu/index.php/dissemination/software/ScoreP.tar.gz
tar -xzvf ScoreP.tar.gz
```

2.3. Preparing the Score-P directory

Please prepare the Score-P build directory as follows:

```
cd ScoreP
mkdir build
cd build
```

2.4. Configuring and installing Score-P

You may use the following naming scheme for the "--prefix" argument:

```
<Desired path for Score-P installation>/scorep/scorep-readex-<mpi version>-<compiler
    version>

<mpi version>:          for example, intelmpi2017.2.174
<compiler version>:     for example, intel2017.2.174
```

To run configure please do:

```
../configure        '--prefix=<Desired_path_for_Score-P_installation>/scorep/scorep-readex-<
    mpi_version>-<compiler_version>/' \
                    '--enable-backend-test-runs' \
                    '--with-nocross-compiler-suite=<gcc|intel>' \
                    '--with-mpi=<bullxmpi|intel3|...>' \
                    '--with-papi-header=<path_to_PAPI_include>' \
                    '--with-papi-lib=<path_to_PAPI_lib>' \
                    '--with-libbfd=no' \
                    '--disable-silent-rules' \
                    '--without-gui' \
                    '--enable-static' \
                    '--enable-shared' \
                    '--enable-debug'\
                    'CFLAGS=-g -O3 -fno-omit-frame-pointer' \
```

```
                    'CXXFLAGS=-g -O3 -fno-omit-frame-pointer' \
make -j
make install
```

For more details on installing Score-P, refer to Section 2.1 in the Score-P User Manual that can be downloaded from https://github.com/readex-eu/readex-scorep/blob/master/doc/pdf/scorep.pdf.

3. PTF

This section outlines how to build the Periscope Tuning Framework (PTF) for READEX.

3.1. Requirements

The build procedure for the READEX version of PTF requires the following tools to be already installed:

- Score-P extension for READEX as described in Section 2.
- Intel compiler version $\geq$ 2017.2.174/2018.1.163 or GCC (G++ and GFortran) version $\geq$ 6.3.0/7.1.0. Other Intel or GCC compiler versions can also be used, but have not been explicitly tested by the READEX developers.
- PAPI version $\geq$ 5.5.1 (http://icl.utk.edu/papi/software/).
- Boost version $\geq$ 1.62.0 (https://www.boost.org/users/download/).
- Cereal version $\geq$ 1.2.1 (https://github.com/USCiLab/cereal).
- Bison version $\geq$ 3.0.4 (https://www.gnu.org/software/bison/).
- Python version $\geq$ 3.6 (https://www.python.org/downloads/).
- Ace version 6.3.3 (http://www.dre.vanderbilt.edu/~schmidt/ACE.html).
- Flex version $\geq$ 2.5.39 (https://github.com/westes/flex).
- Score-P developer tools containing patched Libtool version $\geq$ 2.4.6, Autoconf version $\geq$ 2.69, Automake version $\geq$ 1.13.4, Doxygen version $\geq$ 1.8.10 and M4 version $\geq$ 1.4.16.

Please make sure that the Score-P version is also compiled with the same compiler as the one used for PTF.

NOTE: Please note that it may be necessary to load the compiler's environment (loading modules or adding directories/locations to environment variables) before loading the environment for the Boost library when installing PTF.

3.2. Download

Please download the READEX branch of PTF and Score-P development tools from the following locations, and unpack them:

```
http://www.readex.eu/index.php/dissemination/software/PTF.tar.gz
tar -xzvf PTF.tar.gz

http://www.readex.eu/index.php/dissemination/software/scorep-dev-06.tar.gz
tar -xzvf scorep-dev-06.tar.gz
```

3.3. Preparing the PTF directory

Please bootstrap PTF as follows:

```
cd PTF
./bootstrap
mkdir build
cd build
```

3.4. Configuring and installing PTF

Add the paths to the libraries and executables of the Score-P developer tools to the `PATH` and `LD_LIBRARY_PATH` environment variables:

```
export PATH=<path to Score-P developer tools>/bin:$PATH
export LD_LIBRARY_PATH=<path to Score-P developer tools>/lib:$LD_LIBRARY_PATH
```

You may use the following naming scheme for "--prefix":

```
<Desired path for PTF installation>/ptf/ptf_readex_<mpi version>_<compiler version>
    _with_scorep/

<compiler version>     for example: intel2017.2.174
<mpi version>          for example: intelmpi2017.2.174
```

To configure and install PTF please now do:

```
../configure         '--prefix=<Desired_path_for_PTF_installation>/ptf/ptf_readex_<mpi_
    version>_<compiler_version>_with_scorep/' \
                     --enable-developer-mode \
                     --with-starter=<slurm|superMUC|interactive> \
                     --with-ace-include=<path to ACE include> \
                     --with-ace-lib=<path to ACE lib>\
                     --with-boost-include=<path to Boost include>\
                     --with-boost-lib=<path to Boost lib>\
                     --with-cube-include=<path to Cube include> \
                     --with-cube-lib=<path to Cube lib> \
                     --with-scorep-include=<path to Score-P include> \
                     --with-scorep-lib=<path to Score-P lib> \
                     --with-cereal-include=<path to Cereal include>

make -j 24
make install
```

If you want to use the Intel compiler to compile PTF, please add the following to "../configure" :

```
    --with-compiler-suite=intel
```

Please be aware that no special Cube module is needed if you are using the same compiler for Score-P and PTF. Therefore the options for `--with-cube-include` and `--with-cube-lib` are the same as for `--with-scorep-include` and `--with-scorep-lib`, respectively.

Known issue: It is observed that when building PTF with an ACE library with some versions $\geq$ `6.3.3`, a build error may occur due to `ambiguating new declaration of 'int operator<<(ACE_OutputCDR&, const string&)'`. To avoid this, it is advisable to use ACE library version `6.3.3`.

3.4.1. Managing starter plugins

Since PTF is an automatic distributed tool, the frontend starts the parallel application. To do so, it has to use the right command and parameters which are clearly depending on your batch system and your local installation. PTF comes with several starters, e.g., a generic starter named "interactive", another one for the SuperMUC cluster at LRZ and a generic one for Slurm batch system.

PTF uses plugins for defining the start command for the application and the analysis agents. You select the plugin to be used on your machine in the configuration step.

A starter plugin defines the command to start the application. For example, on SuperMUC you use `mpiexec`, while on a Slurm-based system you use `srun`. Furthermore, you give the necessary parameters. For example, the Slurm plugin uses a separate node for running the PTF processes while on SuperMUC the processes are assigned to the first node where your batch script is started. As said, decisions taken here are clearly dependent on your machine setup.

Please have a look at the available plugins. It might be necessary to create your own plugin using one of the available plugins as a template.

In order to create a new starter plugin on a new machine, follow these steps inside the PTF source:

1. Go to the root directory of the PTF repository.
2. Now go to the `starterplugin` folder.
3. Create a new folder with your machine specific folder name.
4. Copy your chosen `ptf-plugin.cc` template file from the "interactive" folder and paste it into your created folder.
5. Now make the specific changes to start the application and the analysis agents.

Next change the configure command's `--with-starter` option with the newly created starter name (for example: `--with-starter=slurm`) during the configuration step.

In case you need help in developing your own starter, please contact the PTF support at periscope@lists.lrz.de.

For more details on installing PTF, refer to Section 2.3 in the PTF Installation Guide which is available for download at http://periscope.in.tum.de/releases/latest/pdf/PTF_Installation_Guide.pdf.

4. RRL

This section outlines how to build the READEX Runtime Library (RRL).

4.1. Requirements

The build procedure for RRL requires the following tools to be already installed:

- Score-P extension for READEX as described in Section 2.
- Intel compiler version $\geq$ 2017.2.174/2018.1.163 or GCC (G++ and GFortran) version $\geq$ 6.3.0/7.1.0. Other Intel or GCC compiler versions can also be used, but have not been explicitly tested by the READEX developers.
- CMake version $\geq$ 3.11 (https://cmake.org/download/).

Please make sure that the Score-P version is also compiled with the same compiler as the one used for RRL.

4.2. Download

Please download RRL from the following location and unpack it:

```
http://www.readex.eu/index.php/dissemination/software/RRL.tar.gz
tar -xzvf RRL.tar.gz
```

4.3. Preparing the RRL directory

Please switch to the RRL directory of the downloaded repository:

```
cd RRL
mkdir build
cd build
```

4.4. Configuring and installing the RRL

You may use the following naming scheme for `--DCMAKE_INSTALL_PREFIX`:

```
<Desired path for RRL installation>/readex-rrl/rrl-readex-<mpi version>-<compiler version>

<compiler version>      for example: intel2017.2.174
<mpi version>           for example: intelmpi2017.2.174
```

Now, to build the RRL please do:

```
cmake ../ -DCMAKE_INSTALL_PREFIX=<Desired path for RRL installation>/readex-rrl/rrl-readex-
    <mpi version>-<compiler version> -DDISABLE_CALIBRATION=ON
make -j 24
make install
```

NOTE: To build RRL with calibration enabled, omit the option `-DDISABLE_CALIBRARION=ON` in the build step and install the dependencies `x86_adapt` and PAPI version $\geq$ 5.5.1. You may install `x86_adapt` from https://github.com/tud-zih-energy/x86_adapt, and include the path to `libx86_adapt.so` in your `LD_LIBRARY_PATH` as shown below:

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:<path to libx86_adapt.so>
```

5. PCPs

This section outlines how to build the different Parameter Control Plugins (PCPs) for READEX.

5.1. Requirements

The build procedure for the PCPs requires the following tools to be already installed:

- READEX Runtime Library (RRL) as described in Section 4.
- Intel compiler version $\geq$ 2017.2.174/2018.1.163 or GCC (G++ and GFortran) version $\geq$ 6.3.0/7.1.0. Other Intel or GCC compiler versions can also be used, but have not been explicitly tested by the READEX developers.

Please make sure that the RRL version is also compiled with the same compiler as the one used for the PCPs.

5.2. Download

Please download the PCP from the following location and unpack it:

```
http://www.readex.eu/index.php/dissemination/software/PCPs.tar.gz
tar -xzvf PCPs.tar.gz
```

5.3. Configuring and installing the PCP's

You may use the following naming scheme for the argument of the build script:

```
<Desired path for PCPs installation>/parameter_control_plugins/pcp_readex_<mpi version>_<
    compiler version>

<compiler version>     for example: intel2017.2.174
<mpi version>          for example: intelmpi2017.2.174
```

To build the PCPs please now do:

```
export RRL_INC=<Path to RRL include directory>
cd PCPs
./build.sh <Desired path for PCPs installation>/parameter_control_plugins/pcp_readex_<mpi
    version>_<compiler version>
```

6. ATP

This section outlines how to build the Application Tuning Parameter library (ATP library) for READEX.

6.1. Requirements

The build procedure for the ATP library requires the following tools to be already installed:

- READEX Runtime Library (RRL) as described in Section 4.
- Intel compiler version $\geq$ 2017.2.174/2018.1.163 or GCC (G++ and GFortran) version $\geq$ 6.3.0/7.1.0. Other Intel or GCC compiler versions can also be used, but have not been explicitly tested by the READEX developers.
- CMake version $\geq$ 3.11 (https://cmake.org/download/).
- LUA interpreter and libraries version $\geq$ 5.1 (https://www.lua.org/download.html). Prerequisites for LUA are:
  - `readline` library (https://tiswww.case.edu/php/chet/readline/rltop.html).
  - `ncurses` library (https://www.gnu.org/software/ncurses/).

Please make sure that the RRL version is also compiled with the same compiler as the one used for the ATP library.

If LUA is installed in a custom location set LUA_DIR environment variable for CMake to be able to find the correct installation.

6.2. Download

Please download the ATP library from the following location and unpack it:

```
http://www.readex.eu/index.php/dissemination/software/ATP.tar.gz
tar -xzvf ATP.tar.gz
```

6.3. Preparing the ATP library directory

Please do:

```
cd ATP
mkdir build
cd build
```

6.4. Configuring and installing the ATP library

You may use the following naming scheme for `-DCMAKE_INSTALL_PREFIX`:

```
<Desired path for ATP library installation>/readex-atp/atp_readex_<mpi version>_<compiler
    version>

<compiler version>      for example: intel2017.2.174
<mpi version>           for example: intelmpi2017.2.174
```

To build the ATP please now do:

```
cmake ../ -DCMAKE_INSTALL_PREFIX=<Desired path for ATP library installation>/readex-atp/
    atp_readex_<mpi version>_<compiler version>
make -j 24
make install
```

7. Cluster Prediction

This section outlines how to build the Cluster Prediction library for READEX.

7.1. Requirements

The build procedure for the Cluster Prediction library requires the following tools to be already installed:

- READEX Runtime Library (RRL) as described in Section 4.
- Intel compiler version $\geq$ 2017.2.174/2018.1.163 or GCC (G++ and GFortran) version $\geq$ 6.3.0/7.1.0. Other Intel or GCC compiler versions can also be used, but have not been explicitly tested by the READEX developers.
- CMake version $\geq$ 3.11 (https://cmake.org/download/).
- PAPI version $\geq$ 5.5.1 (http://icl.utk.edu/papi/software/).

Please make sure that the RRL version is also compiled with the same compiler as the one used for the Cluster Prediction library.

7.2. Download

Please download Cluster Prediction library from the following location and unpack it:

```
http://www.readex.eu/index.php/dissemination/software/Cluster_Prediction.tar.gz
tar -xzvf Cluster_Prediction.tar.gz
```

7.3. Preparing the Cluster Prediction library directory

Please do:

```
cd Cluster_Prediction
mkdir build
cd build
```

7.4. Configuring and installing the Cluster Prediction library

You may use the following naming scheme for `-DCMAKE_INSTALL_PREFIX`:

```
<Desired path for Cluster Prediction library installation>/cluster_prediction/
      cluster_prediction_readex_<mpi version>_<compiler version>

<compiler version>   for example: intel2017.2.174
<mpi version>     for example: intelmpi2017.2.174
```

To build the Cluster Prediction library please now do:

```
cmake ../ -DCMAKE_INSTALL_PREFIX=<Desired path for Cluster Prediction library installation
     >/cluster_prediction/cluster_prediction_readex_<mpi version>_<compiler version> -
     DCMAKE_PREFIX_PATH=<papi installation root folder>
make
make install
```

8. Score-P Metric Plugins

The Score-P metric plugin interface makes it possible for programmers to increase the event stream with metric data from additional data sources that are otherwise not accessible for Score-P.

READEX currently supports various backends for energy measurement. This includes: Intel RAPl, AMD RAPL, and AMD APM (via the Processor Energy Event Plugin), Bull/ATOS HDEEM and ANTAREX EXAMON. In this section, we describe how they can be installed with brief usage details.

8.1. Processor Energy Event Plugin

Version 2 of this metric plugin available at https://github.com/readex-eu/scorep_plugin_x86_energy/tree/x86_energy_v_2 provides energy measurement for most contemporary x86-based processors and platforms.

8.1.1. Installation

To compile this plugin, you need:

- A C++ 14 compiler
- CMake
- Score-P
- A processor of the following architectures:
  - Intel with Sandy Bridge architecture or newer
  - AMD Family15h (Bulldozer) or AMD Zen
- Each of the following kernel modules will grant you energy measurement access:
  - `intel_powerclamp` kernel module (Intel architectures, recommended, part of recent Linux kernels. There's a backport for older kernels available at http://content.allinea.com/downloads/allinea-powercap-backport-20150601.tar.bz2)
  - `intel_rapl_perf` kernel module (Intel architectures, part of recent Linux kernels)
  - `msr` and `msr-safe` kernel module (Intel architectures, https://github.com/LLNL/msr-safe)
  - `x86_adapt` kernel module (Intel architectures, AMD Zen, https://github.com/tud-zih-energy/x86_adapt)
  - `likwid` plus `msr`/`msr-safe` kernel module (Intel architectures, https://github.com/RRZE-HPC/likwid)
  - `fam15h_power` kernel module (AMD Bulldozer, part of recent Linux kernels)

To build the plugin as follows:

1. Create a build directory

```
mkdir build
cd build
```

2. Invoke Cmake

```
cmake ..
```

   The following settings can be customized:

   - `SCOREP_CONFIG`: Path to the `scorep-config` tool including the file name. If you have `scorep-config` in your `PATH`, it should be found by CMake.
   - `CMAKE_INSTALL_PREFIX`: Directory where the resulting plugin will be installed (`lib/` suffix will be added)
   - `ENABLE_MPI` (only applicable to the `examon_sync_plugin`): Enables MPI communication for the sync plugin, and allows to run more than one MPI process per node. This is mandatory for READEX!

3. Invoke make

```
make
```

4. Install the files

```
make install
```

8.1.2. Usage in READEX

For READEX purposes, we use the `x86_energy_sync_plugin`, which enables us to attribute energy to specific regions. To use the plugin, it must be enabled during runs with PTF. To do so, use the following environment variables:

```
export SCOREP_METRIC_PLUGINS=x86_energy_sync_plugin
export SCOREP_METRIC_PLUGINS_SEP=";"
export SCOREP_METRIC_X86_ENERGY_SYNC_PLUGIN="BLADE/E"
export SCOREP_METRIC_X86_ENERGY_PLUGIN_INTERVALL_US=0
export SCOREP_METRIC_X86_ENERGY_SYNC_PLUGIN_OFFSET=70 # This is the offset that is set to
     the sum of package and DRAM consumptions to compute BLADE power and highly system
     dependent
```

Furthermore, the metric source definition of your `readex_config.xml` should look like this:

```
<metricPlugin>
  <name>x86_energy_sync_plugin</name>
</metricPlugin>
<metrics>
  <node_energy>x86_energy/BLADE/E</node_energy>
</metrics>
```

8.2. HDEEM Energy Measurement Plugin

High Definition Energy Efficiency Monitoring (HDEEM) is the power measurement infrastructure provided by contemporary blades from the HPC vendor Bull(Atos). It provides a high resolution with inband and out-of band interfaces for accessing energy and power data. More details can be found at https://tu-dresden.de/zih/forschung/projekte/hdeem.

8.2.1. Installation

To compile this plugin, you need:

- A C++ 14 compiler
- Score-P Version 2+ (`SCOREP_METRIC_PLUGIN_VERSION` ≥ 1)
- The hdeem library and header files by BULL/ATOS

To build the plugin as follows:

1. Create a build directory

```
mkdir build
cd build
```

2. Invoke Cmake

```
cmake ..
```

   The following settings can be customized:

   - `SCOREP_CONFIG`: Path to the `scorep-config` tool including the file name. If you have `scorep-config` in your `PATH`, it should be found by CMake.
   - `CMAKE_INSTALL_PREFIX`: Directory where the resulting plugin will be installed (`lib/` suffix will be added)
   - `ENABLE_MPI` (only applicable to the `examon_sync_plugin`): Enables MPI communication for the sync plugin, and allows to run more than one MPI process per node. This is mandatory for READEX!
   - `HDEEM_BMC_USER HDEEM_BMC_PASS`: Required for out of band access
   - `HDEEM_INCLUDE_DIRS`: Directory where `hdeem.h` is located
   - `HDEEM_LIBRARIES`: The full path of `libhdeem.so`, including the file name, e.g. `/usr/local/hdeem/libhdeem.so.1`

3. Invoke make

```
make
```

4. Install the files

```
make install
```

8.2.2. Usage in READEX

For READEX purposes, we use the `hdeem_sync_plugin`, which enables us to attribute energy to specific regions. To use the plugin, it must be enabled during runs with PTF. To do so, use the following environment variables:

```
export SCOREP_METRIC_PLUGINS=hdeem_sync_plugin
export SCOREP_METRIC_PLUGINS_SEP=";"
export SCOREP_METRIC_HDEEM_SYNC_PLUGIN_CONNECTION="INBAND"
export SCOREP_METRIC_HDEEM_SYNC_PLUGIN_VERBOSE="WARN"
export SCOREP_METRIC_HDEEM_SYNC_PLUGIN_STATS_TIMEOUT_MS=1000
```

Furthermore, the metric source definition of your `readex_config.xml` should look like this:

```
<metricPlugin>
  <name>hdeem_sync_plugin</name>
</metricPlugin>
<metrics>
  <node_energy>hdeem/BLADE/E</node_energy>
  <cpu0_energy>hdeem/CPU0/E</cpu0_energy>
  <cpu1_energy>hdeem/CPU1/E</cpu1_energy>
</metrics>
```

8.3. EXAMON Energy Event Plugin

EXAMON is a scalable HPC measurement infrastructure with different data sources and plugins. More details can be found at https://github.com/EEESlab/examon. EXAMON was developed as part of the ANTAREX project (www.antarex-project.eu).

8.3.1. Installation

To compile this plugin, you need:

- A C++ 14 compiler
- Score-P Version 2+ (`SCOREP_METRIC_PLUGIN_VERSION` ≥ 1)
- CMake 3.8+
- Other dependencies will be downloaded during the cmake/make process

To build the plugin as follows:

1. Create a build directory

   ```
   mkdir build
   cd build
   ```

2. Invoke Cmake

   ```
   cmake ..
   ```

   The following settings can be customized:

   - `SCOREP_CONFIG`: Path to the `scorep-config` tool including the file name. If you have `scorep-config` in your `PATH`, it should be found by CMake.

- `CMAKE_INSTALL_PREFIX`: Directory where the resulting plugin will be installed (`lib/` suffix will be added)
- `ENABLE_MPI` (only applicable to the `examon_sync_plugin`): Enables MPI communication for the sync plugin, and allows to run more than one MPI process per node. This is mandatory for READEX!

3. Invoke make

```
make
```

4. Install the files

```
make install
```

8.3.2. Usage in READEX

For READEX purposes, we use the `examon_sync_plugin`, which enables us to attribute energy to specific regions. To use the plugin, it must be enabled during runs with PTF. To do so, use the following environment variables:

```
export SCOREP_METRIC_PLUGINS=examon_sync_plugin
export SCOREP_METRIC_PLUGINS_SEP=";"
export SCOREP_METRIC_EXAMON_SYNC_PLUGIN_BROKER=<MQTT broker address>
#
# You can use this option, if your blade counter does not use the systems hostname but
    something else;
export SCOREP_METRIC_EXAMON_SYNC_PLUGIN_EXAMON_HOST=<hostname override&>
#
# the default is org/antarex/cluster/testcluster
export SCOREP_METRIC_EXAMON_SYNC_PLUGIN_CHANNEL=<the default channel configured in examon's
    _pmu_pub.conf_key_"topic">
#
#_lower_means_more_overhead,_but_higher_granularity.
#_The_granularity_is_also_limited_from_the_EXAMON_publisher,_which_provides_the_data!
export_SCOREP_METRIC_EXAMON_SYNC_PLUGIN_INTERVAL=<Readout_delay_in_seconds_(e.g.,_0.001_for
    _1_ms)>
#
export_SCOREP_METRIC_EXAMON_SYNC_PLUGIN="EXAMON/BLADE/E"
#
#_the_whole_counter_name_used_with_MQTT_is:
#_$SCOREP_METRIC_EXAMON_SYNC_PLUGIN_EXAMON_HOST/$SCOREP_METRIC_EXAMON_SYNC_PLUGIN_CHANNEL/
    $SCOREP_METRIC_EXAMON_SYNC_PLUGIN_READEX_BLADE
export_SCOREP_METRIC_EXAMON_SYNC_PLUGIN_READEX_BLADE="<The_systems_blade_counter>INT64_s
    =0.001";
```

Furthermore, the metric source definition of your `readex_config.xml` should look like this:

```
<metricPlugin>
  <name>examon_sync_plugin</name>
</metricPlugin>
<metrics>
  <node_energy>EXAMON/BLADE/E</node_energy>
</metrics>
```

A. READEX tool suite installation on Taurus cluster (TU Dresden)

A.1. The module environment

Taurus uses so called modules to allow a dynamic environment. This avoids setting different environment variables by hand. It is therefore recommended to use these modules instead of manually setting environment variables. To use READEX related modules please do:

```
module use /projects/p_readex/modules/
```

It is assumed throughout this document that the modules directory is loaded.

A.2. Allocating a node

Please allocate a node to build the READEX Tool Suite to keep the login server free. You can do so by running:

```
srun -p haswell --time=4:00:00 --pty --exclusive -n 1 -c 24 \
    --mem=60000 -A p_readex bash -l -i
```

A.3. Score-P

To build Score-P please follow these instructions.

A.3.1. Modules

If you have previously built anything else please do:

```
module use /projects/p_readex/modules/
```

Please load the following modules to build Score-P using gcc/5.3.0 and Bull XMPI:

```
module load \
    svn/1.9.3 \
    papi/5.5.1 \
    bison/3.0.4 \
    scorep-dev/05 \
    gcc/6.3.0 \
    bullxmpi/1.2.8.4
```

If you would like to use the Intel compiler and Intel MPI instead please load:

```
module load \
    papi/5.5.1 \
    bison/3.0.4 \
    scorep-dev/05 \
    intel/2017.2.174 \
    intelmpi/2017.2.174
```

A.3.2. Download

Please Download Score-P from the following location, using your Score-P access details:

```
svn co https://silc.zih.tu-dresden.de/svn/silc-root/branches/
     TRY_READEX_online_access_call_tree_extensions
```

A.3.3. Preparing the Score-P directory

Next please do:

```
cd TRY_READEX_online_access_call_tree_extensions
./bootstrap
mkdir build
cd build
```

A.3.4. Configuring and installing Score-P

You can use your desired directory to install your Score-P version. In order to keep an overview about the different versions, please stick to the following naming scheme for the `--prefix` argument:

```
<Desired path for READEX installation>/scorep/
     TRY_READEX_online_access_call_tree_extensions_<mpi version>_<compiler verison>

<mpi version>:          for example, bullxmpi
<compiler version>:     for example, gcc6.3.0
```

To run configure please do:

```
../configure        '--prefix=<Desired_path_for_READEX_installation>/scorep/
     TRY_READEX_online_access_call_tree_extensions_<mpi_version>_<compiler_version>/' \
                                '--enable-backend-test-runs' \
                                '--with-nocross-compiler-suite=gcc' \
                                '--with-mpi=bullxmpi' \
                                '--disable-silent-rules' \
                                '--with-libbfd=no' \
                                '--with-papi-header=/sw/taurus/libraries/papi/5.5.1/include
                                    ' \
                                '--with-papi-lib=/sw/taurus/libraries/papi/5.5.1/lib' \
                                '--with-machine-name=taurus.hrsk.tu-dresden.de' \
                                '--without-gui' \
                                '--enable-static' \
                                '--enable-shared' \
                                '--enable-debug'\
                                'CFLAGS=-g -O0' \
                                'CXXFLAGS=-g -O0'\
make -j 24
make install
```

For Intel compiler and Intel MPI please change the following flags to:

```
                                '--with-nocross-compiler-suite=intel' \
                                '--with-mpi=intel3' \
```

A.3.5. Creating the related Score-P module file

Please build a module file at your desired installation path for READEX in order to use your Score-P version. Please change to the Score-P module directory:

```
cd <Desired path for READEX installation>/modules/scorep
```

Please create there a file with the following name:

```
TRY_READEX_online_access_call_tree_extensions_<mpi_version>_<compiler_version>
```

And insert the following content:

```
#%Module10######################################################################
##
## This is a modules template. Adapt it to your requirements.
## Module file for <software xyz>
##
```

```
## Source zih-modules helper script. It provides the framework for an uniform modules
     system.
## Additionally, it sets global variables that provide you information about the
## application to be loaded and some other information. See the following list:
##
## soft_arch       Provides information about the architecture.
## soft_class      Provides the software class. E.g. [applications|compiler|tools]
## soft_host       Provides the host name.
## soft_machine    Provides the machine name.
## soft_version    Provides the software version number to be loaded.
## soft_ware       Provides the software name to be loaded.
## user            Provides the user name.

source /sw/modules/global/modulescripts/zih_modules.tcl

set scorep_root "<Desired_path_for_READEX_installation>/scorep/$soft_version"

## Set modules dependencies with version information !!! e.g. intel/11.1.069

#set soft_dependencies "_"

append soft_dependencies "papi/5.5.1_<your_mpi_verison>_<your_compiler_version>"

## Set variable shortDescription if you want to add specific information that
## should be displayed while the module is loaded.

#set shortDescription "Use_\"bsub_-n<number-of-cpus>_...\"__to_start_the_application_xyz"


## Set variable longDescription if you want to add further information about the
## software that should be displayed at the module help command.

set longDescription "This_$soft_ware_provides_a_highly_scalable_measurement_infrastructure_
     to_trace_your_parallel_applications."

append longDescription "\nCompile_with_scorep_<usual_compiler_command>."


## Set the specific environment variables for your software package

#set lib_path /sw/<global|$soft_machine>/$soft_class/$soft_ware/$soft_version/$soft_arch

#environment settings for Score-P
setenv SCOREP_ROOT $scorep_root
setenv SCOREP_INC  $scorep_root/include
setenv SCOREP_LIB  $scorep_root/lib

prepend-path LD_LIBRARY_PATH $scorep_root/lib
prepend-path PATH $scorep_root/bin


## Source modules action information
source /sw/modules/global/modulescripts/zih_modules_action.tcl
```

where

```
<your mpi verison>     : for example, bullxmpi
<your compiler version>: for example, gcc/6.3.0
```

A.4. Using Score-P

Now you can use Score-P by doing:

```
module load scorep/TRY_READEX_online_access_call_tree_extensions_<mpi_version>_<
     compiler_version>
```

A.5. PTF

This section outlines how to build and use PTF.

A.5.1. Modules

If you have previously built anything else please do:

```
module purge
```

Now load the modules to build PTF with gcc:

```
module load \
    papi/5.5.1 \
    boost/1.63.0-gnu6.3 \
    automake/1.14 \
    m4/1.4.16 \
    python/2.7 \
    autoconf/2.69 \
    libtool/2.4.2 \
    autotools/2015 \
    ace/6.3.3 \
    libunwind/1.1 \
    git \
    gcc/6.3.0  \
    flex/2.5.39 \
    bison/3.0.4 \
    cereal/1.2.1

module load scorep/TRY-READEX-online-access-call-tree-extensions-<mpi version>-<compiler
     version>
```

```
<mpi version>:          for example: bullxmpi
<compiler version>:     for example: gcc6.3.0
```

For Intel please do:

```
module load \
    papi/5.5.1 \
    boost/1.63.0-gnu6.3 \
    automake/1.14 \
    m4/1.4.16 \
    python/2.7 \
    autoconf/2.69 \
    libtool/2.4.2 \
    autotools/2015 \
    ace/6.3.3 \
    libunwind/1.1 \
    git \
    intel/2017.2.174 \
    flex/2.5.39 \
    bison/3.0.4 \
    cereal/1.2.1 \

module load scorep/TRY-READEX-online-access-call-tree-extensions-<mpi version>-<compiler
     version>
```

Please be sure that the Score-P version is also compiled with the Intel compiler.

A.5.2. Download

Please download the READEX branch of PTF:

```
git -c http.sslVerify=false clone https://periscope.in.tum.de/git/Periscope.git
cd Periscope
git checkout readex
```

A.5.3. Preparing the PTF directory

Please do:

```
./bootstrap
mkdir build
cd build
```

A.5.4. Configuring and installing PTF

You can use your desired directory to install PTF. Please stick to the following naming scheme for `--prefix`:

```
<Desired path for READEX installation>/ptf/ptf-<day of build>-readex-<compiler version>-
    slurm_starter-<mpi version>-with-scorep/

<compiler version>      for example: gcc6.3.0
<mpi version>           for example: bullxmpi
```

To configure and install PTF please now do:

```
../configure        '--prefix=<Desired_path_for_READEX_installation>/ptf/ptf-<day_of_build
    >-readex-<compiler_version>-slurm_starter-<mpi_version>-with-scorep/' \
                    --enable-developer-mode \
                    --with-starter=slurm \
                    --enable-ace \
                    --enable-boost \
                    --with-boost-lib=$BOOST_LIB\
                    --enable-cube \
                    --with-cube-include=$SCOREP_INC \
                    --with-cube-lib=$SCOREP_LIB \
                    --enable-scorep \
                    --with-scorep-include=$SCOREP_INC \
                    --with-scorep-lib=$SCOREP_LIB \
                    --enable-cereal \
                    --with-cereal-include=$CEREAL_INC

make -j 24
make install
```

If you want to use the Intel compiler to compile PTF, please add the following to "../configure" :

```
    --with-compiler-suite=intel
```

Please be aware that no special Cube module is needed if you are using the same compiler for Score-P and PTF. Therefore the options for `--with-cube-include` and `--with-cube-lib` are the same as for `--with-scorep-include` and `--with-scorep-lib`.

A.5.5. Creating the related PTF module file

Please build a module file in order to use your PTF version, and make it accessible for others. Please change to the PTF module directory:

```
cd <Desired path for READEX installation>/modules/ptf
```

Please create there a file with the following name:

```
ptf-<day of build>-readex-<compiler version>-slurm_starter-<mpi version>-with-scorep
```

And insert the following content:

```
#%Module10######################################################################
##
## This is a modules template. Adapt it to your requirements.
## Module file for <software xyz>
##

## Source zih-modules helper script. It provides the framework for an uniform modules
    system.
## Additionally, it sets global variables that provide you information about the
## application to be loaded and some other information. See the following list:
##
## soft_arch       Provides information about the architecture.
## soft_class      Provides the software class. E.g. [applications|compiler|tools]
## soft_host       Provides the host name.
## soft_machine    Provides the machine name.
## soft_version    Provides the software version number to be loaded.
```

```
## soft_ware       Provides the software name to be loaded.
## user            Provides the user name.

source /sw/modules/global/modulescripts/zih_modules.tcl

set ptf_root "<Desired_path_for_READEX_installation>/ptf/$soft_version"

## Set modules dependencies with version information !!!

#set soft_dependencies "_"

append soft_dependencies "<mpi_version>_boost/1.63.0-gnu6.3_<compiler_version>_ace/6.3.3_
    libunwind/1.1"


## Set variable shortDescription if you want to add specific information that
## should be displayed while the module is loaded.

#set shortDescription "Use_\"bsub_-n<number-of-cpus>_...\"_to_start_the_application_xyz"


## Set variable longDescription if you want to add further information about the
## software that should be displayed at the module help command.

#set longDescription "This_$soft_ware_provides_a_highly_scalable_measurement_infrastructure
    _to_trace_your_parallel_applications."

#append longDescription "\nCompile_with_scorep_<usual_compiler_command>."


## Set the specific environment variables for your software package

#set lib_path /sw/<global|$soft_machine>/$soft_class/$soft_ware/$soft_version/$soft_arch

#environment settings for Score-P
setenv PTF_ROOT $ptf_root
setenv PTF_INC  $ptf_root/include
setenv PTF_LIB  $ptf_root/lib

prepend-path LD_LIBRARY_PATH $ptf_root/lib
prepend-path PATH $ptf_root/bin


## Source modules action information
source /sw/modules/global/modulescripts/zih_modules_action.tcl
```

where

```
<compiler version>      for example: gcc/5.3.0
<mpi version>           for example: bullxmpi
```

A.5.6. Using PTF

Now you can use PTF by doing:

```
module load ptf/ptf-<day of build>-readex-<compiler version>-slurm_starter-<mpi version>-
    with-scorep
```

A.6. RRL

This section outlines how to build and use RRL.

A.6.1. Modules

If you have previously built anything else please do:

```
module purge
```

Now load the modules to build the RRL:

```
module load modenv/both
module load scorep/TRY_READEX_online_access_call_tree_extensions_<mpi version>_<compiler
    version>
module load cmake/<3.11 or later>
module load papi/5.5.1
module load protobuf/3.5.0-intel2017.2.174
module load TensorFlow/1.6.0-intel-2018a-Python-3.6.4
```

where

```
<mpi version>:          for example: bullxmpi
<compiler version>:     for example: gcc6.3.0
```

A.6.2. Download

Please download the READEX RRL git:

```
git clone --recursive git@gitlab.hrz.tu-chemnitz.de:READEX/RRL.git
```

You'll need to be registered at the TU Chemnitz gitlab. If you are not, please log in into https://gitlab.hrz.tu-chemnitz.de/ and place your public ssh key there. Please have a look at https://docs.gitlab.com/ce/ssh/README.html for details about ssh keys.

A.6.3. Preparing the RRL directory

Please do:

```
cd readex-rrl/TUD_RRL/
mkdir build
cd build
```

A.6.4. Configuring and installing the RRL

You can use your desired directory to install RRL. Please stick to the following naming scheme for `-DCMAKE_INSTALL_PREFIX`:

```
<Desired path for READEX installation>/readex-rrl/rrl-<compiler version>-<mpi version>

<compiler version>      for example: gcc6.3.0
<mpi version>           for example: bullxmpi
```

Now, to build the RRL please now do:

```
export CPATH=/sw/taurus/eb/TensorFlow/1.6.0-intel-2018a-Python-3.6.4/include:$CPATH
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib:/sw/taurus/eb/TensorFlow/1.6.0-intel
    -2018a-Python-3.6.4/lib/python3.6/site-packages/tensorflow/../_solib_k8/
    _U_S_Sthird_Uparty_Smkl_Cintel_Ubinary_Ublob___Uexternal_Smkl_Slib/

cmake ../ -DCMAKE_INSTALL_PREFIX=<Desired path for READEX installation>/readex-rrl/rrl-<
    compiler version>-<mpi version> -DEXTERN_TENSORFLOW=ON -DEXTERN_PROTOBUF=ON
make -j24
make install
```

A.6.5. Creating the related RRL module file

Please build a module file in order to use your RRL version, and make it accessible for others. Please change to the RRL module directory:

```
cd <Desired path for READEX installation>/modules/readex-rrl
```

Please create there a file with the following name:

```
rrl-<compiler version>-<mpi version>
```

And insert the following content:

```
#%Module10######################################################################
##
## This is a modules template. Adapt it to your requirements.
## Module file for <software xyz>
##

## Source zih-modules helper script. It provides the framework for an uniform modules
   system.
## Additionally, it sets global variables that provide you information about the
## application to be loaded and some other information. See the following list:
##
## soft_arch       Provides information about the architecture.
## soft_class      Provides the software class. E.g. [applications|compiler|tools]
## soft_host       Provides the host name.
## soft_machine    Provides the machine name.
## soft_version    Provides the software version number to be loaded.
## soft_ware       Provides the software name to be loaded.
## user            Provides the user name.

source /sw/modules/global/modulescripts/zih_modules.tcl

set rrl_root "<Desired_path_for_READEX_installation>/readex-rrl/$soft_version"

## Set modules dependencies with version information !!!

#set soft_dependencies

append soft_dependencies "<your_mpi_verison>_<your_compiler_version>"

## Set variable shortDescription if you want to add specific information that
## should be displayed while the module is loaded.

#set shortDescription "Use_\"bsub_-n<number-of-cpus>_...\"__to_start_the_application_xyz"


## Set variable longDescription if you want to add further information about the
## software that should be displayed at the module help command.

set longDescription "TODO"

append longDescription "\nTODO"


## Set the specific environment variables for your software package

#set lib_path /sw/<global|$soft_machine>/$soft_class/$soft_ware/$soft_version/$soft_arch

#environment settings for Score-P
setenv RRL_ROOT $rrl_root
setenv RRL_INC  $rrl_root/include
setenv RRL_LIB  $rrl_root/lib

prepend-path LD_LIBRARY_PATH $rrl_root/lib
prepend-path PATH $rrl_root/bin
prepend-path CPAHT $rrl_root/bin/include


## Source modules action information
source /sw/modules/global/modulescripts/zih_modules_action.tcl
```

where

```
  <mpi version>                 : mpi version of the scorep version you used
  <compiler version>            : compiler of the scorep version you used
```

A.6.6. Using RRL

Now you can use RRL by doing:

```
module load readex-rrl/rrl-<compiler version>-<mpi version>
export SCOREP_SUBSTRATE_PLUGINS='rrl'
export SCOREP_RRL_VERBOSE="DEBUG"

export SCOREP_TUNING_PLUGINS='OpenMPTP,...'
<or>
export SCOREP_RRL_PLUGINS='OpenMPTP,...'

#optional, if a tuning model is present
#export SCOREP_RRL_TMM_PATH="/path/to/the/tuning/model.json"
```

A.6.7. Building doc

Please go to your build folder and do:

```
module load doxygen/1.8.11
make doc
```

A.7. PCP's

This section outlines how to build the different Parameter Control Plugins.

A.7.1. Modules

If you have previously built anything else please do:

```
module purge
```

Now load the modules to build the RRL:

```
module load readex-rrl/rrl-<compiler version>-<mpi version>
```

Where is:

```
<mpi version>:          for example: bullxmpi
<compiler version>:     for example: gcc6.3.0
```

A.7.2. Download

Please download the PCP git:

```
git clone git@gitlab.hrz.tu-chemnitz.de:READEX/PCPs.git
```

You'll need to be registered at the TU Chemnitz gitlab. If you are not, please log in into https://gitlab.hrz.tu-chemnitz.de/ and place your public ssh key there. Please have a look at https://docs.gitlab.com/ce/ssh/README.html for details about ssh keys.

A.7.3. Configuring and installing the PCP's

You can use your desired directory to install the PCP's. Please stick to the following naming scheme for the second argument of the build script:

```
<Desired path for READEX installation>/parameter_control_plugins/pcp-<compiler version>

<compiler version>      for example: gcc6.3.0
```

To build the PCP's please now do:

```
export RRL_INC=<Path to RRL include directory>
cd PCPs
./build.sh <Desired path for READEX installation>/parameter_control_plugins/pcp-<compiler
    version>
```

A.7.4. Creating the related PCP's module file

Please build a module file in order to use your PCP version, and make it accessible for others. Please change to the PCP module directory:

```
cd <Desired path for READEX installation>/modules/pcp/
```

Please create there a file with the following name:

```
pcp-<compiler version>
```

And insert the following content:

```
#%Module10######################################################################
##
## This is a modules template. Adapt it to your requirements.
## Module file for <software xyz>
##

## Source zih-modules helper script. It provides the framework for an uniform modules
    system.
## Additionally, it sets global variables that provide you information about the
## application to be loaded and some other information. See the following list:
##
## soft_arch       Provides information about the architecture.
## soft_class      Provides the software class. E.g. [applications|compiler|tools]
## soft_host       Provides the host name.
## soft_machine    Provides the machine name.
## soft_version    Provides the software version number to be loaded.
## soft_ware       Provides the software name to be loaded.
## user            Provides the user name.

source /sw/modules/global/modulescripts/zih_modules.tcl

set pcp_root "<Desired_path_for_READEX_installation>/parameter_control_plugins/pcp-<
    compiler_version>"

## Set modules dependencies with version information !!!

#set soft_dependencies ""

append soft_dependencies "<your_mpi_verison>_<your_compiler_version>"

## Set variable shortDescription if you want to add specific information that
## should be displayed while the module is loaded.

#set shortDescription "Use_\"bsub_-n<number-of-cpus>_...\"__to_start_the_application_xyz"


## Set variable longDescription if you want to add further information about the
## software that should be displayed at the module help command.

set longDescription "TODO"

append longDescription "\nTODO"


## Set the specific environment variables for your software package
```

```
#set lib_path /sw/<global|$soft_machine>/$soft_class/$soft_ware/$soft_version/$soft_arch

#environment settings for Score-P
setenv PCP_ROOT $pcp_root
setenv PCP_LIB  $pcp_root/lib

prepend-path LD_LIBRARY_PATH $pcp_root/lib

## Source modules action information
source /sw/modules/global/modulescripts/zih_modules_action.tcl
```

where

```
<mpi version>:          for example: bullxmpi
<compiler version>:     for example: gcc6.3.0
```

A.7.5. Using PCP's

Now you can use the PCP's by adding them to `SCOREP_RRL_PLUGINS`:

```
module load pcp/pcp-<compiler version>
export SCOREP_RRL_PLUGINS='OpenMPTP,cpu_freq_plugin,epb_plugin,uncore_freq_plugin,...'
```

A.8. ATP

This section outlines how to build and use the ATP library.

A.8.1. Modules

If you have previously built anything else please do:

```
module purge
```

Now load the modules to build the ATP library:

```
module load readex-rrl/rrl-<compiler version>-<mpi version>
module load cmake/<3.11 or later>
```

```
<mpi version>:          for example: bullxmpi
<compiler version>:     for example: gcc6.3.0
```

A.8.2. Download

Please download the ATP git:

```
git clone git@gitlab.hrz.tu-chemnitz.de:READEX/ATP.git
```

You'll need to be registered at the TU Chemnitz gitlab. If you are not, please log in into https://gitlab.hrz.tu-chemnitz.de/ and place your public ssh key there. Please have a look at https://docs.gitlab.com/ce/ssh/README.html for details about ssh keys.

A.8.3. Preparing the ATP directory

Please do:

```
cd ATP/
mkdir build
cd build
```

A.8.4. Configuring and installing the ATP

You can use your desired directory to install ATP. Please stick to the following naming scheme for `-DCMAKE_INSTALL_PREFIX`:

```
<Desired path for READEX installation>/readex-atp/atp-<compiler version>-<mpi version>

<compiler version>      for example: gcc6.3.0
<mpi version>           for example: bullxmpi
```

To build the ATP please now do:

```
cmake ../ -DCMAKE_INSTALL_PREFIX=<Desired path for READEX installation>/readex-atp/atp-<
    compiler version>-<mpi version>
make -j24
make install
```

A.8.5. Creating the related ATP module file

Please build a module file in order to use your ATP version, and make it accessible for others. Please change to the ATP module directory:

```
cd <Desired path for READEX installation>/modules/readex-atp
```

Please create there a file with the following name:

```
atp-<compiler version>-<mpi version>
```

And insert the following content:

```
#%Module10######################################################################
##
## This is a modules template. Adapt it to your requirements.
## Module file for <software xyz>
##

## Source zih-modules helper script. It provides the framework for an uniform modules
    system.
## Additionally, it sets global variables that provide you information about the
## application to be loaded and some other information. See the following list:
##
## soft_arch       Provides information about the architecture.
## soft_class      Provides the software class. E.g. [applications|compiler|tools]
## soft_host       Provides the host name.
## soft_machine    Provides the machine name.
## soft_version    Provides the software version number to be loaded.
## soft_ware       Provides the software name to be loaded.
## user            Provides the user name.

source /sw/modules/global/modulescripts/zih_modules.tcl

set atp_root "<Desired_path_for_READEX_installation>/readex-atp/$soft_version"

## Set modules dependencies with version information !!! e.g. intel/11.1.069

#set soft_dependencies

append soft_dependencies "readex-rrl/rrl-gcc6.3.0-bullxmpi"

## Set variable shortDescription if you want to add specific information that
## should be displayed while the module is loaded.

#set shortDescription "Use_\"bsub_-n<number-of-cpus>_...\"__to_start_the_application_xyz"


## Set variable longDescription if you want to add further information about the
## software that should be displayed at the module help command.

set longDescription "TODO"

append longDescription "\nTODO"


## Set the specific environment variables for your software package
```

```
#set lib_path /sw/<global|$soft_machine>/$soft_class/$soft_ware/$soft_version/$soft_arch

#environment settings for ATP
setenv ATP_PATH $atp_root/bin

prepend-path LD_LIBRARY_PATH $atp_root/lib
prepend-path PATH $atp_root/bin
prepend-path CPATH $atp_root/include


## Source modules action information
source /sw/modules/global/modulescripts/zih_modules_action.tcl
```

```
  <mpi version>                 : mpi version of the scorep version you used
  <compiler version>            : compiler of the scorep version you used
```

A.8.6. Using ATP

Now you can use ATP by doing:

```
module load readex-atp/atp-<compiler version>-<mpi version>
```

A.9. Cluster Prediction

This section outlines how to build and use the Cluster Prediction library.

A.9.1. Modules

If you have previously built anything else please do:

```
module purge
```

Now load the modules to build the ATP library:

```
module load readex-rrl/rrl-<compiler version>-<mpi version>
module load cmake/<3.11 or later>
module load papi/<5.5.1 or later>
```

```
<mpi version>:      for example: bullxmpi
<compiler version>:     for example: gcc6.3.0
```

A.9.2. Download

Please download the Cluster Prediction git:

```
git clone git@gitlab.hrz.tu-chemnitz.de:READEX/Cluster_Prediction.git
```

You'll need to be registered at the TU Chemnitz gitlab. If you are not, please log in into https://gitlab.hrz.tu-chemnitz.de/ and place your public ssh key there. Please have a look at https://docs.gitlab.com/ce/ssh/README.html for details about ssh keys.

A.9.3. Preparing the Cluster Prediction directory

Please do:

```
cd Cluster_Prediction/
mkdir build
cd build
```

A.9.4. Configuring and installing the Cluster Prediction library

You can use your desired directory to install Cluster Prediction. Please stick to the following naming scheme for `-DCMAKE_INSTALL_PREFIX`:

```
<Desired path for READEX installation>/cluster_prediction/cluster_prediction-<compiler
    version>-<mpi version>

<compiler version> for example: gcc6.3.0
<mpi version>      for example: bullxmpi
```

To build the Cluster Prediction library please now do:

```
cmake ../ -DCMAKE_INSTALL_PREFIX=<Desired path for READEX installation>/cluster_prediction/
    cluster_prediction-<compiler version>-<mpi version> -DCMAKE_PREFIX_PATH=<papi
    installation root folder>
make
make install
```

A.9.5. Creating the related Cluster Prediction module file

Please build a module file in order to use your Cluster Prediction version, and make it accessible for others. Please change to the Cluster Prediction module directory:

```
cd <Desired path for READEX installation>/modules/cluster_prediction
```

Please create there a file with the following name:

```
cluster_prediction-<compiler version>-<mpi version>
```

And insert the following content:

```
#%Module10######################################################################
##
## This is a modules template. Adapt it to your requirements.
## Module file for <software xyz>
##

## Source zih-modules helper script. It provides the framework for an uniform modules
    system.
## Additionally, it sets global variables that provide you information about the
## application to be loaded and some other information. See the following list:
##
## soft_arch       Provides information about the architecture.
## soft_class      Provides the software class. E.g. [applications|compiler|tools]
## soft_host       Provides the host name.
## soft_machine    Provides the machine name.
## soft_version    Provides the software version number to be loaded.
## soft_ware       Provides the software name to be loaded.
## user            Provides the user name.

source /sw/modules/global/modulescripts/zih_modules.tcl

set cluster_prediction_root "<Desired_path_for_READEX_installation>/cluster_prediction/
    $soft_version"

## Set modules dependencies with version information !!! e.g. intel/11.1.069

#set soft_dependencies

append soft_dependencies "readex-rrl/rrl-gcc6.3.0-bullxmpi"

## Set variable shortDescription if you want to add specific information that
## should be displayed while the module is loaded.

#set shortDescription "Use \"bsub -n<number-of-cpus> ...\" to start the application xyz"


## Set variable longDescription if you want to add further information about the
## software that should be displayed at the module help command.

set longDescription "TODO"

append longDescription "\nTODO"
```

```
## Set the specific environment variables for your software package

#set lib_path /sw/<global|$soft_machine>/$soft_class/$soft_ware/$soft_version/$soft_arch

#environment settings for CLUSTER_PREDICTION
setenv CLUSTER_PREDICTION_LIB $cluster_prediction_root/lib
setenv CLUSTER_PREDICTION_INC $cluster_prediction_root/include
setenv CLUSTER_PREDICTION_PATH $cluster_prediction_root


prepend-path LD_LIBRARY_PATH $cluster_prediction_root/lib


## Source modules action information
source /sw/modules/global/modulescripts/zih_modules_action.tcl
```

```
  <mpi version>          : mpi version of the scorep version you used
  <compiler version>        : compiler of the scorep version you used
```

A.9.6. Using Cluster Prediction

Now you can use Cluster Prediction by doing:

```
module load cluster_prediction/cluster_prediction-<compiler version>-<mpi version>
```

B. READEX Docker Image

A docker file to create a docker image with the READEX tool suite installed on an Ubuntu OS is available at https://github.com/readex-eu/readex-docker.

The steps to create a docker image using the provided docker file are as follows:

1. Install docker.
2. Download the READEX docker file (Dockerfile.readex_gcc7.3.0).
3. Build the READEX docker image from the dockerfile in the current directory using the following command:

```
docker build -t="docker/readex_gcc7.3.0" -f Dockerfile.readex_gcc7.3.0 .
```

To run the docker image, execute the following command:

```
docker run -it --rm docker/readex_gcc7.3.0
```

Note: All environment variables that are required to use the individual tools in the READEX tool suite are set in the docker image.

C. READEX Integrated Installation Script

A script for downloading and installing the software components of the READEX Tool Suite is available at https://github.com/readex-eu/readex-install-script.

Before running the installation script, edit it and provide the right paths to the required software, if necessary.

C.1. Requirements

Please make sure you have installed the required software on your system:

- GCC (G++ and GFortran) 6.3.0/7.1.0 or Intel compiler 2017.2.174/2018.1.163
- Bison 3.0.4
- PAPI 5.5.1 or higher
- Python 3.6 or higher
- Ace 6.5.0
- Flex 2.5.39
- Boost 1.65.0
- Cereal 1.2.1
- CMake 3.11 or higher
- Lua 5.1 or higher

If the installation script (`readex-install.sh`) doesn't find right versions of the required software, the script will install it in the installation path or if the right version is in the package repository, it will be installed with `sudo apt-get install` ⟨`package`⟩.

C.2. Usage

To run the installation script, execute the following command:

```
./readex-install.sh <installation path>
```

If no installation path is specified, it will be installed in the `/opt` directory.
After checking the required software you can continue or abort the installation.

```
Do you want to continue with the Installation of READEX Tool Suite? (yes|no)
```

The installation of `x86_adapt` is optional, so you can decide if it should be installed.

```
Do you want to install x86_adapt? (yes|no)
```

After the installation, you can install Environment Modules to get easy access to the READEX Tool Suite.

```
Do you want to use Environment Modules to get easy access to the READEX Tool Suite? (yes|no
    )
```

In case, you want to use Modules, you can decide if it should be installed or if you have already installed and configured it.

```
Do you have Environment Modules already installed and configured? (yes|no)
```

If you have a functional version, a modulefile is provided, which you can place in your `modulefilesdir`. In the other case, the script will automatically install Environment Modules in the given installation path and configure it. So you can use the READEX Tool Suite with `module load Readex\⟨mpi_version⟩_⟨compiler⟩`.

If you don't want to use Modules at all, you will get commands to append your `$PATH` and `$LD_LIBRARY_PATH` variable, so you can use the READEX Tool Suite.

After the script finished successfully, you can clean your work directory or keep the archive files and directories.

```
Do you want to clean the current directory? (yes|no)
```