2nd Workshop on Power-Aware Computing 2017

Domain Knowledge Specification for Energy Tuning

Speaker: Anamika Chowdhury

Authors:
TUM: Anamika Chowdhury, Madhura Kumaraswamy, Michael Gerndt,
Intel ExaScale Labs, Paris, France: Zakaria Bendifallah, Othman Bouizi
IT4Innovations National Supercomputing Centre: Lubomir Riha, Ondrej Vysocky, Martin Beseda, Jan Zapletal

READEX
Runtime Exploitation of Application Dynamism for Energy-efficient eXascale computing

European Commission
Horizon 2020
European Union funding for Research & Innovation

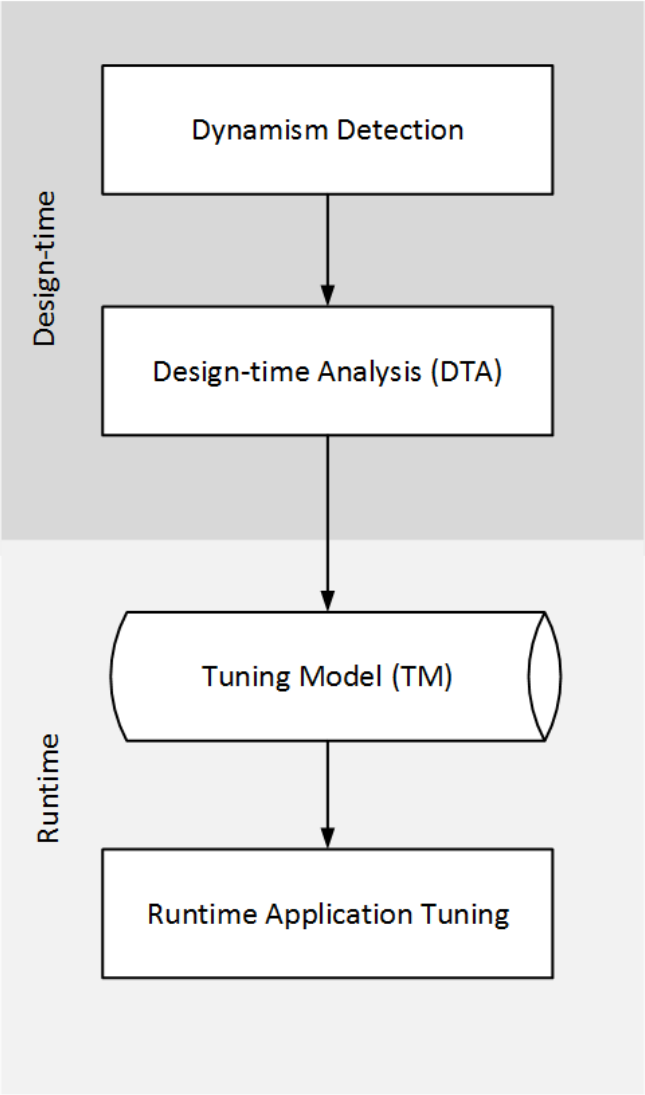
Project Overview

- READEX:
  Runtime **E**xploitation of **A**pplication **D**ynamism for **E**nergy-efficient e**X**ascale Computing
- Starting Date:
  1 September 2015
- Duration:
  3 years
- Funding
  European Commission Horizon 2020 grant agreement 671657
- Collaboration with 6 other institutions all over Europe

READEX Objectives

- Tuning HPC applications dynamically for energy efficiency.
- Improve energy efficiency by influencing tuning parameters
- Switching between configurations
  - Exploit dynamic characteristics
- Develop tool aided auto-tuning methodology.
  - Design-time Analysis
  - Runtime Application Tuning
- Detect at design-time, exploit at runtime.
- Compute better configuration
  - Specify domain knowledge by application owners
    - *Identifiers*
  - Application Tuning parameter

The READEX Tool Suite



Terminology: Phase Region and Phase

```c
int main(void) {

    // Initialize application
    // Initialize experiment variables

    int num_iterations = 2;
    for (int iter = 1; iter <= num_iterations; iter++) {
       // Start phase region
       // Read PhaseCharct
       laplace3D();          // significant region
       residue = reduction(); // insignificant region
       fftw_execute();       // significant region
       // End phase region
    }

    // Post-processing:
    // Write noise matrices to disk for visualization
    // Terminate application

    MPI_Finalize();
    return 0;
}
```

Phase region

Phase

Scenario

FREQ=2 GHz

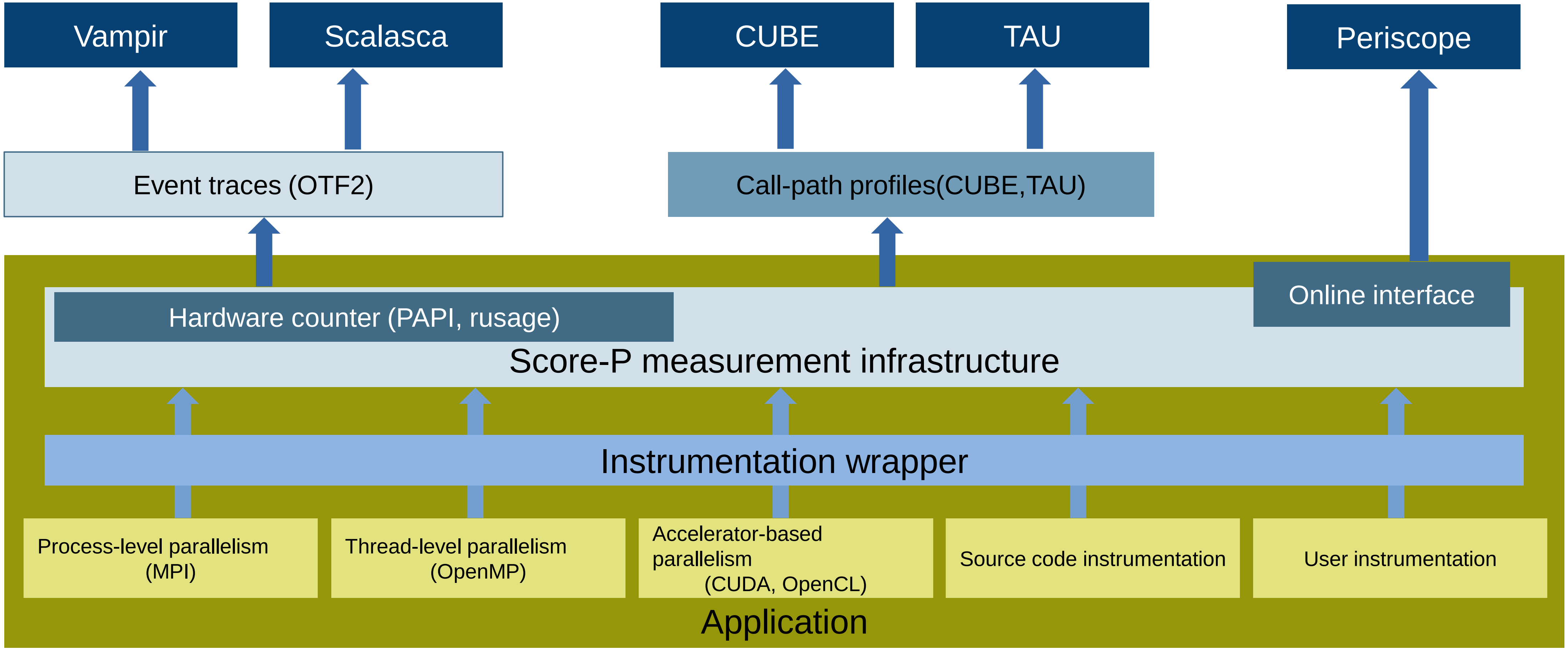
FREQ=1.5 GHz

Significant region

Runtime situation

Score-P

- Scalable Performance Measurement Infrastructure for Parallel Codes
  - Common instrumentation and measurement infrastructure



European Commission | Horizon 2020 European Union funding for Research & Innovation

Periscope Tuning Framework (PTF)

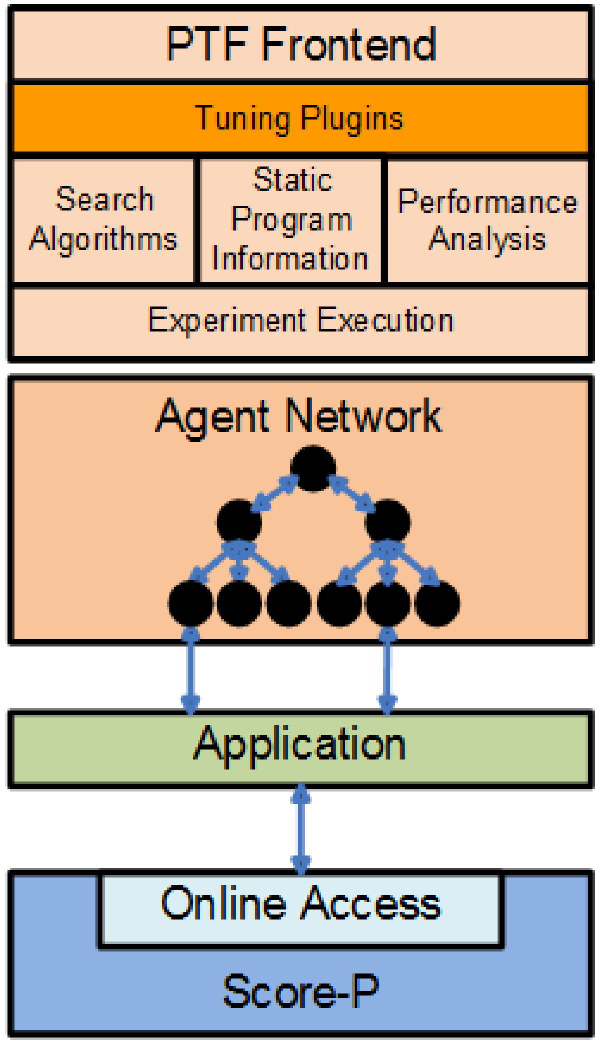
Automatic application analysis & tuning

- Tune performance and energy (statically)
- Plug-in-based architecture
- Evaluate alternatives online
- Scalable and distributed framework

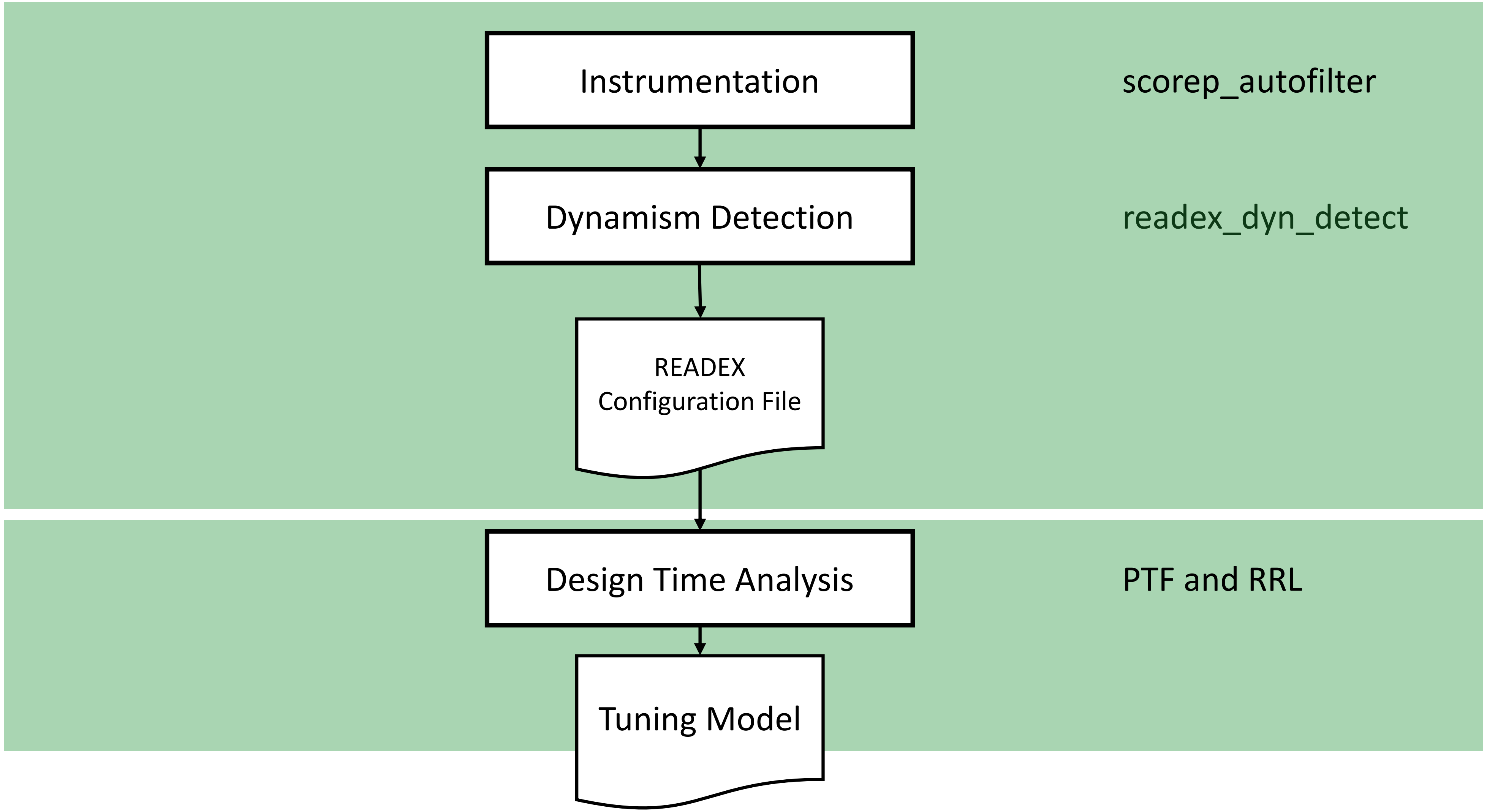
Support variety of parallel paradigms

- MPI, OpenMP, OpenCL, Parallel pattern

Developed in the AutoTune EU-FP7 project



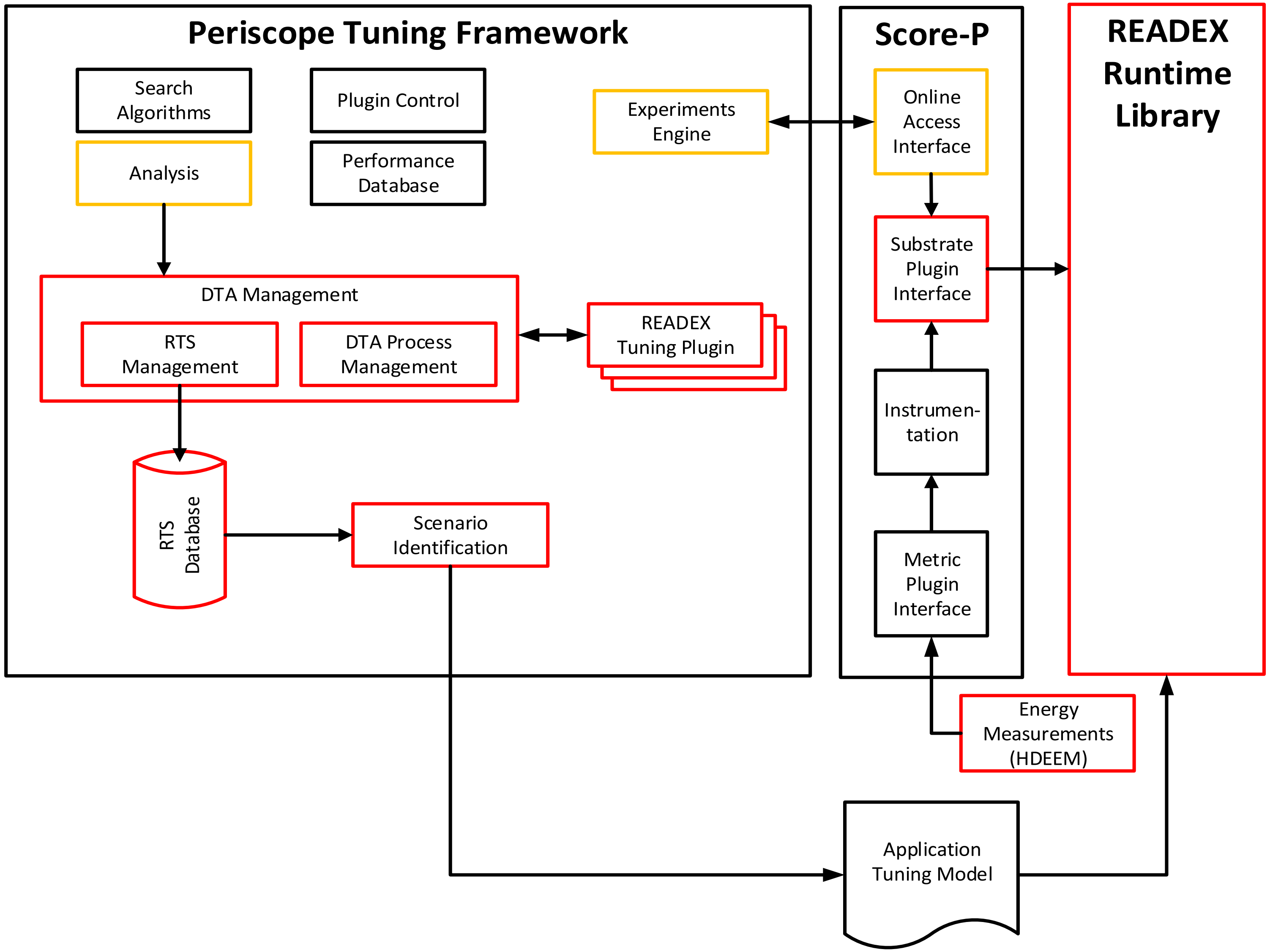
Design Time Analysis



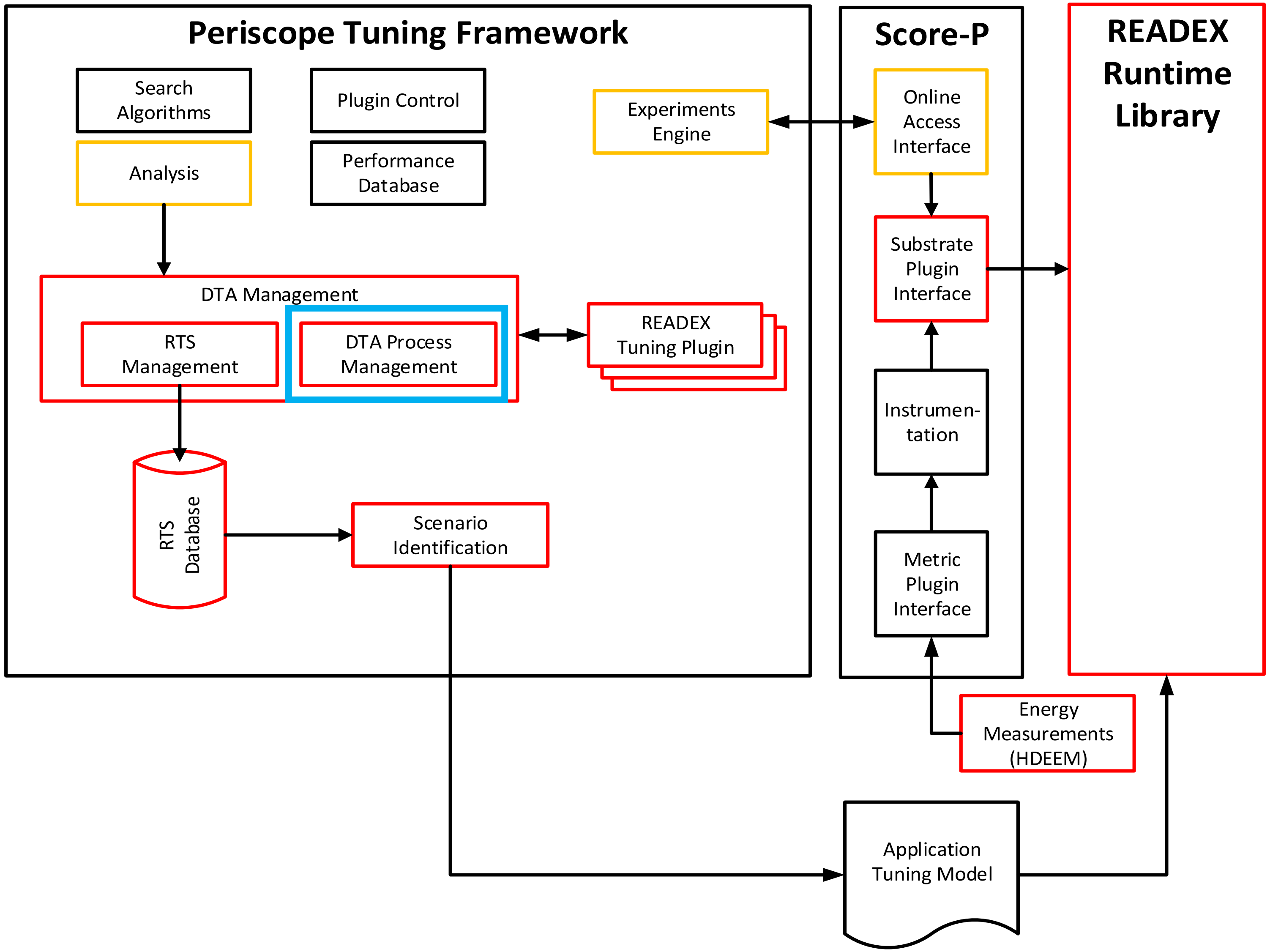
READEX Tuning Plugin

- Tuning plugin supporting
  - Core and uncore frequencies, numthreads parameters
  - Configurable search space via READEX Configuration File
  - Several objective functions: energy, CPUenergy, EDP, EDP2, time
  - Several search strategies: exhaustive, individual, random, genetic
- Approach
  - Experiment with default configuration
  - Experiments for selected configurations
  - Configuration set for phase region
  - Energy and time measured for all runtime situations
  - Identification of static best for phase and specific best configurations for rts‘s

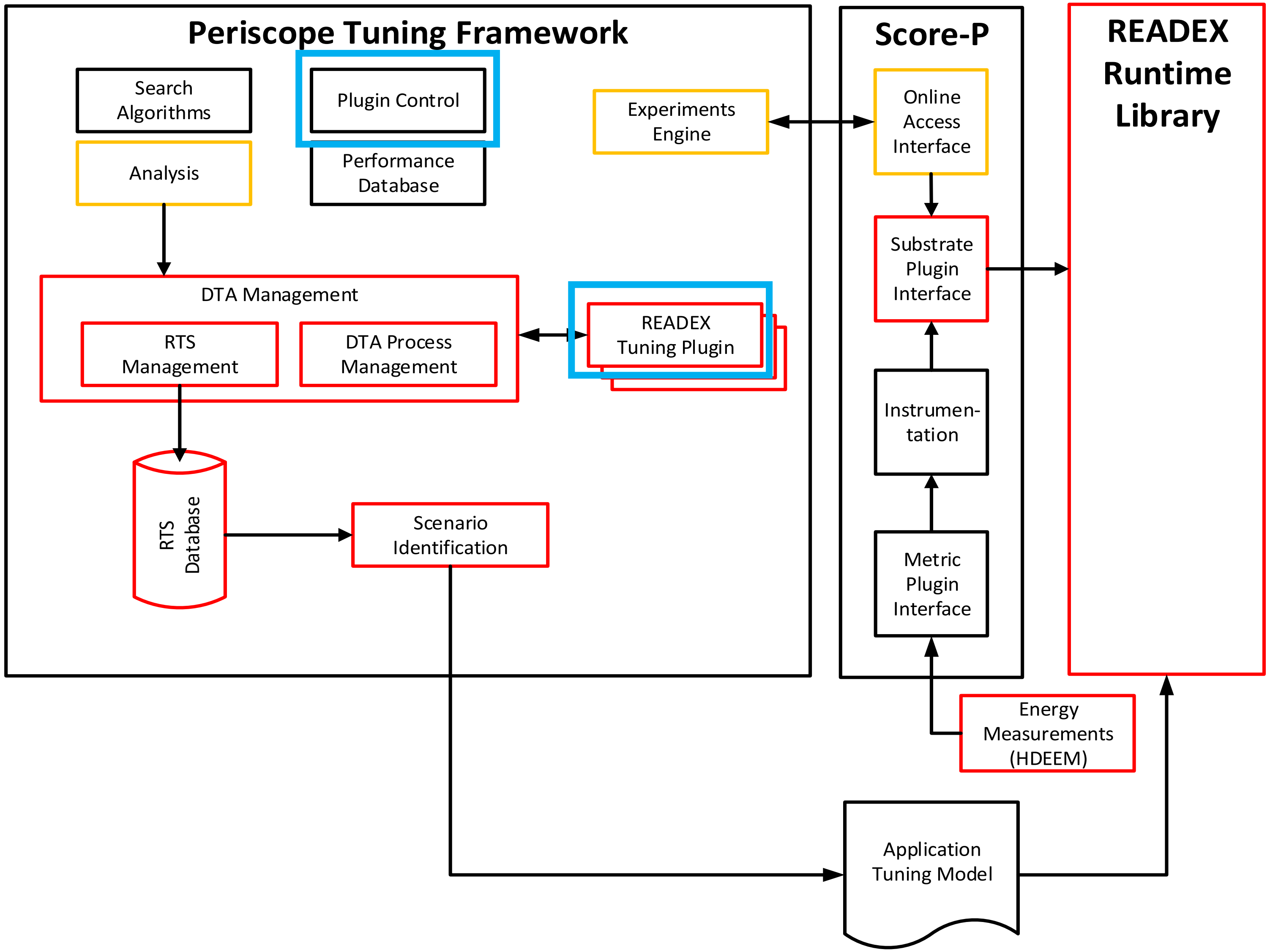
Pre-Computation of Configurations



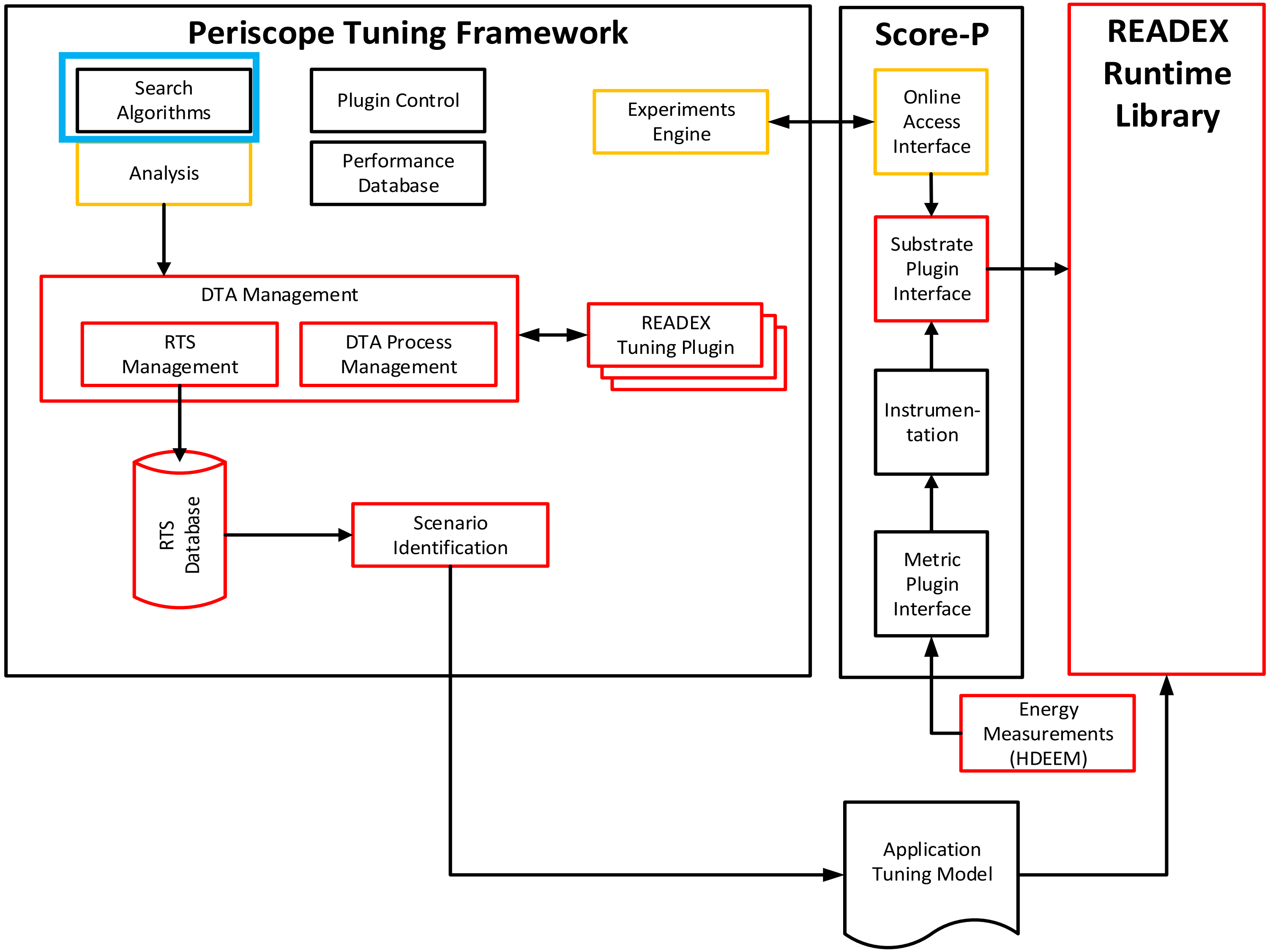
Pre-Computation of Configurations



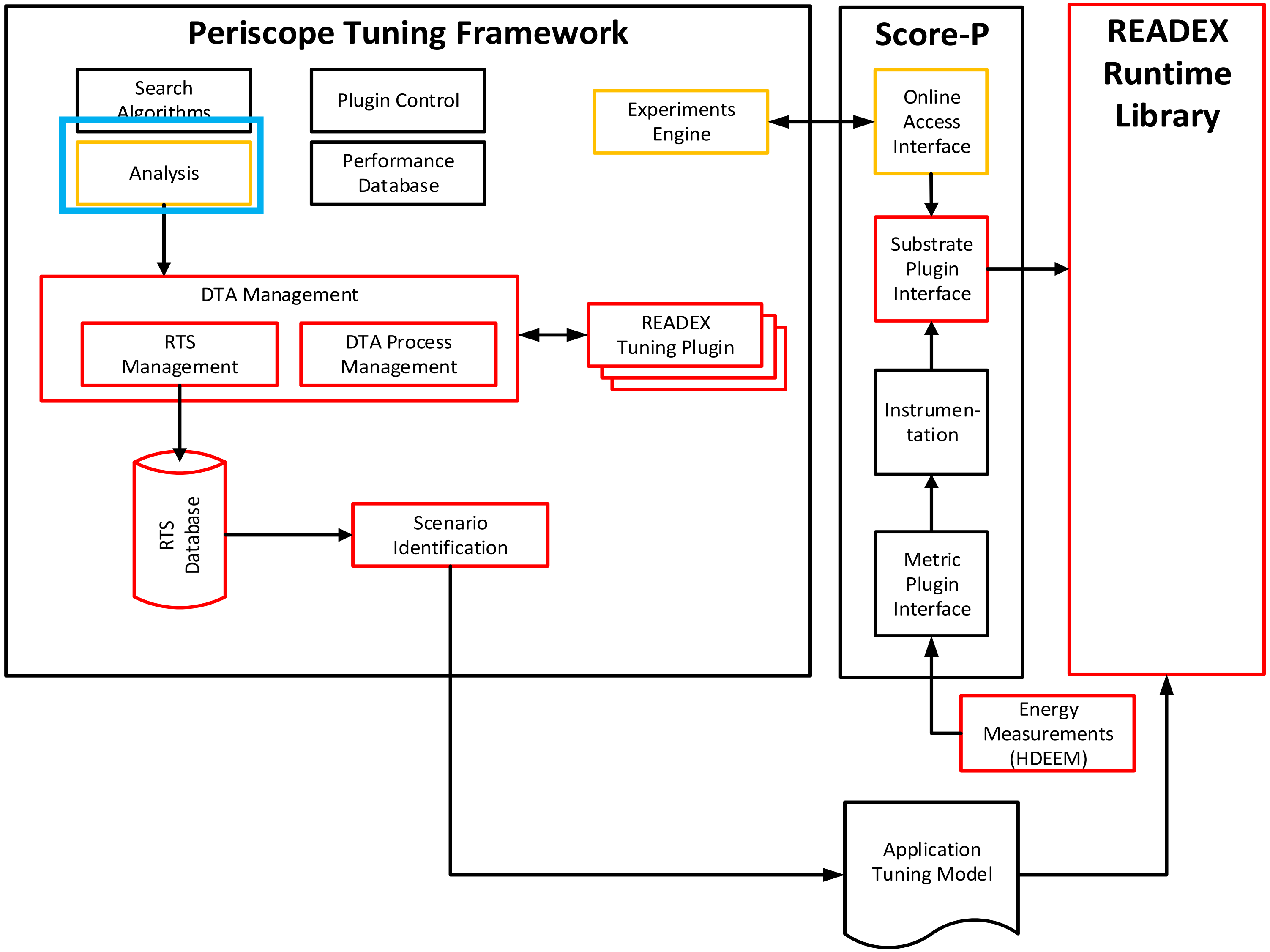
Pre-Computation of Configurations



Pre-Computation of Configurations



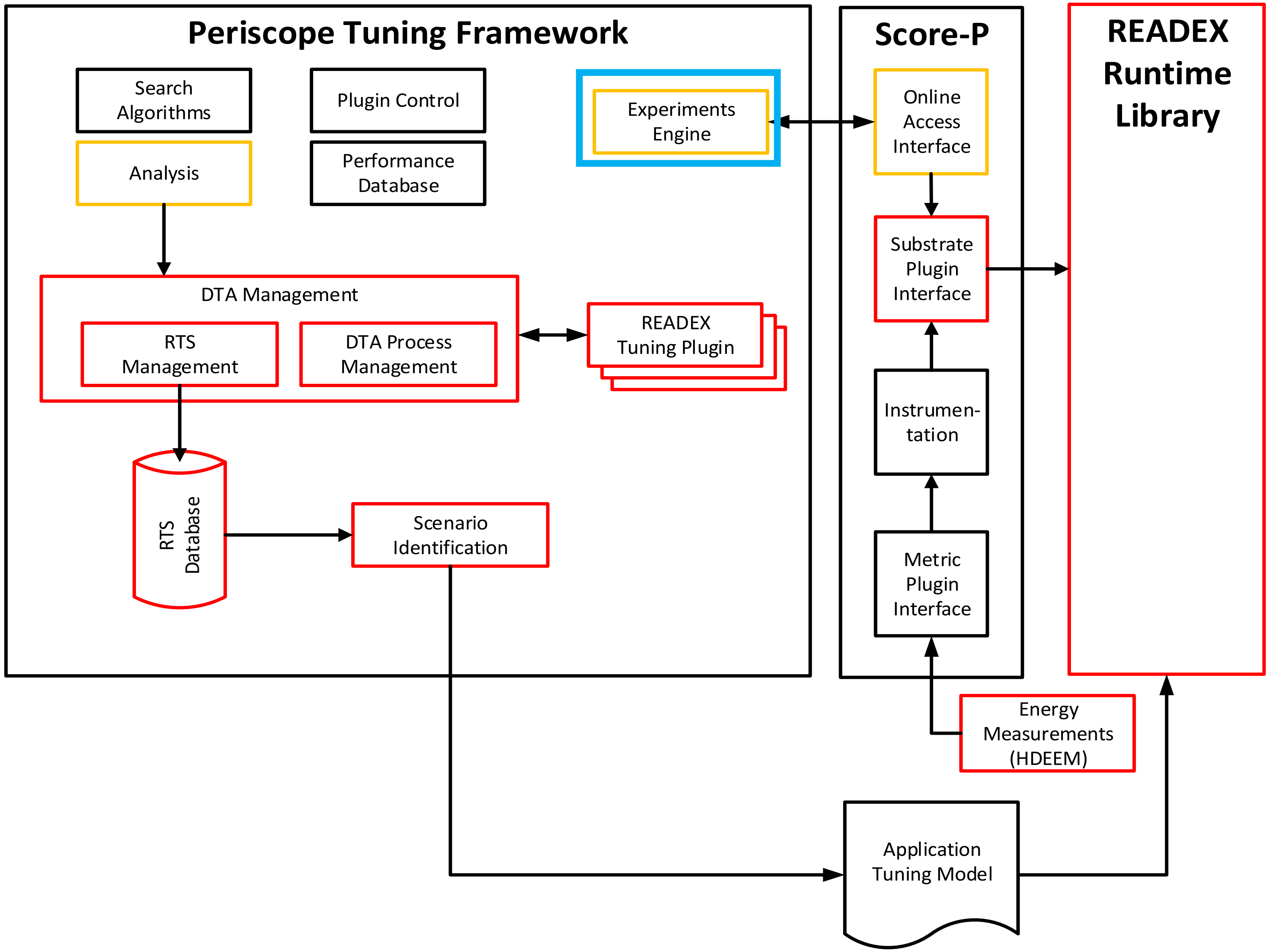
Pre-Computation of Configurations



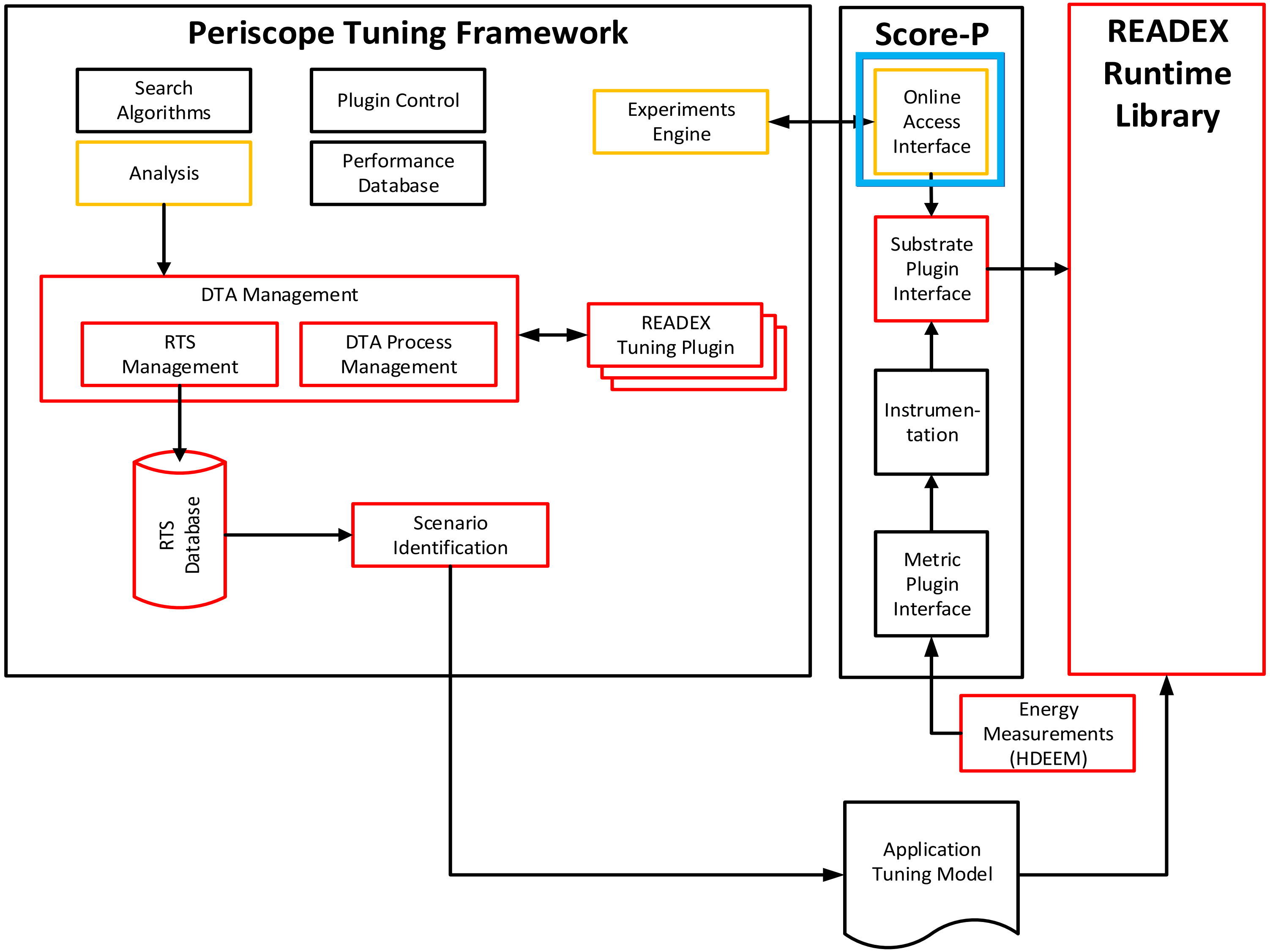
READEX
Runtime Exploitation of Application Dynamism for Energy-efficient eXascale computing

European Commission
Horizon 2020
European Union funding for Research & Innovation

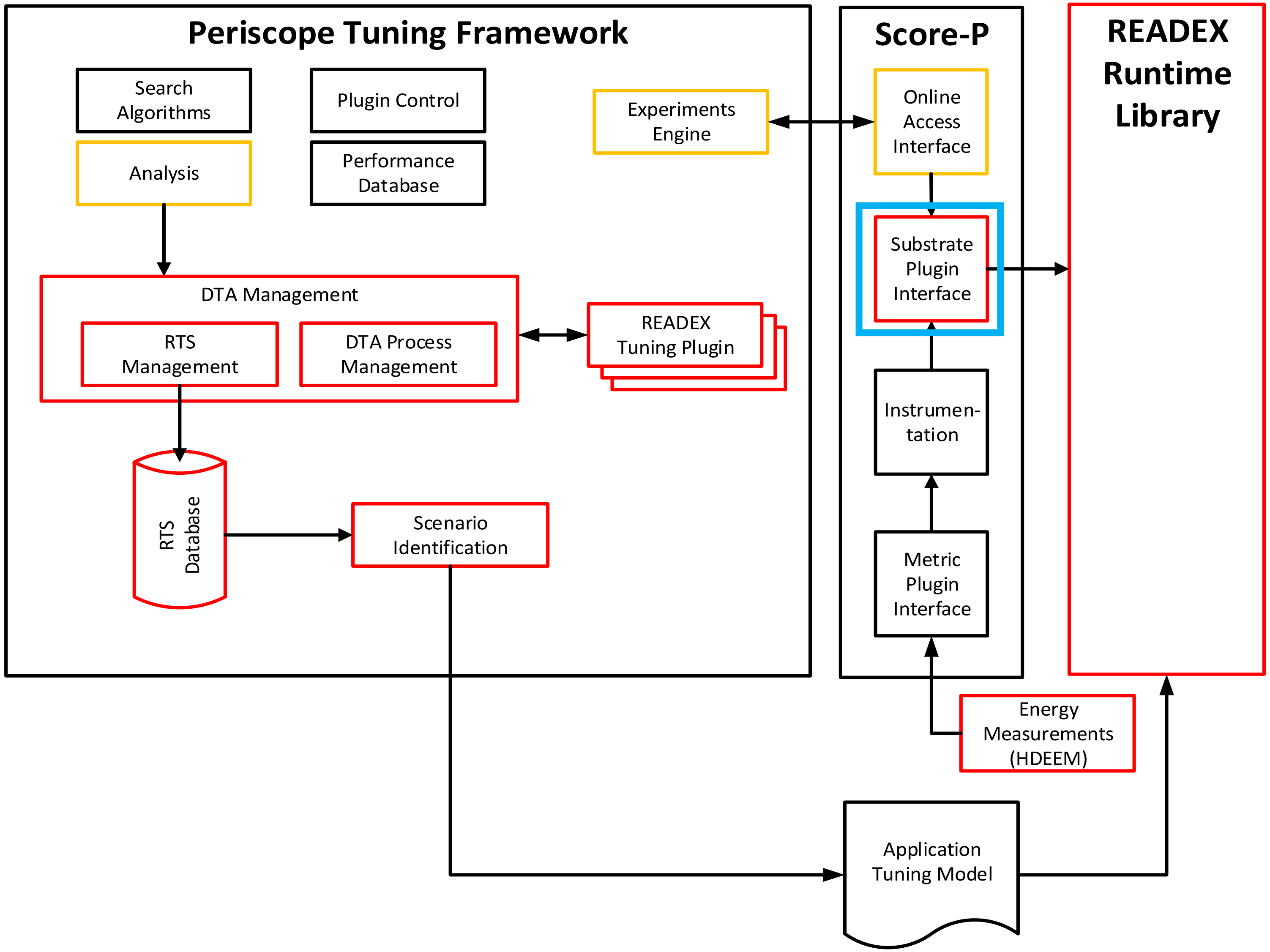
Pre-Computation of Configurations



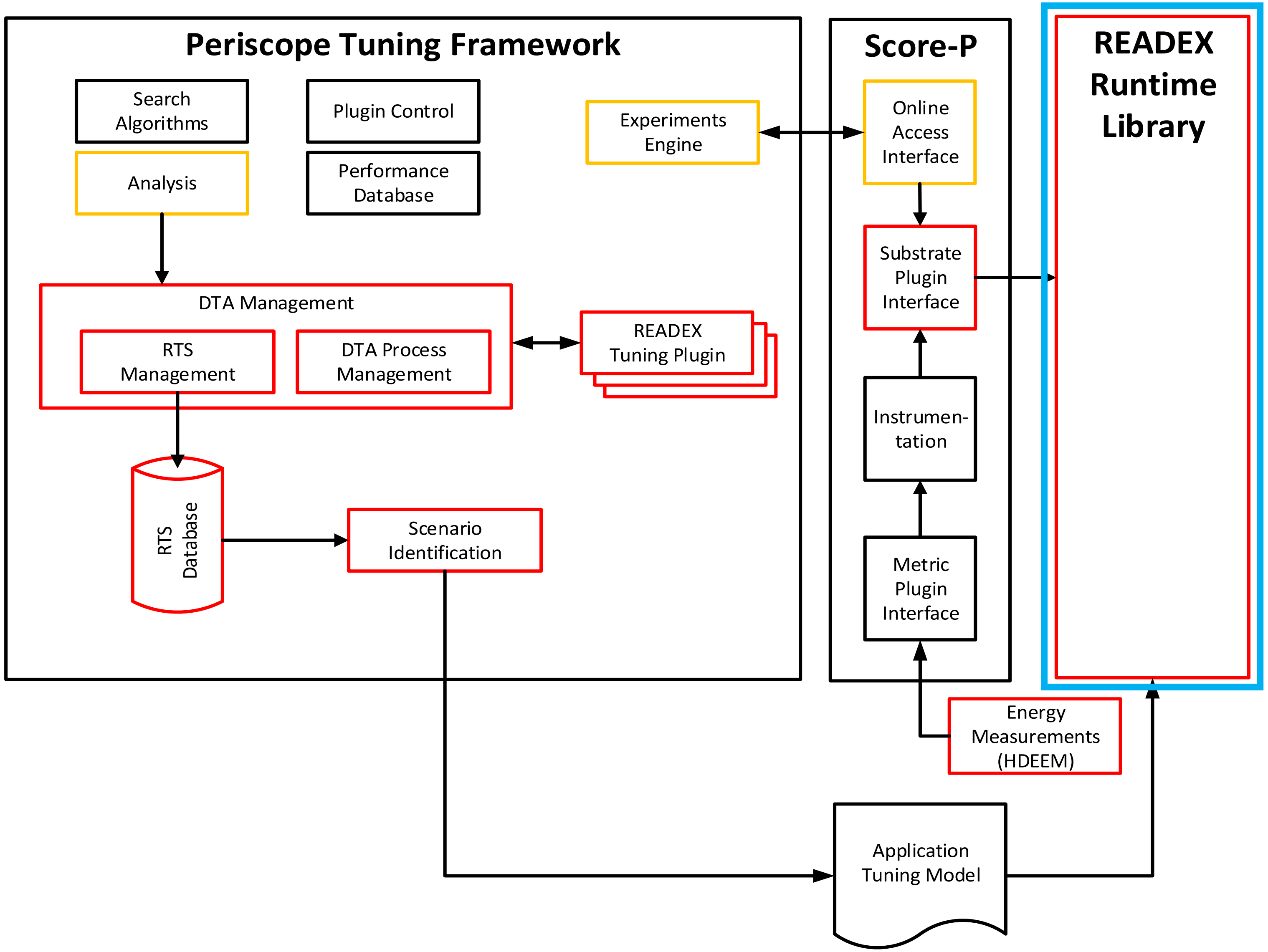
Pre-Computation of Configurations



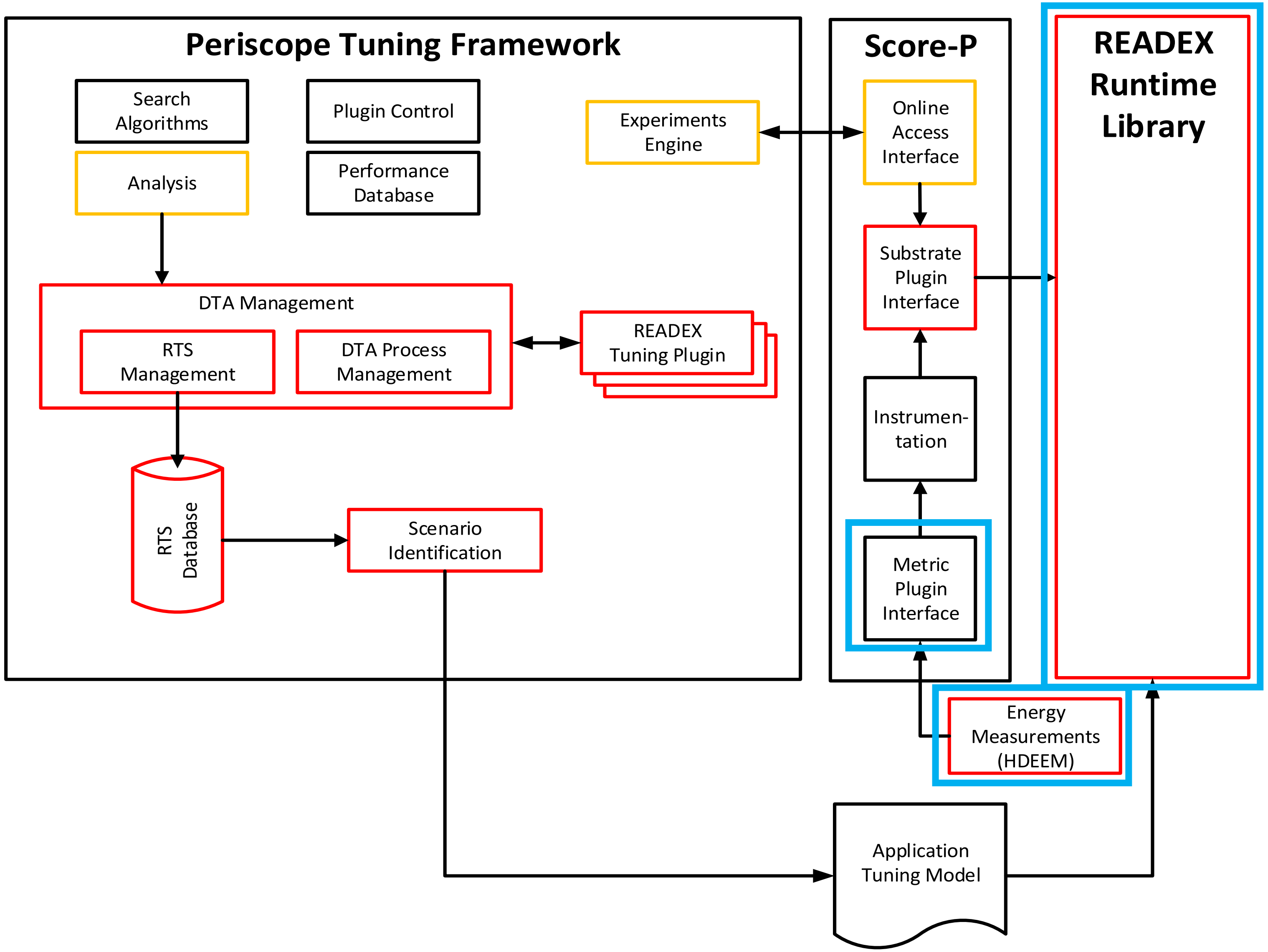
Pre-Computation of Configurations



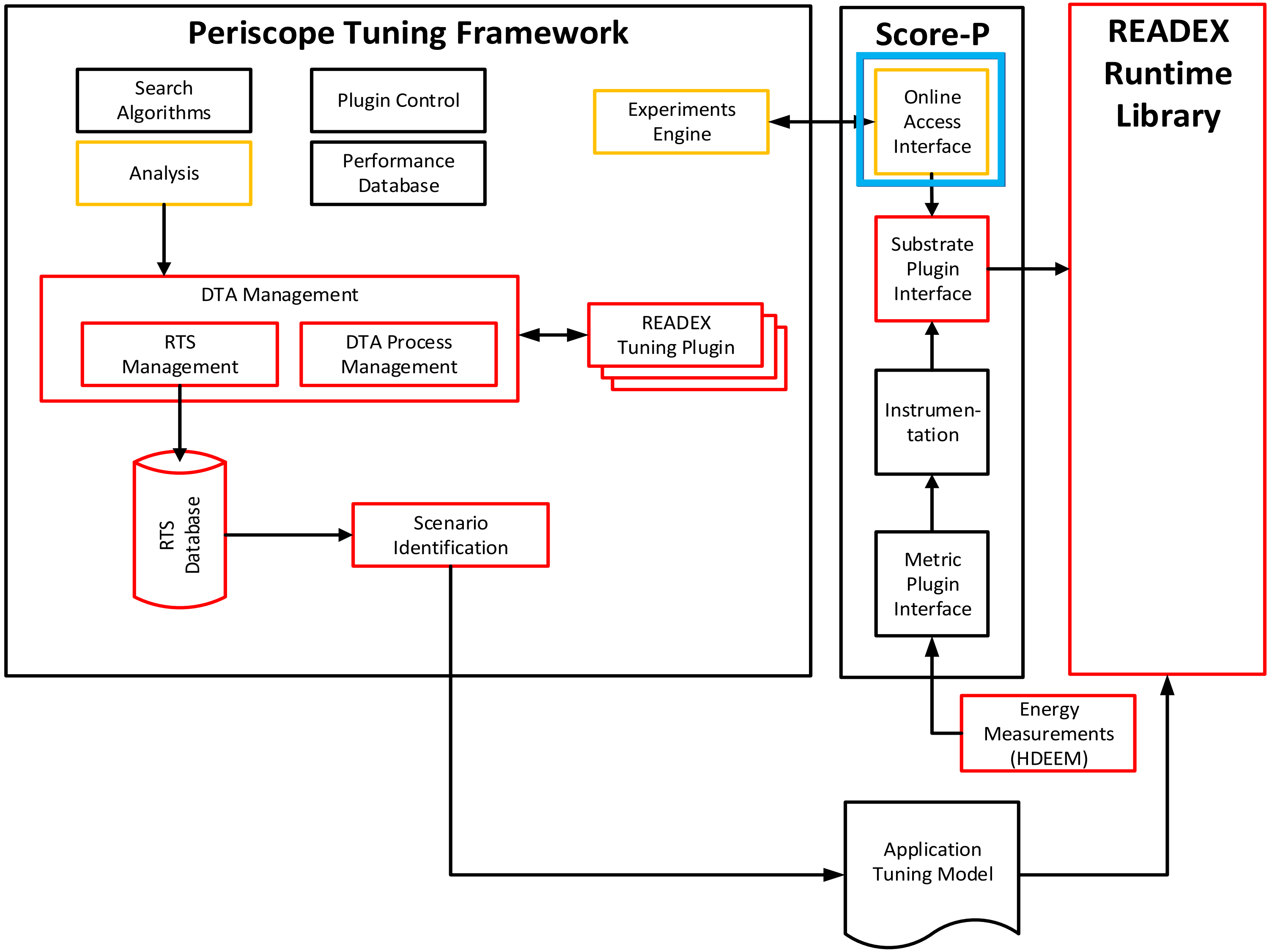
Pre-Computation of Configurations



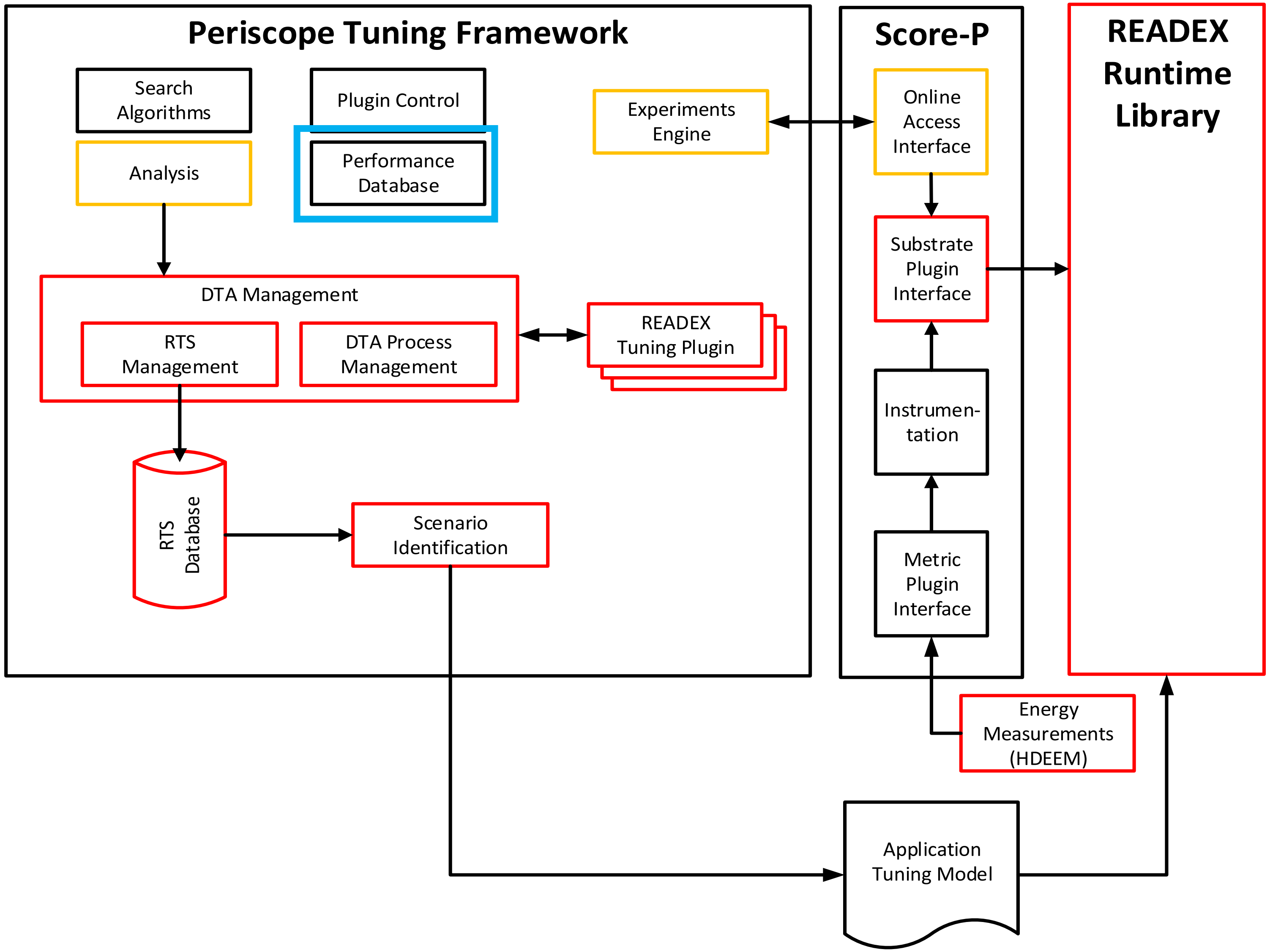
Pre-Computation of Configurations



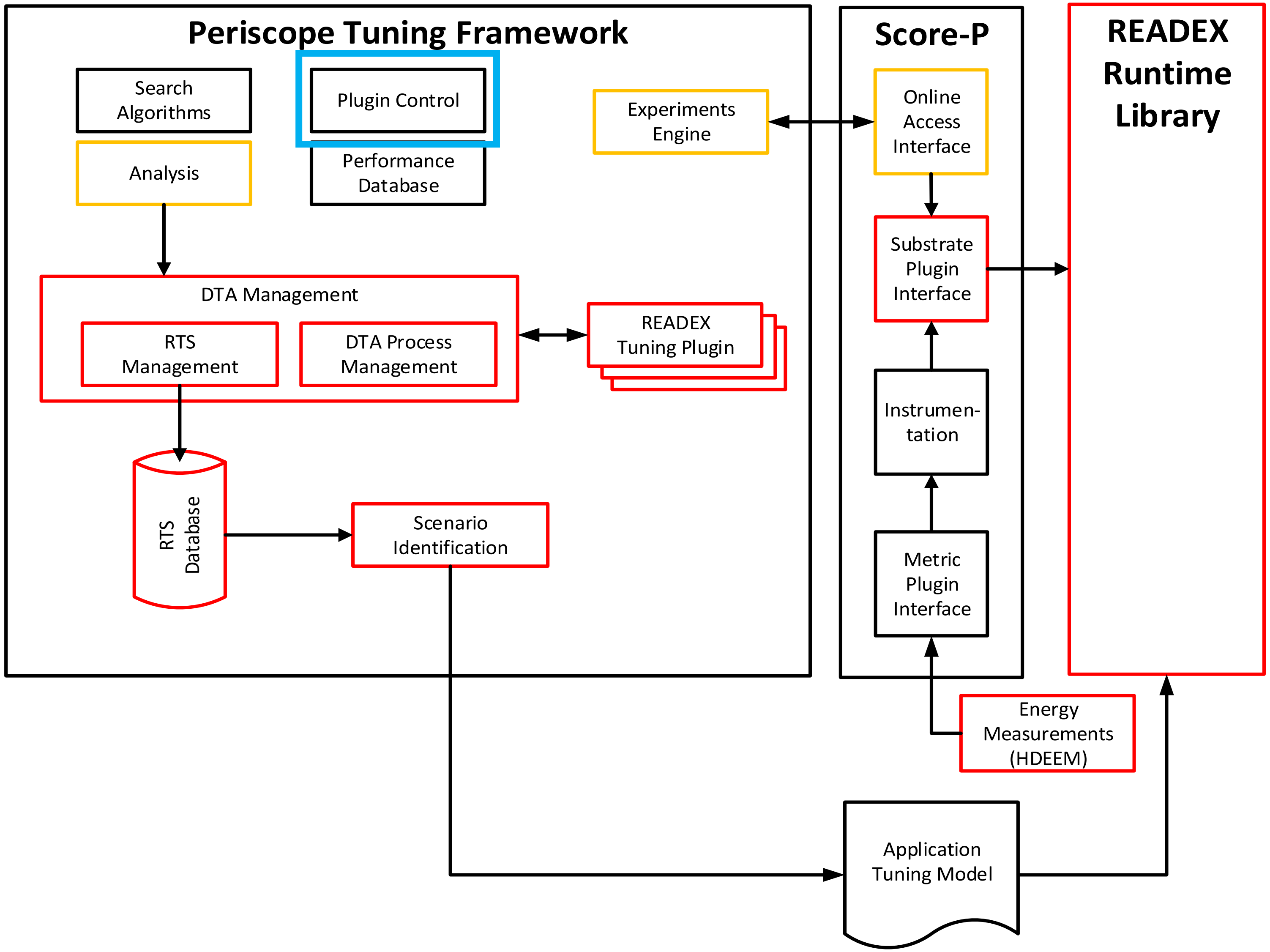
Pre-Computation of Configurations



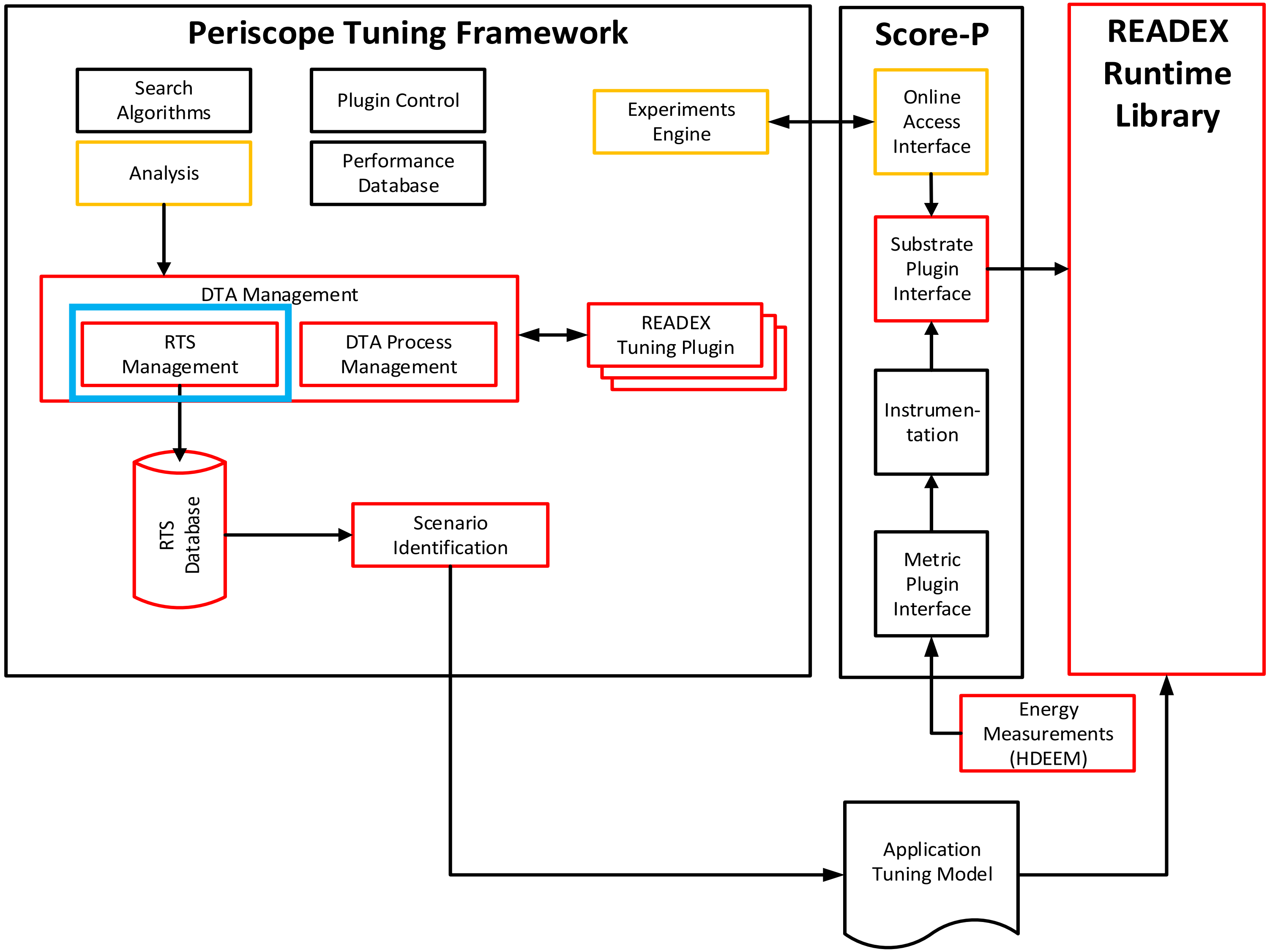
Pre-Computation of Configurations



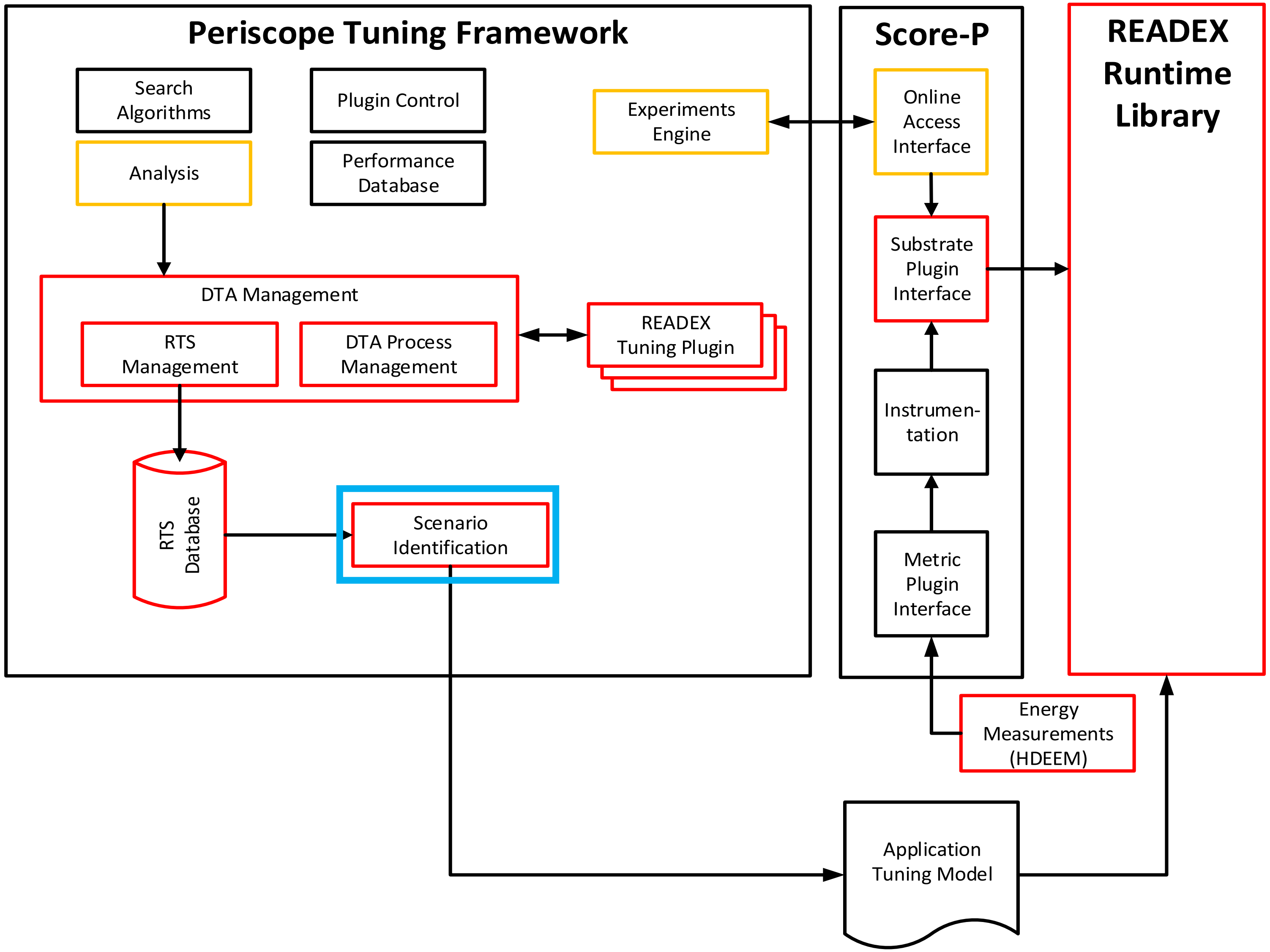
Pre-Computation of Configurations



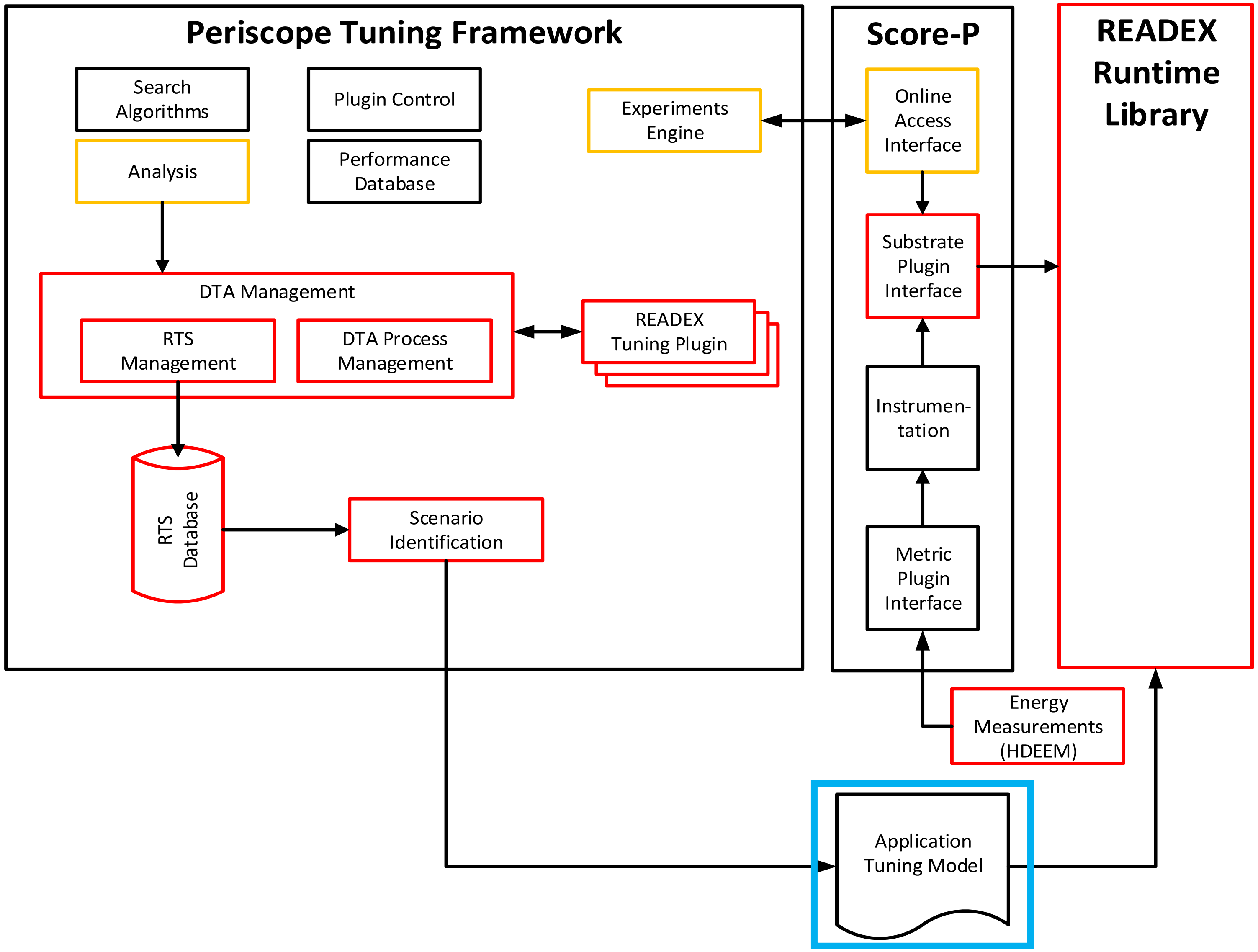
Pre-Computation of Configurations



Pre-Computation of Configurations



Pre-Computation of Configurations



Domain Knowledge Specification

- Improve tuning model
  - Distinguish more rts's
- So-called *Identifiers* to compute better system configurations
  - rts's
    - Region identifiers
  - Phase characteristics across phases
    - Phase identifiers
  - Executions with different application inputs
    - Input identifiers
- Application Tuning Parameter (ATP)
  - Switch application control flow

Identifier Specification- *region identifiers*

- Specify identifiers via Score-P user parameter
  - Identify different characteristics in rts's
- *Interpolate* the minimum grid level to the maximum
  - Computation switches from compute bound to memory bound
  - For DTA
    - determine special system configurations for compute and memory bound rts's
    - Add a region identifier to code for the grid level
    - Region name, call path and region identifier are used as identifiers.

Listing: Region Identifiers

```
!--- level k-1 to level k ---!
do k = min_level+1,max_level
    call interpolate(...,k)
    call resid(...)
    call psinv(...)
end do
```

```
!--- Interpolate to level k region ---!
subroutine interpolate(...,k)
SCOREP_USER_PARAMETER("level", k)
...
end subroutine
```

Identifier Specification- *phase* and *input identifiers*

Phase Identifiers

- Exploit dynamism of the application across phases

Input Identifiers

- Improve tuning model
  - Identify system configurations for different inputs characteristics.
- Example:
  - Grid level computation switches from compute to memory bound
    - depends on the resolution of the finest grid and the number of MPI processes.
      - The finer the grid, the more levels are memory bound.
      - The more processes are used, the fewer levels are memory bound due to an increased amount of cache.

Application Tuning Parameters (ATP)

- exploit the dynamism
    - through the use of different code paths (e.g. preconditioners)
- Identify the control variables responsible for control flow switching.
    - provides APIs to annotate the source code

Motivational ATP Example (Espreso)

- Finite Element (FEM) tools and domain decomposition based Finite Element Tearing and Interconnect (FETI) solvers.
- FETI Solver
  - contains a projected conjugate gradient (PCG) solver.
  - Convergence can be improved by several preconditioners.
- Evaluated preconditioners on a structural mechanics problem with 23 million unknowns
  - On a single compute node with 24 MPI processes.

15.9 s
4091.5 j

Preconditioner	# iterations	1 iteration		Solution	
None	172	125 ms	31.6 J	21.36 s	5 501.31 J
Weight function	100	130+2 ms	32.3+0.53 J	12.89 s	3 284.07 J
Lumped	45	130+10 ms	32.3+3.86 J	6.32 s	1 636.11 J
Light dirichlet	39	130+10 ms	32.3+3.74 J	5.46 s	1 409.82 J
Dirichlet	30	130+80 ms	32.3+20.62 J	6.34 s	1 594.50 J

European Commission
Horizon 2020
European Union funding for Research & Innovation

Conclusion

- Aim to improving the energy efficiency of HPC applications by a dynamic tuning approach.
- At design time, tuning parameters is determined
  - tuning model guides the dynamic switching
- Tuning model enhanced
  - domain knowledge that is provided by the application owner.
- To know more about the project, go to http://www.readex.eu/

Discussion