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

EnA-HPC @ ISC’17

Robert Schöne – TUD

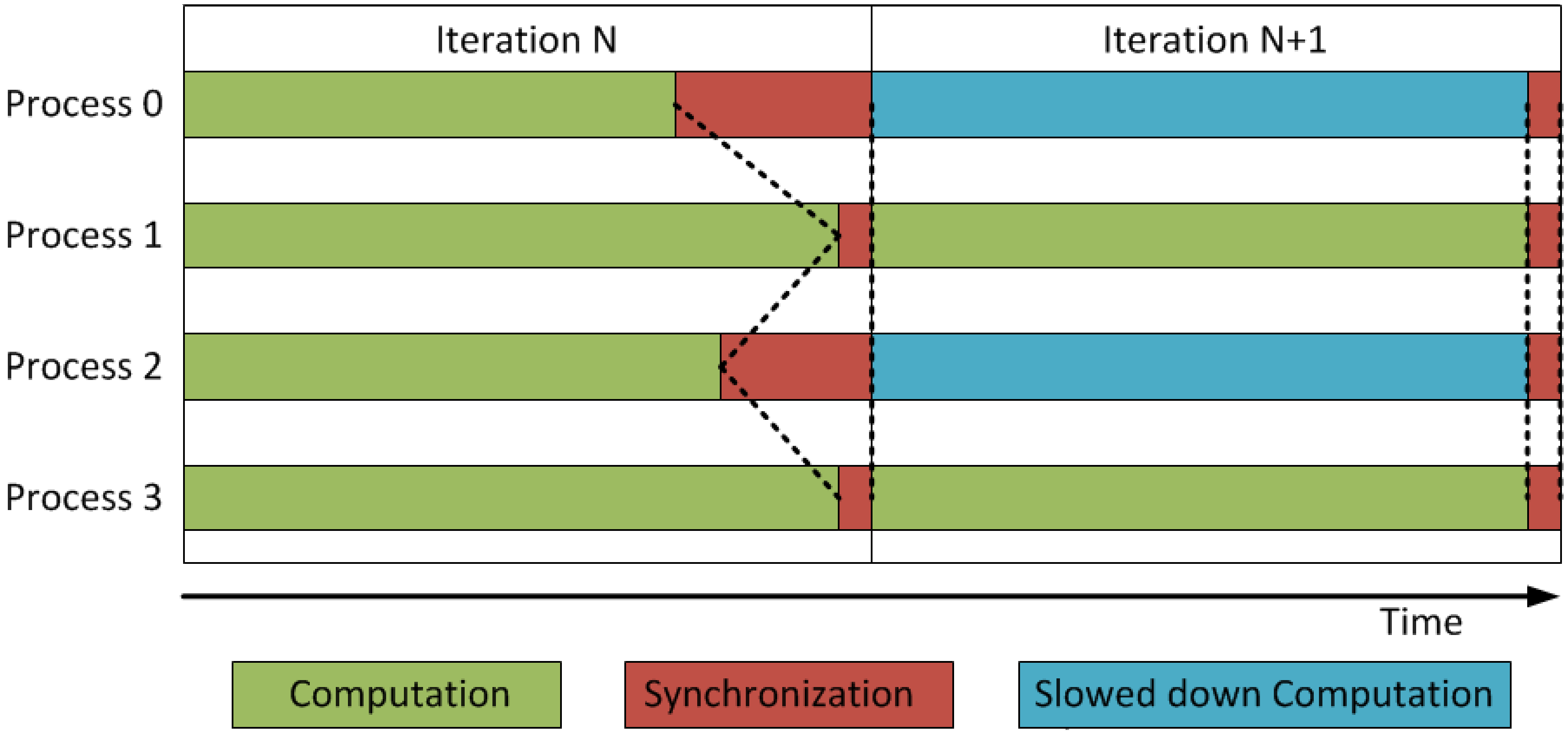
READEX
Runtime Exploitation of Application Dynamism for Energy-efficient eXascale computing

European Commission
Horizon 2020
European Union funding for Research & Innovation

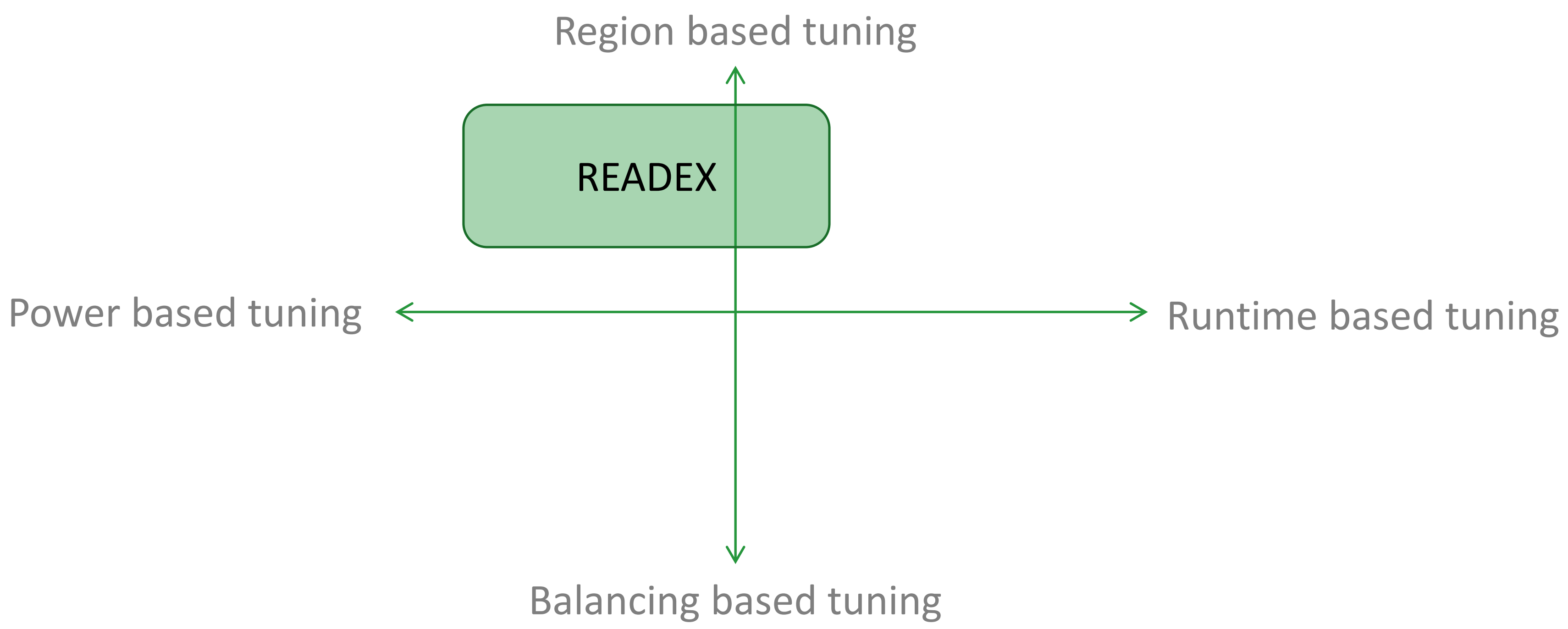
Energy Efficiency Tuning Types

1. Static or dynamic tuning?
   - Uniform or changing behavior of programs
   - Dynamic: sampling or instrumentation
2. Reducing power or runtime?
   - Power:
     - Frequencies
     - C-States
     - Speculative execution (e.g., prefetchers)
   - Runtime:
     - Frequencies (Turbo, various resources share single power budget)
     - Select optimal code paths
     - Optimize code
3. Tackling regions or synchronization?

Energy Efficiency Tuning Types



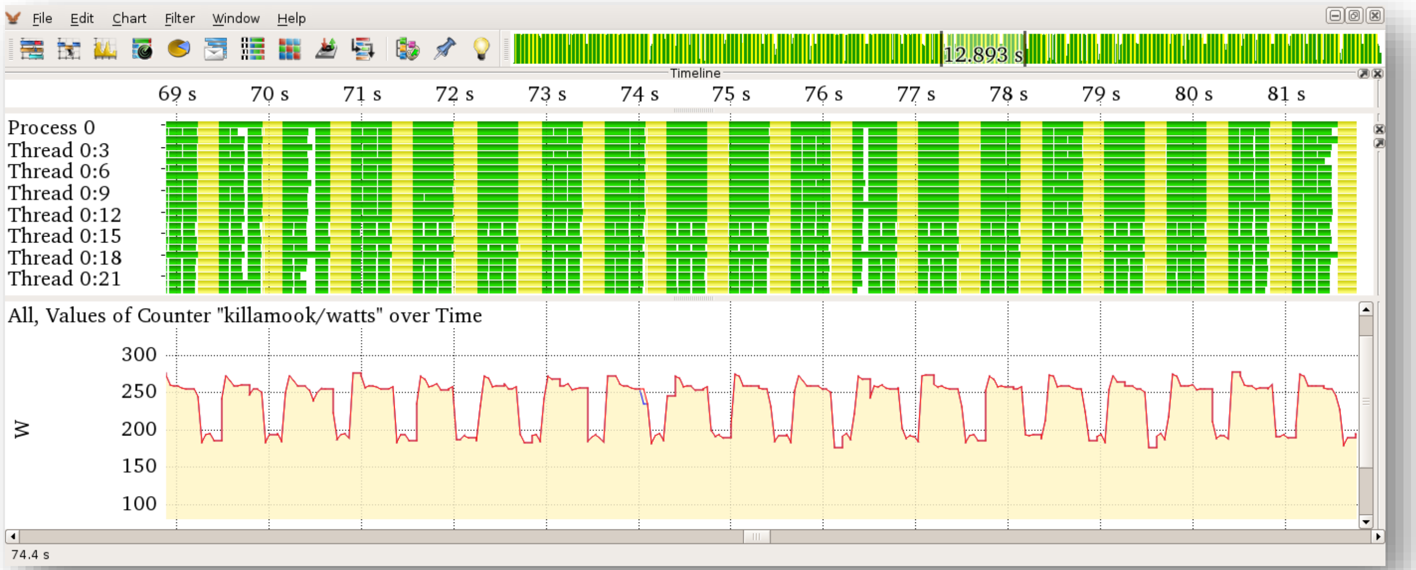
Energy Efficiency Tuning with READEX



Project Motivation

Applications exhibit dynamic behaviour

- Changing resource requirements
- Computational characteristics
- Changing load on processors over time



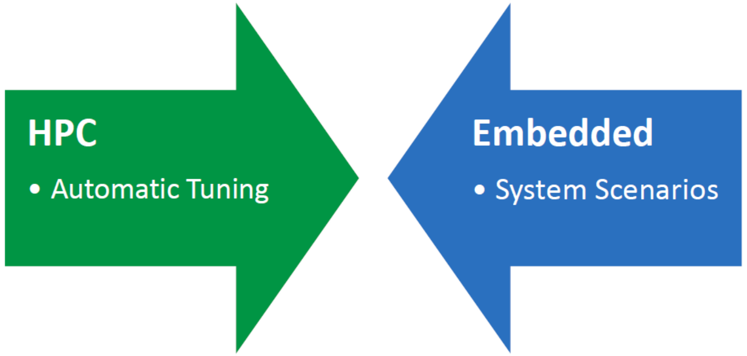
Overview

READEX creates a **tools-aided methodology for automatic tuning** of parallel applications

- Dynamically adjust system parameters to actual resource requirements

Join technologies from embedded systems and HPC

- HPC: PTF, Score-P, and HDEEM
- ES: System scenario methodology

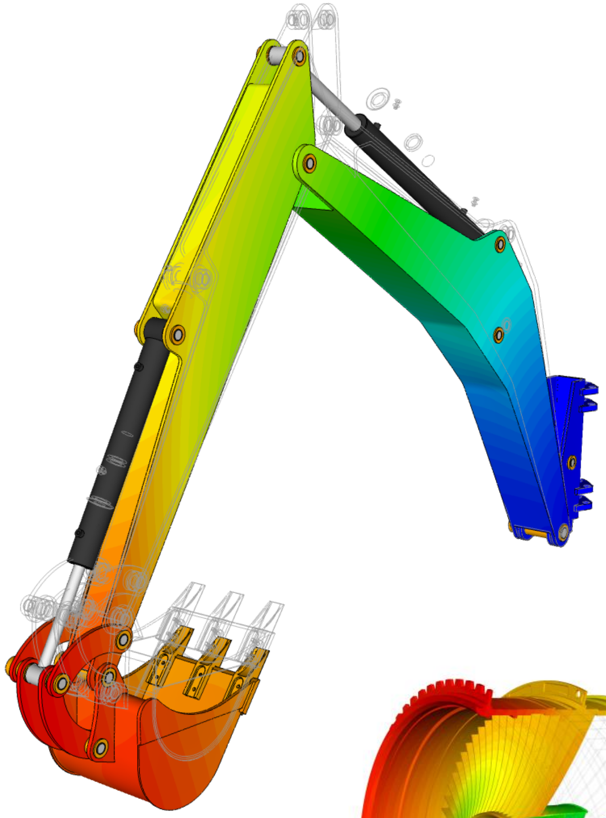


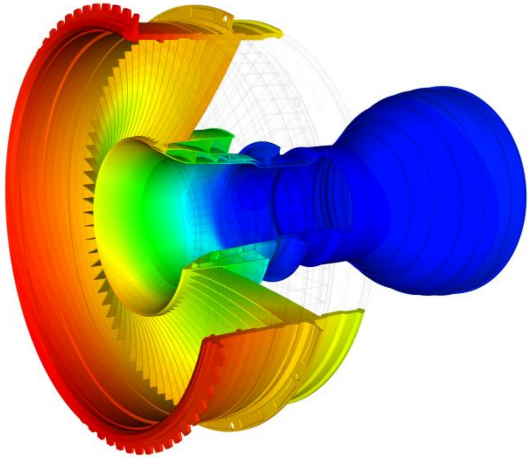
Overview

Co-design approach

- Manual tuning for energy efficiency as a baseline
- Automatic tuning for comparison

- Applications
    - PERMON and ESPRESO (FETI tools from IT4Innovations)
    - Indeed (GNS)
    - CORAL benchmark suite
    - ProxyApps





Project Partners

- Grant agreement No 671657
- Officially started September 1st, 2015

- Technische Universität Dresden/ZIH (Coordinator)
- Norwegian University of Science and Technology
- Technische Universität München
- IT4Innovations, VSB-Technical University of Ostrava
- NUI Galway, Irish Centre for High-End Computing
- Intel France
- Gesellschaft für numerische Simulation mbH

Terminology: Region and Region Instance

```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

Significant region

Runtime situation

Terminology: Tuning Parameter and Intra-Phase Dynamism

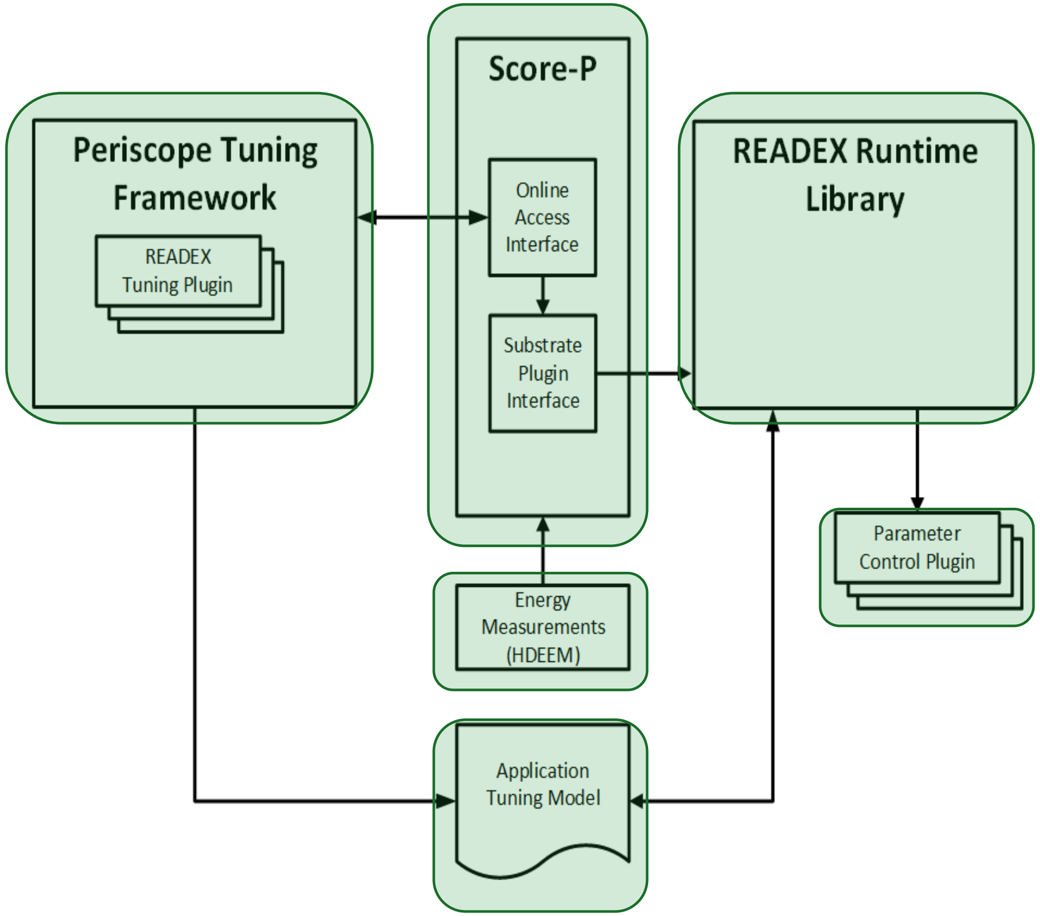
```c
int main(void) {

    // 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();          // signi
       // End phase region
    }

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

    MPI_Finalize();
    return 0;
}
```

Tuning Parameter
FREQ=2 GHz

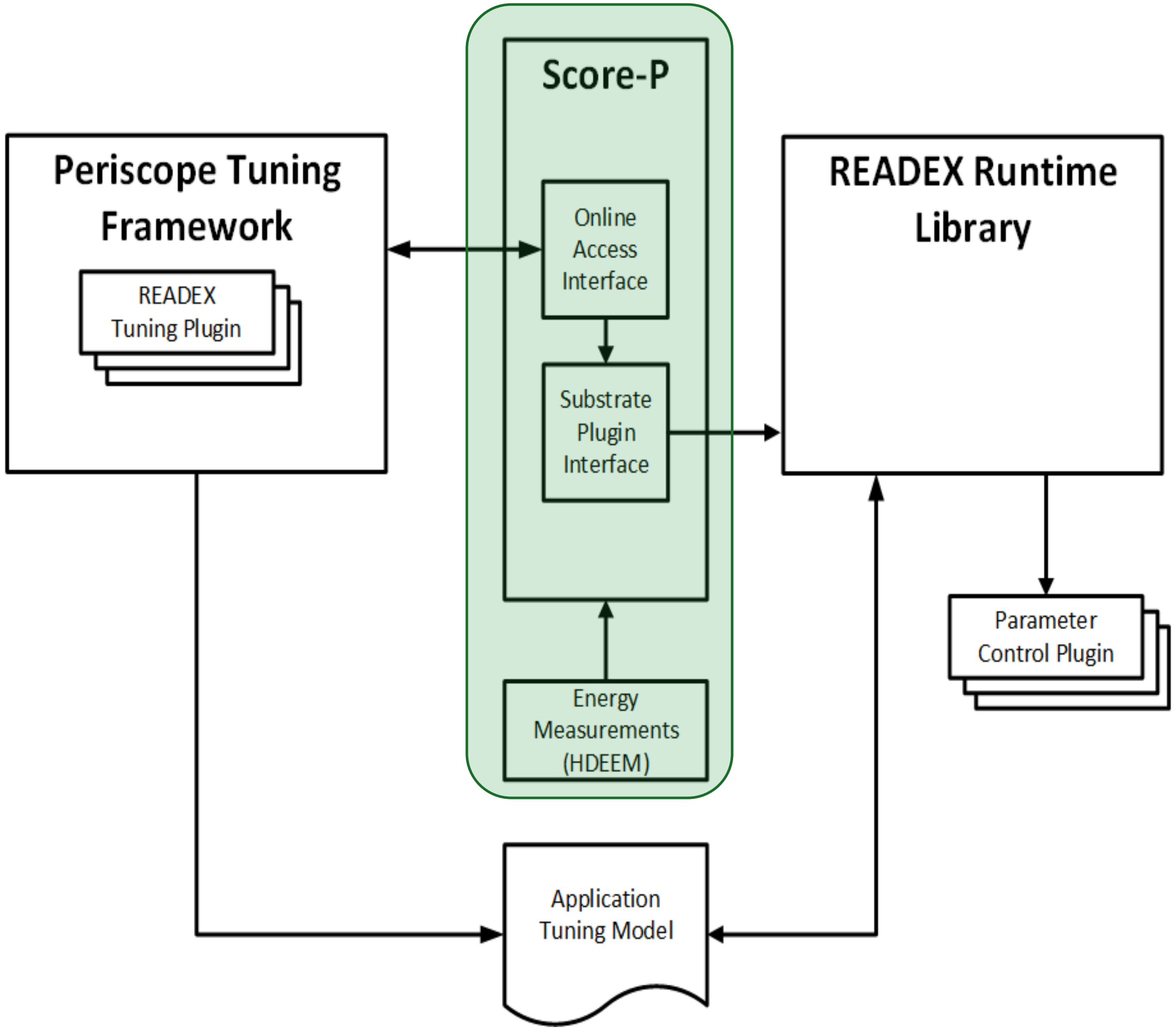
Tuning Parameter
FREQ=1.5 GHz

Workflow

1. Instrument application
   - Score-P provides different kinds of instrumentation
2. Detect dynamism
   - Check whether runtime situations could benefit from tuning
3. Detect energy saving potential and configurations (DTA)
   - Use tuning plugin and power measurement infrastructure to search for optimal configuration
   - Create tuning model
4. Runtime application tuning (RAT)
   - Apply tuning model, use optimal configuration



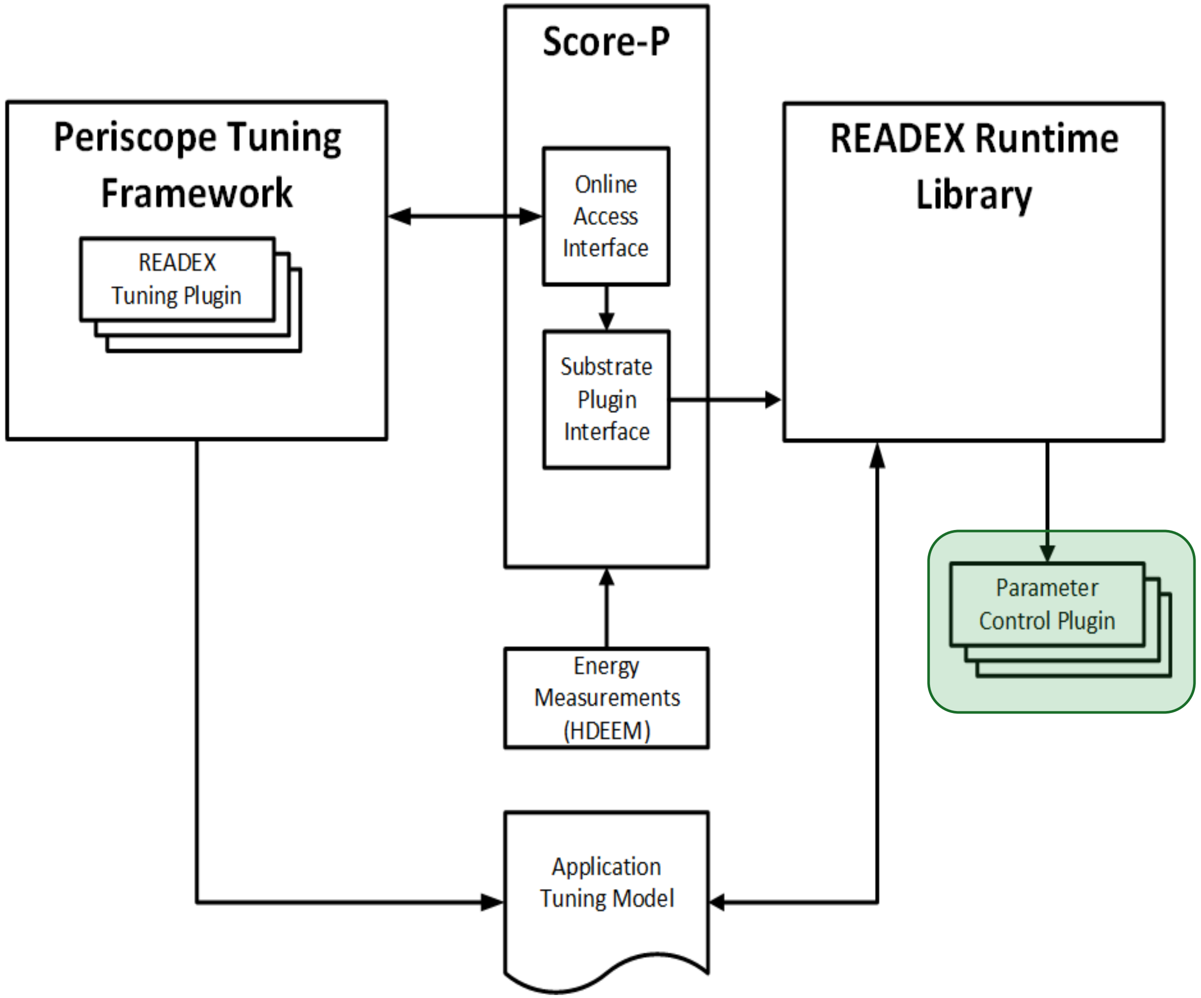
Instrumentation via Score-P



Instrumentation via Score-P

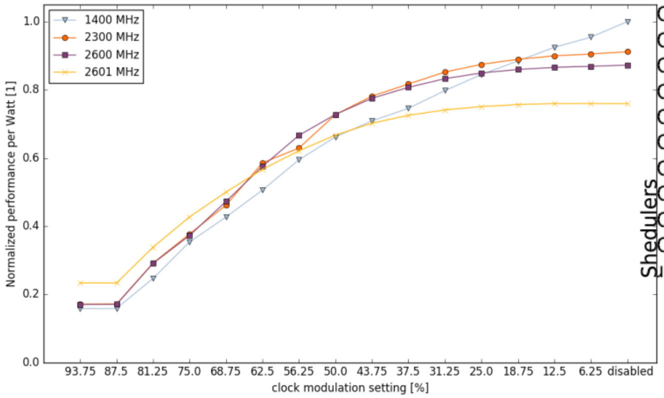
- HPC performance measurement infrastructure
  - Creates CUBEx profiles or OTF2 traces
  - Instrumentation and sampling
  - Supports most HPC programming paradigms
  - Mechanism for online usage of data – Periscope
  - Efficient implementation
  - Power measurement plugins (see talk by T. Ilsche)
- Re-use and extend existing infrastructure
  - Parse CUBEx profiles to find significant regions
  - Score-P Substrate Plugins
  - Use READEX Runtime Library (RRL) to change parameters
  - Support tools to lower measurement overhead via filtering

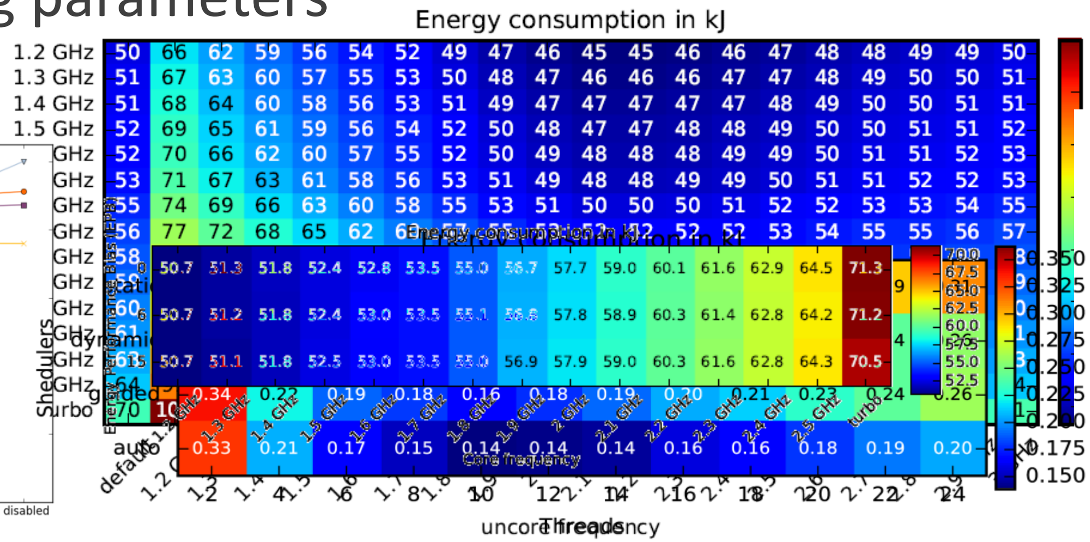
Toggling Parameters

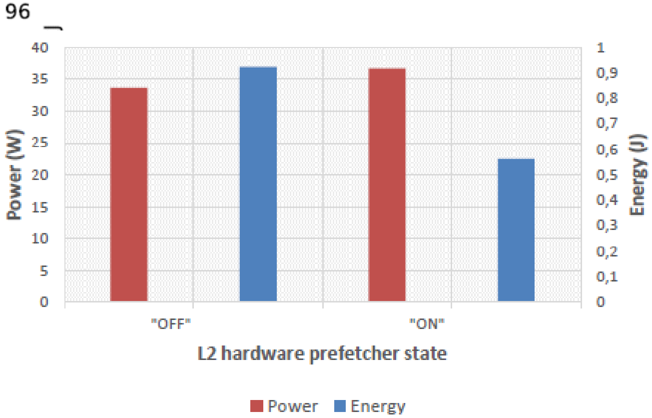


Toggling Parameters

- Hardware parameters
  - Core frequency, uncore frequency
  - Clock modulation, Energy Performance Bias, prefetchers
- Runtime parameters
  - Message Passing Interface, e.g., message size threshold
  - OpenMP parameters, e.g., loop scheduler, number of threads
- Application tuning parameters







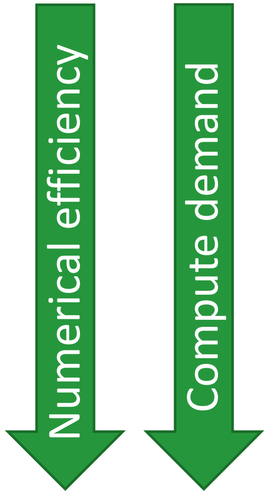
Application Tuning Parameters (Work in Progress)

```
// C example
// register parameters at READEX
  ATP_PARAM_DECLARE("PARAMETER1", ATP_PARAM_TYPE_RANGE, 1, "Domain1");
// declare set of possible values for the parameter
  ATP_PARAM_ADD_VALUES("PARAMETER1", values_array, num_values, "Domain1")
// getting parameter setting from READEX, store in variable app_param
  ATP_PARAM_GET("PARAMETER1",&app_param,"Domain1")
// … usage of app_param, e.g., switch (app_param) {
```

Application Tuning Parameters (Work in Progress)

Application parameters example: different preconditioners in ESPRESO solver

- Full Dirichlet preconditioner is usually the preferred one (the best numerical properties)
- Depends on input dataset / problem that is solved
- All preconditioners have been evaluated with the optimal hardware parameter settings

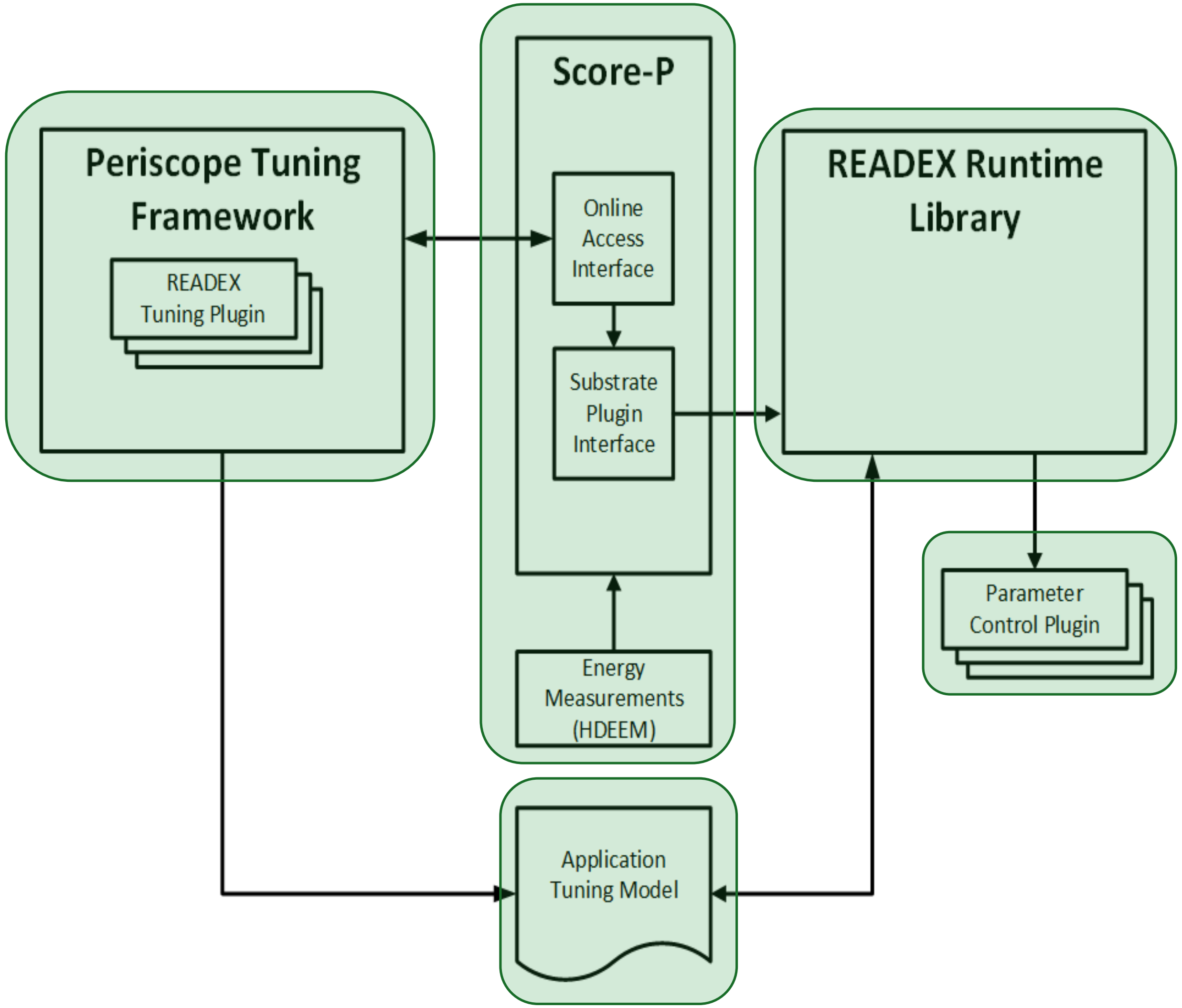


Preconditioner type	Number of iterations	Single iteration cost Time and energy		Total solution cost Time and energy	
No preconditioner	172	130 + 0 ms	32.3 + 0.00 J	21.4 s	5.50 kJ
Weight function	100	130 + 2 ms	32.3 + 0.53 J	12.9 s	3.28 kJ
Lumped	45	130 + 10 ms	32.3 + 3.86 J	6.3 s	1.64 kJ
Light Dirichlet	**39**	**130 + 10 ms**	**32.3 + 3.74 J**	**5.5 s**	**1.41 kJ**
Full Dirichlet (default)	**30**	**130 + 80 ms**	**32.3 + 20.6 J**	**6.3 s**	**1.59 kJ**

11.3% energy savings against the default full Dirichlet preconditioners

Note: 130 ms and 32.3 J – is a baseline for single iteration cost without preconditioner

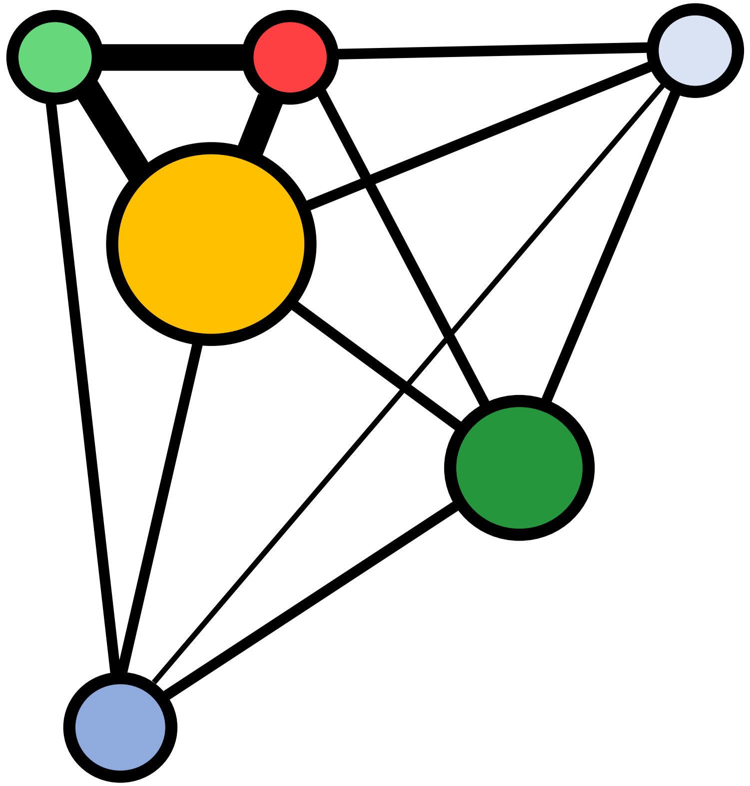
Design Time Analysis



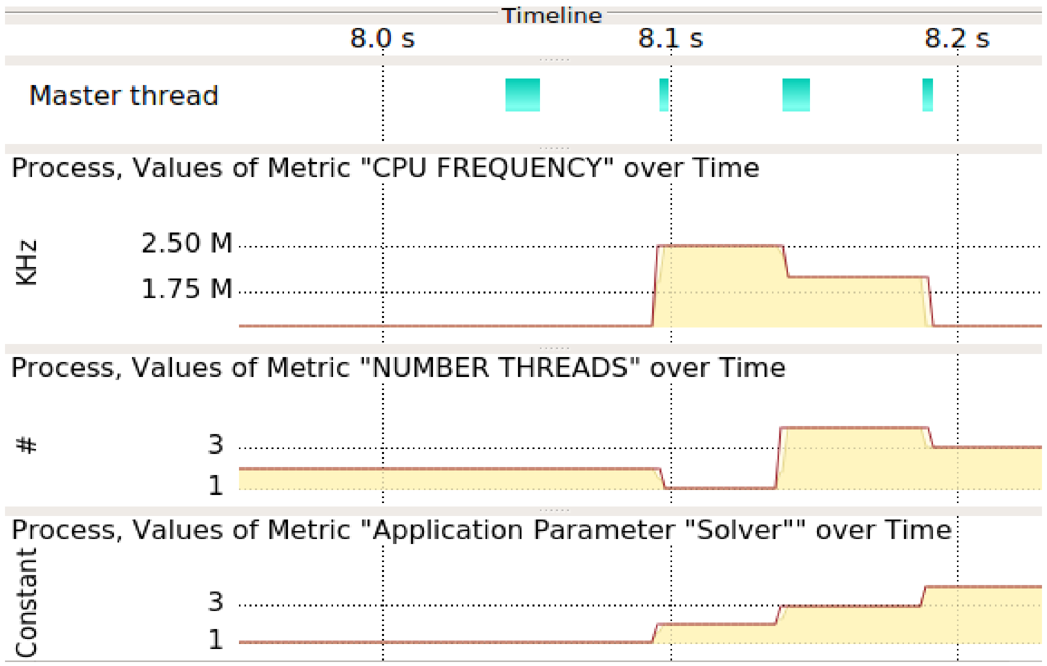
Design Time Analysis

- Periscope Tuning Framework
- Pre-computation of dynamicity and significant regions
- Different objectives (e.g., runtime, energy, EDP)
- Different search strategies (complete, random, genetic)
- Uses RRL to switch parameters
- Determine best configuration for significant regions
- Current work in progress:
    - Cluster regions in scenarios
    - Application tuning parameters

Tuning Model Visualization

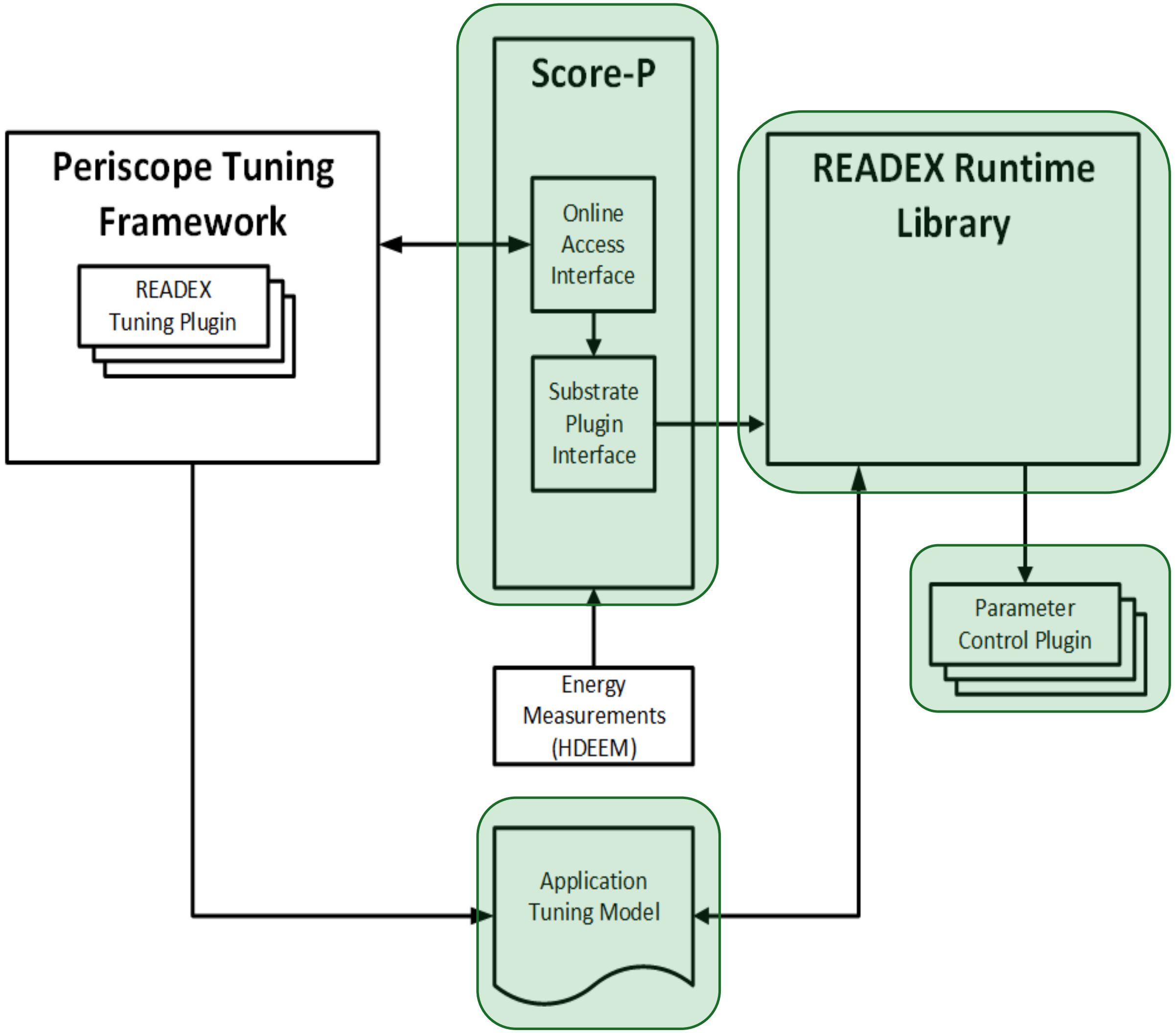


Force graph for scenarios



Vampir visualization of parameter changes

Runtime Tuning



Runtime Tuning

- READEX Runtime Library
- Reads and applies Tuning Model
- Sets and resets configuration at runtime
- Current work in progress:
  - Application tuning parameters
  - Online calibration mechanism
  - Advanced switching decision making

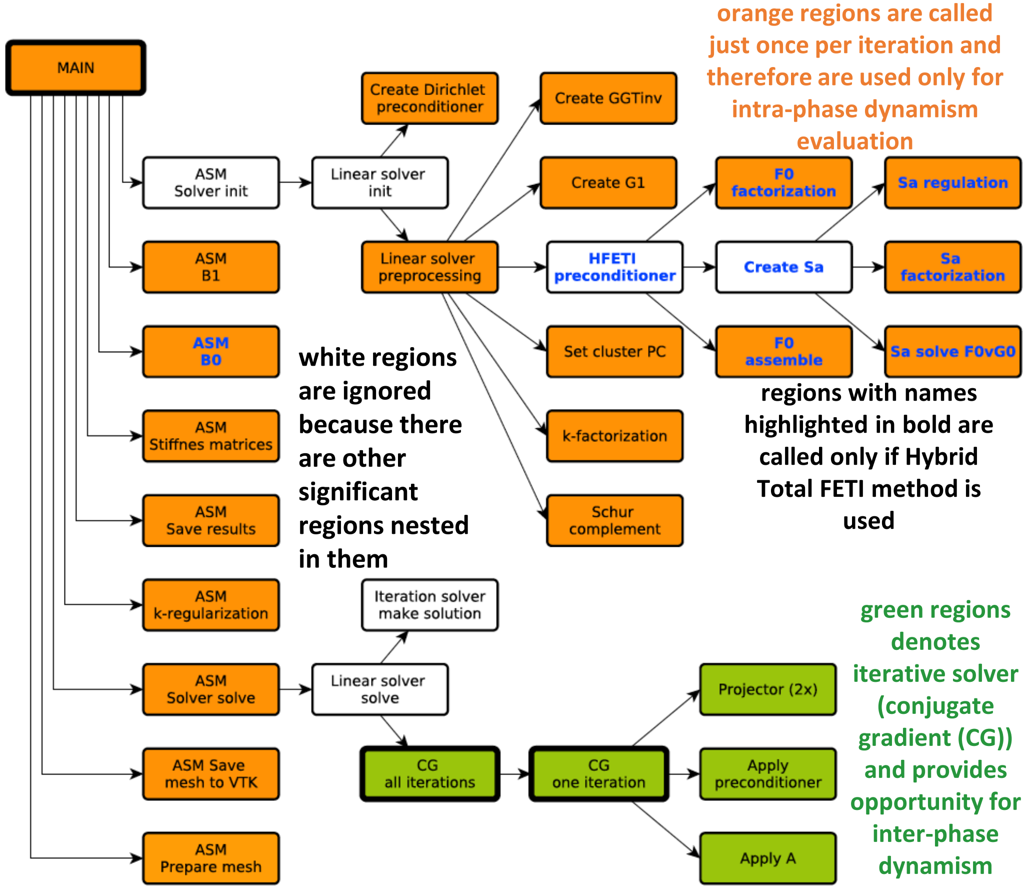
Tuning Potential

static dynamic total

ESPRESO: 12.3% + 9.1% = 20.3%

- Structural mechanics code
- Finite element + sparse FETI solver

Region	% of 1 phase	Best static configuration	Value	Best dynamic configuration	Value	Dynamic savings
Assembler–AssembleStiffMat	14.32	18 threads, 1.8 GHz UCF, 2.5 GHz CF	733.73 J	20 threads, 2.0 GHz UCF, 2.5 GHz CF	731.22 J	2.51 J (0.34%)
Assembler–Assemble-B1	2.23	18 threads, 1.8 GHz UCF, 2.5 GHz CF	114.30 J	2 threads, 2.2 GHz UCF, 2.5 GHz CF	94.15 J	20.15 J (17.63%)
Cluster–CreateF0-FactF0	0.17	18 threads, 1.8 GHz UCF, 2.5 GHz CF	8.71 J	6 threads, 1.6 GHz UCF, 2.5 GHz CF	6.90 J	1.80 J (20.73%)
Assembler–SaveResults	3.10	18 threads, 1.8 GHz UCF, 2.5 GHz CF	158.81 J	2 threads, 1.2 GHz UCF, 2.5 GHz CF	147.66 J	11.16 J (7.03%)
			...			
Cluster–CreateSa-SaReg	0.17	18 threads, 1.8 GHz UCF, 2.5 GHz CF	8.59 J	8 threads, 2.0 GHz UCF, 2.5 GHz CF	7.03 J	1.56 J (18.15%)
Total value for static tuning for significant regions			733.73 + 114.30 + 8.71 + 158.81 + 278.39 + 113.87 + 14.23 + 658.07 + 325.69 + 99.93 + 74.70 + 641.88 + 1578.06 + 13.28 + 24.20 + 278.22 + 8.59 = 5124.66 J			
Total savings for dynamic tuning for significant regions			2.51 + 20.15 + 1.80 + 11.16 + 47.01 + 16.41 + 5.31 + 28.45 + 29.03 + 19.08 + 0.16 + 2.49 + 288.21 + 0.77 + 1.88 + 23.24 + 1.56 = 499.22 J of 5124.66 J (9.74 %)			
Dynamic savings for application runtime			499.22 J of 5493.55 J (9.09 %)			
Total value after savings			4994.33 J (79.72 % of 6265.18 J)			



	Default settings	Default values	Best static configuration	Static Savings	Dynamic Savings
Energy consumption [J], Blade summary	24 threads, 3.0 GHz UCF, 2.5 GHz CF	6265.18 J	18 threads, 1.8 GHz UCF, 2.5 GHz CF	771.63 J (12.32%)	499.2 J of 5493.6 J (9.09 %)
Runtime of function [s], Job info - hdeem	24 threads, 3.0 GHz UCF, 2.5 GHz CF	29.55 s	22 threads, 3.0 GHz UCF, 2.5 GHz CF	0.01 s (0.04%)	0.82 s of 29.54 s (2.76 %)

Discussion


READEX
application
tuning
system
systems
applications
energy-efficiency
performance
auto-tuning
dynamic
energy
HPC
results
dynamism
knowledge
research
methodology
European
analysis
run-time
scenario
partners
data
used
parameters
configurations
manual
framework
users
scenarios
based
automatic
Exascale
community
information
technologies
potential
process
computing
developed
user
future
activities
tools
industry
software
impact
new
Tool
improve
approach
first
also
Suite
time
using
well
domain
objective
available
develop
exploitation

Backup

Tuning Parameters

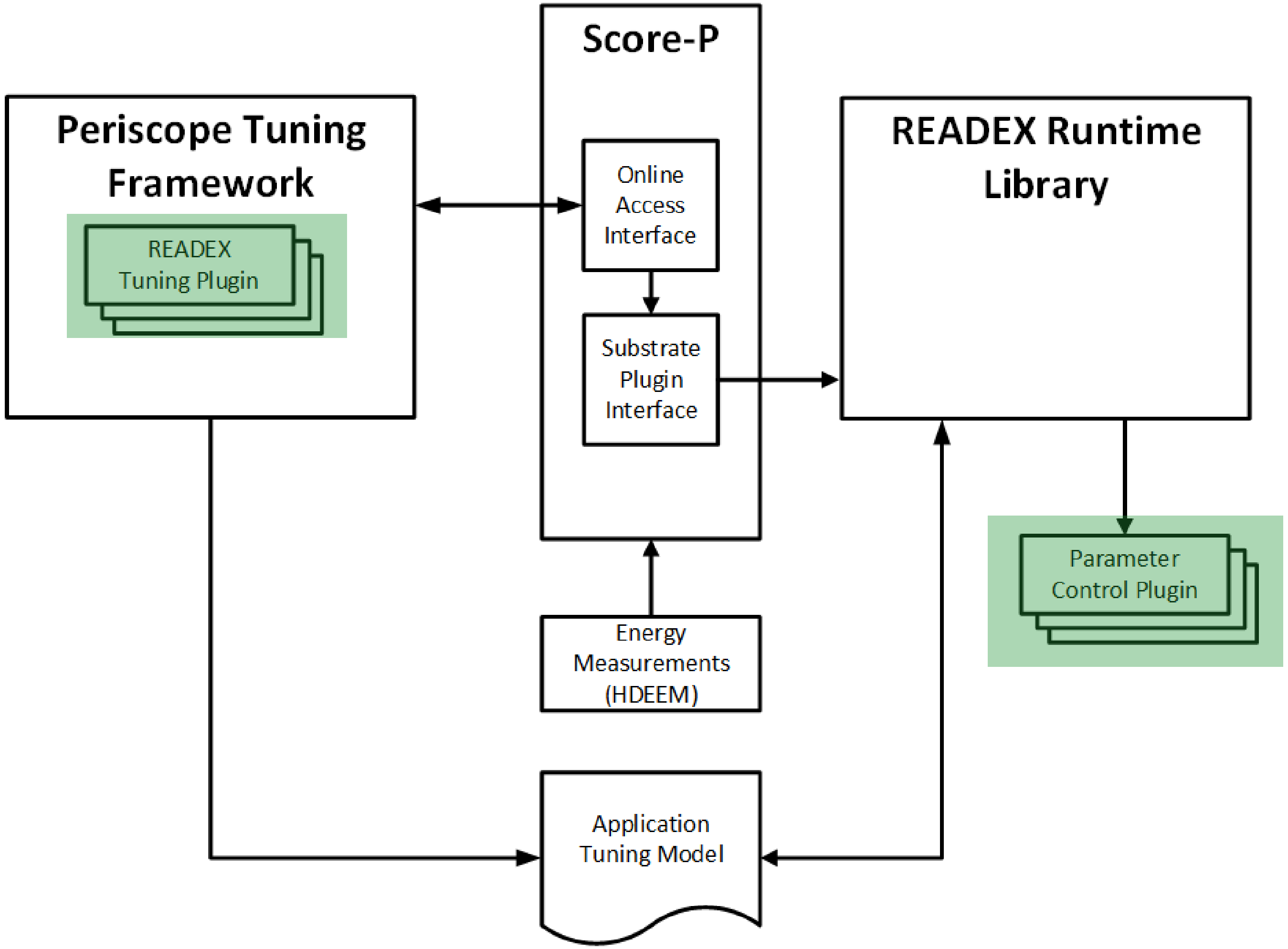
READEX READEX EAB Meeting

Robert Schöne – TUD

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

European Commission
Horizon 2020
European Union funding
for Research & Innovation

WP1 Implementation



Hardware Parameters

TUD, INTEL (M01 – M12)

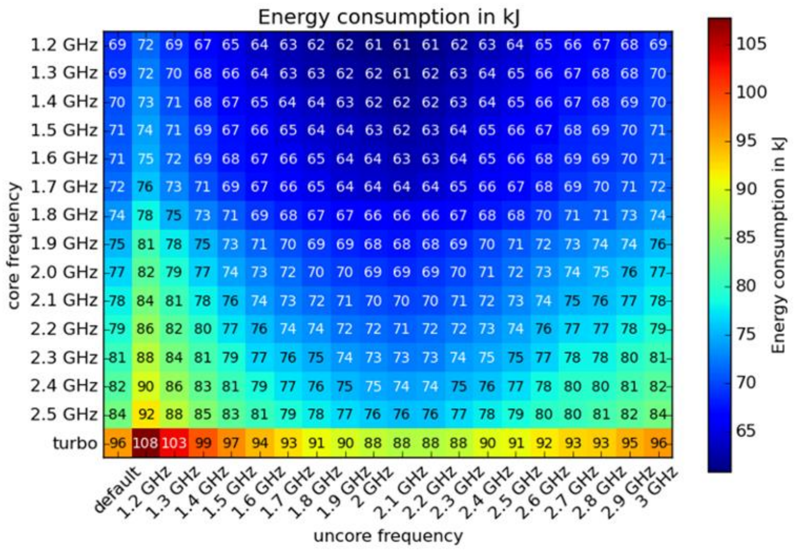
Existing Hardware parameters

- Dynamic Voltage and Frequency Scaling (DVFS)
- Uncore Frequency
- Energy Performance Bias (EPB)

Discarded parameters

- Dynamic Duty Cycle Management (DDCM, T-states)

DRAM

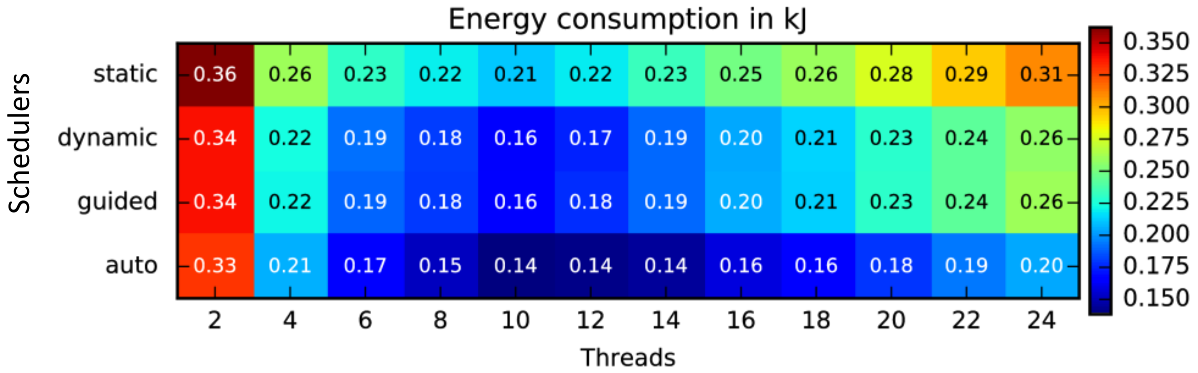


Application Parameters

```
// C example
// register parameters at READEX
  ATP_PARAM_DECLARE("PARAMETER1", ATP_PARAM_TYPE_RANGE, 1, "Domain1");
// declare set of values for the parameter
  ATP_PARAM_ADD_VALUES("PARAMETER1", values_array, num_values, "Domain1")
// getting parameter setting from READEX, store in variable app_param
  ATP_PARAM_GET("PARAMETER1",app_param,"Domain1")
```

System-software Parameters Investigation

<u>TUD</u>, IT4I, INTEL (M01-12)



- Message Passing Interface
  - Short message size threshold
  - SMP-awareness
  - Relevant for MPI_AlltoAll and MPI_Reduce
- OpenMP Threading
  - Dynamic Concurrency Throttling (change number of threads)
  - Workload scheduling algorithm
  - Chunksize

Design Time Analysis

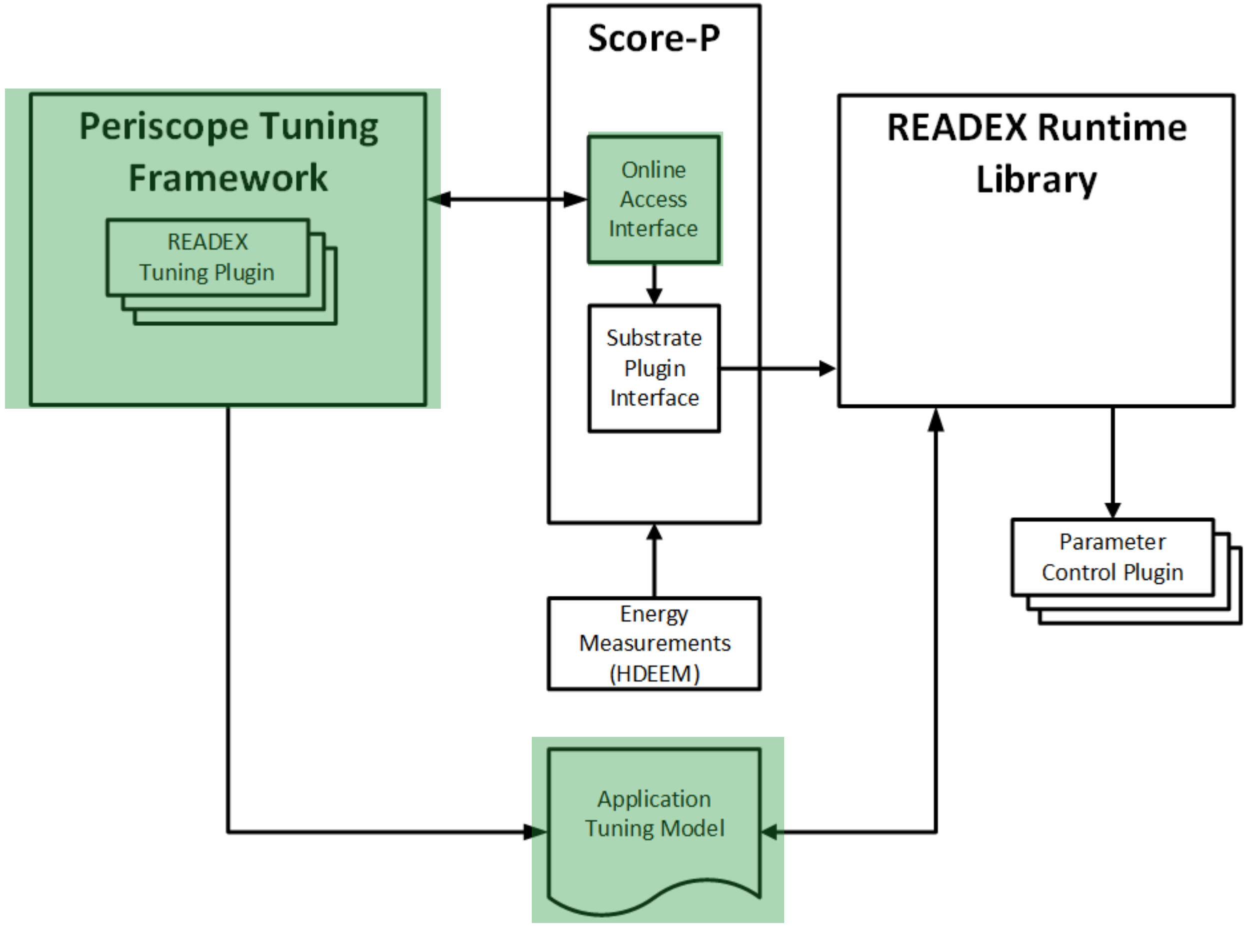
READEX READEX EAB Meeting

Robert Schöne – TUD

Runtime Exploitation of Application Dynamism for Energy-efficient eXascale computing

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

WP2 Implementation



Terminology: Region and Region Instance

```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

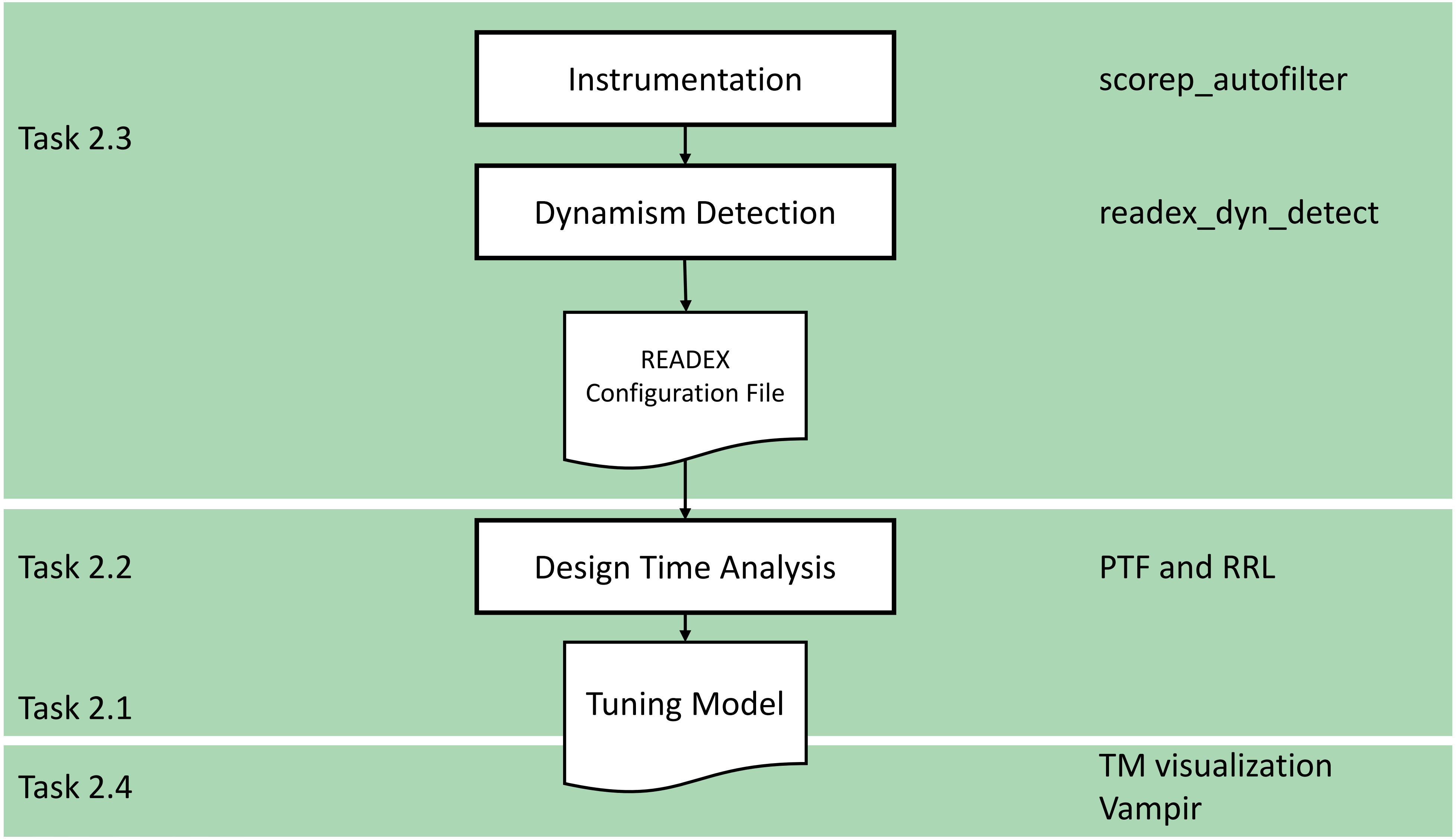
FREQ=2 GHz

FREQ=1.5 GHz

Significant region

Runtime situation

Design Time Analysis



Periscope Tuning Framework

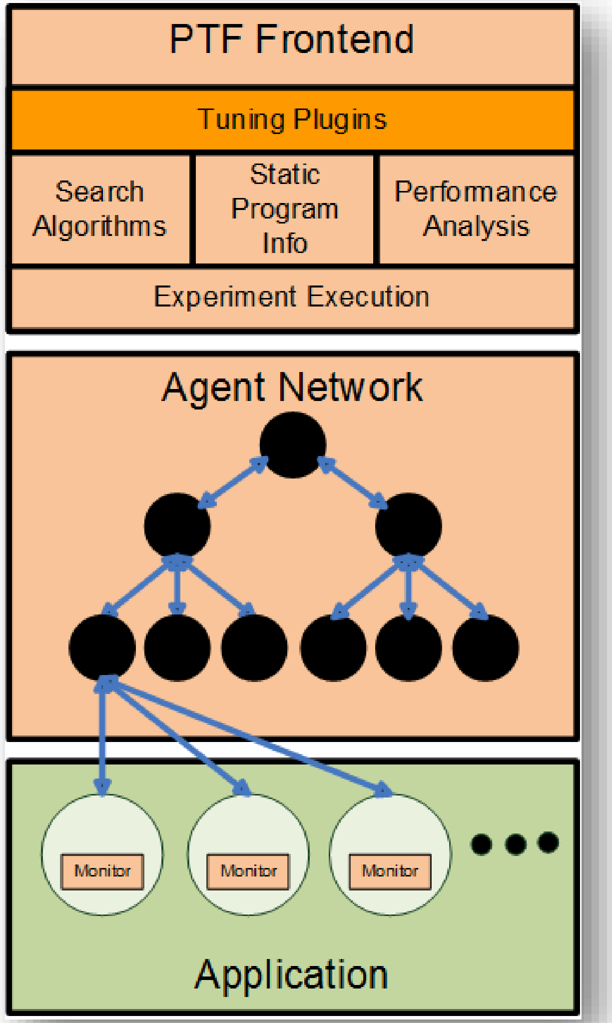
Automatic application analysis & tuning

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

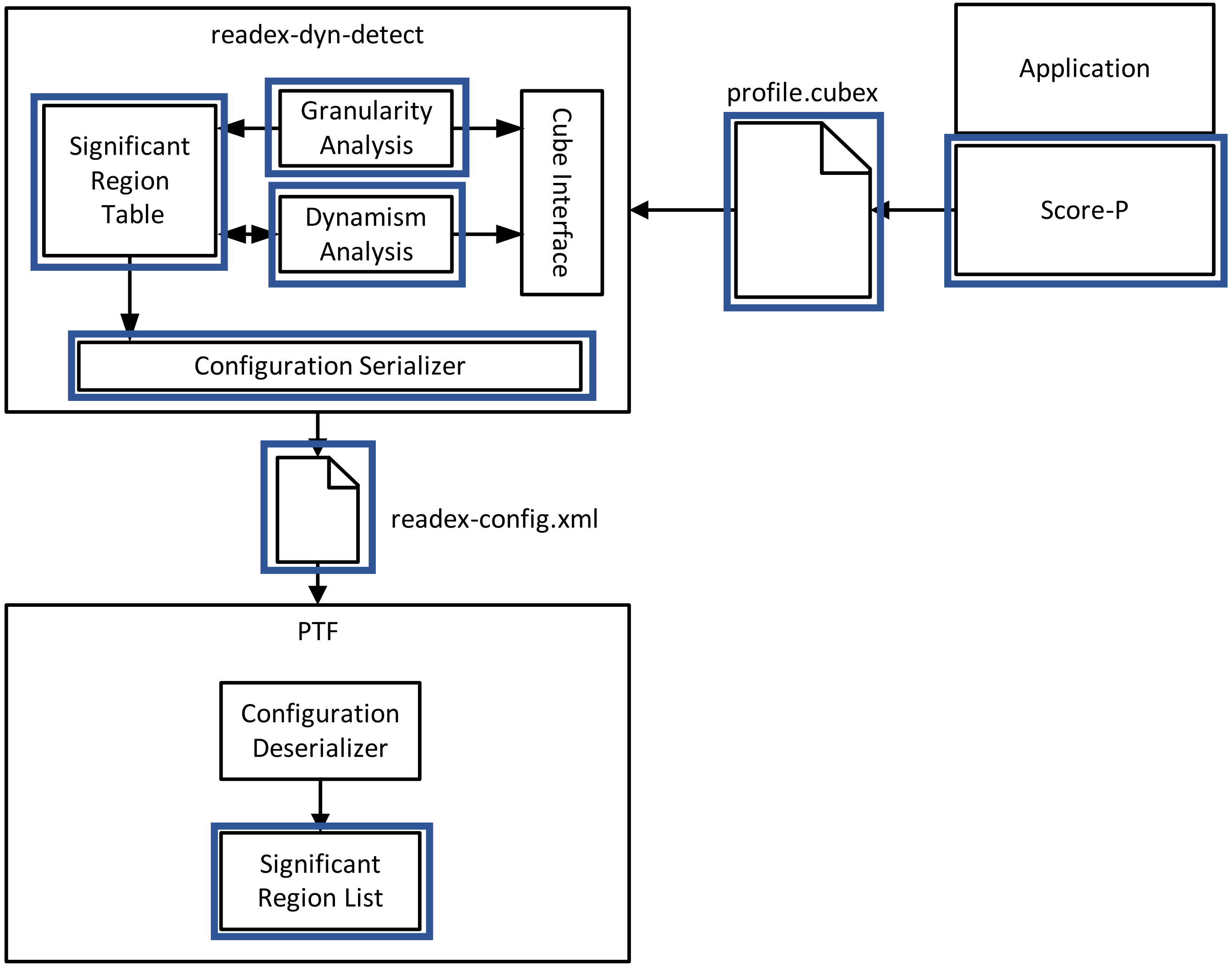
Support variety of parallel paradigms

- MPI, OpenMP, OpenCL, Parallel pattern

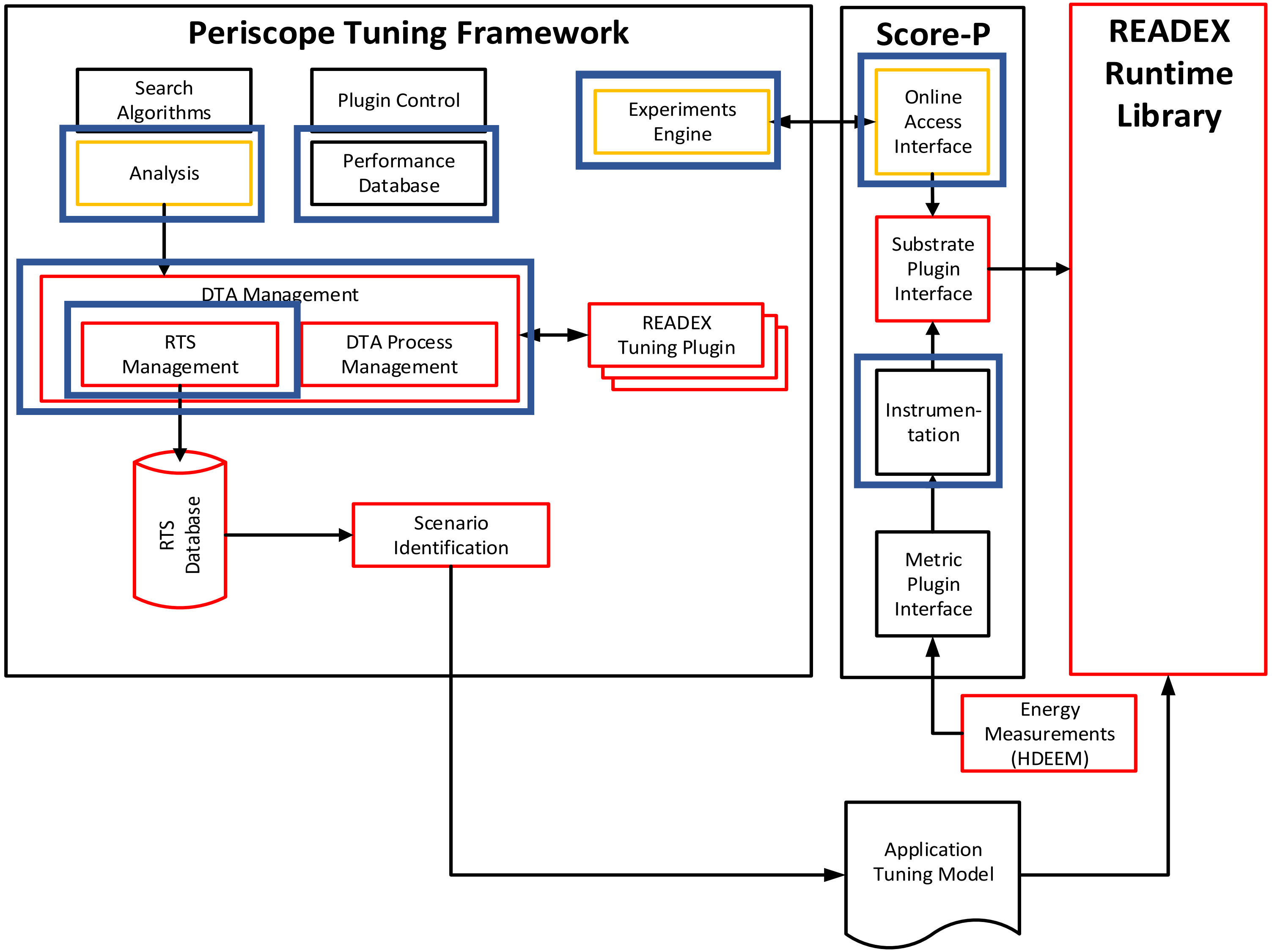
Developed in the AutoTune EU-FP7 project



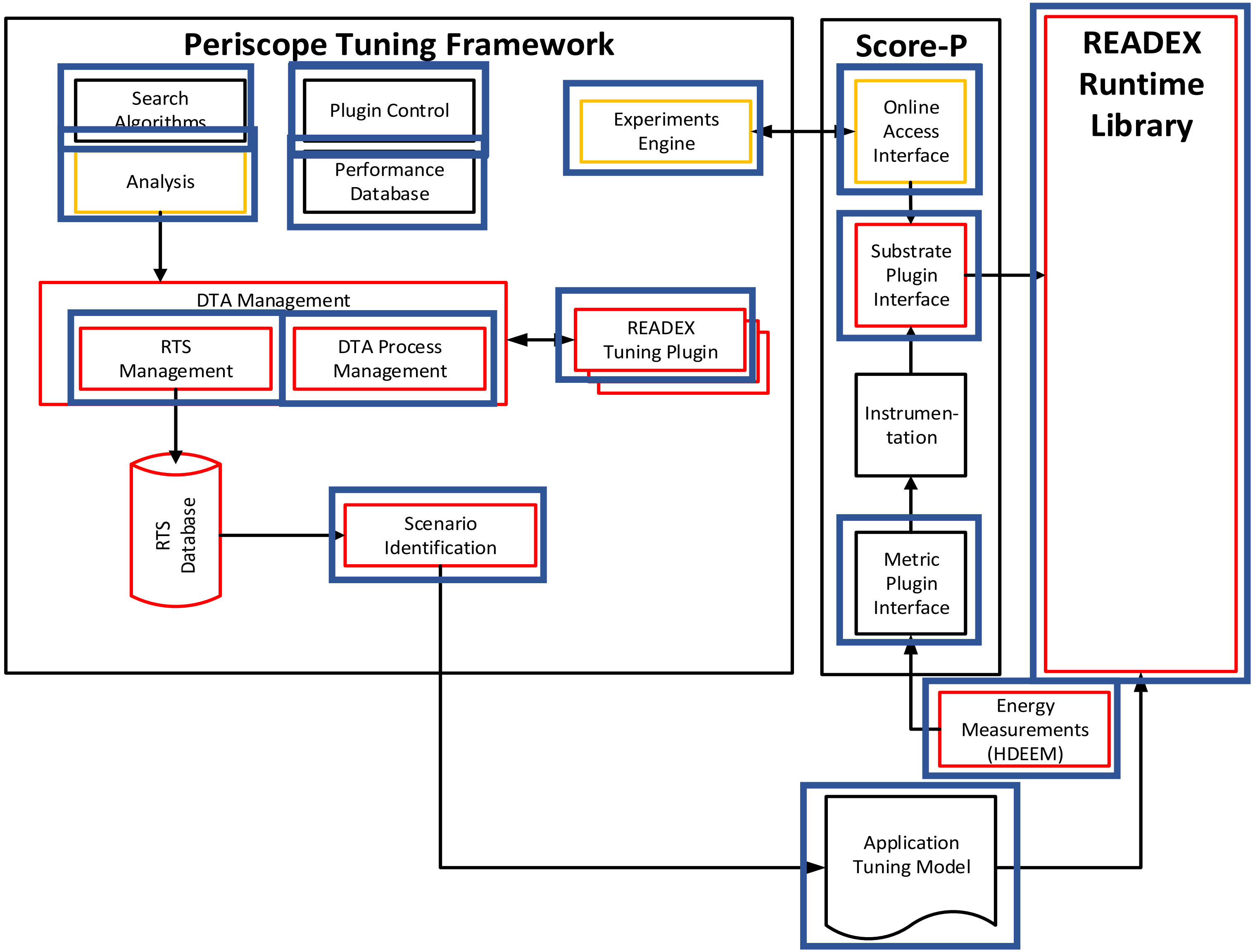
Readex_dyn_detect



rts detection



Pre-Computation of Configurations



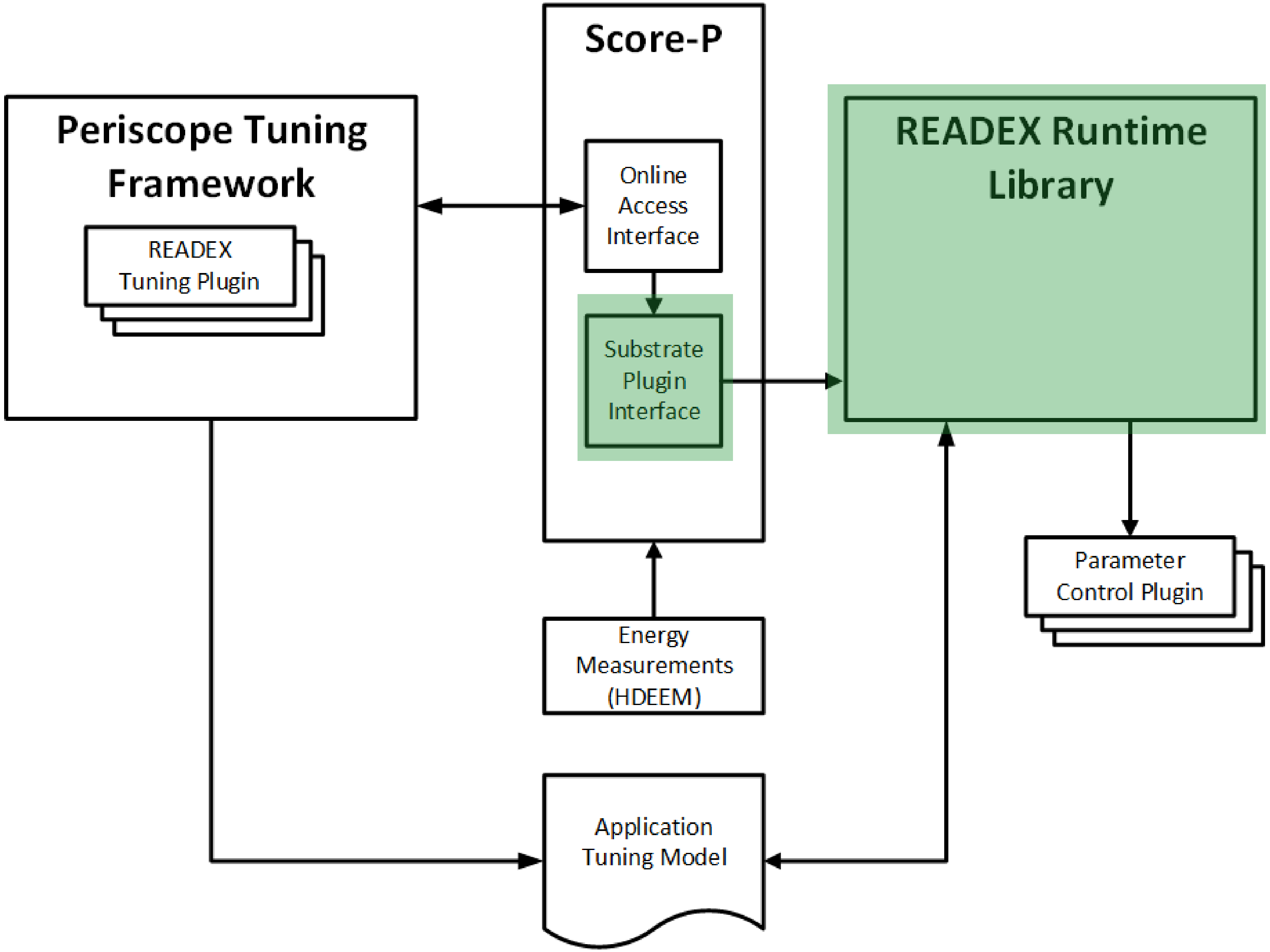
Runtime Tuning

READEX READEX EAB Meeting

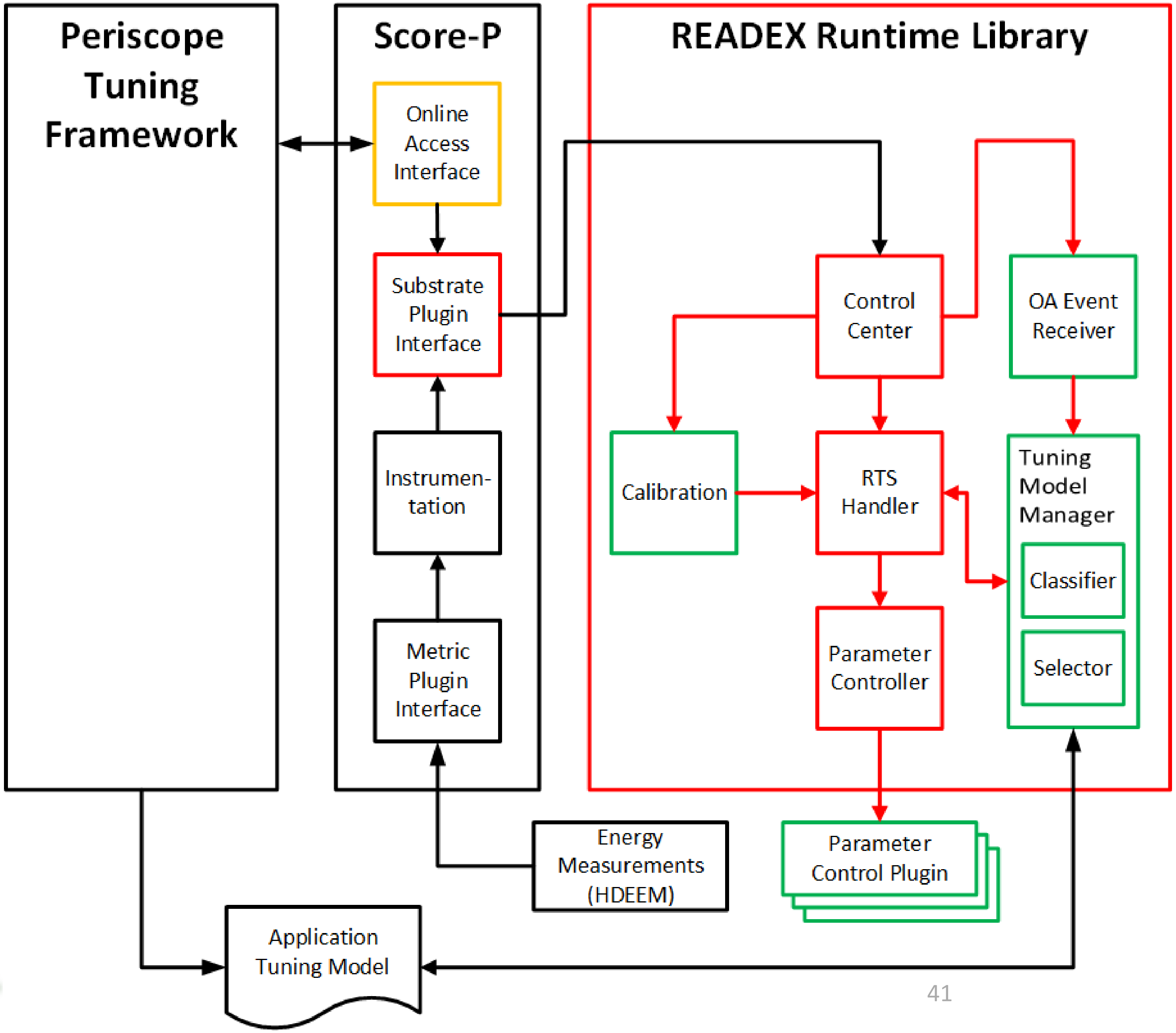
Robert Schöne – TUD

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

WP3 Implementation



Task 3.1: Scalable Runtime Library Architecture … (2)

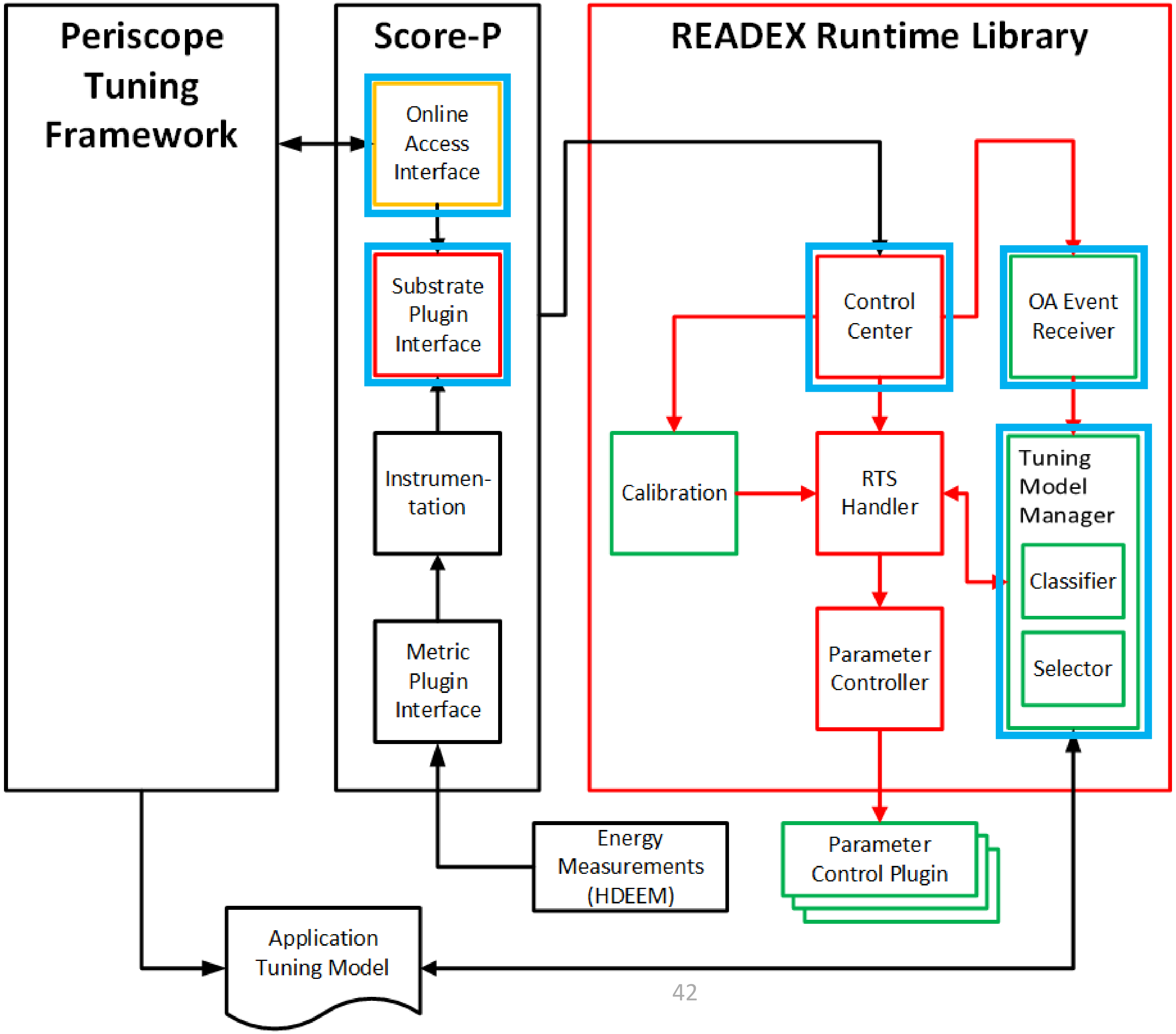


READEX
Runtime Exploitation of Application Dynamism for Energy-efficient eXascale computing

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

Task 3.2: Runtime Scenario Detection … (2)

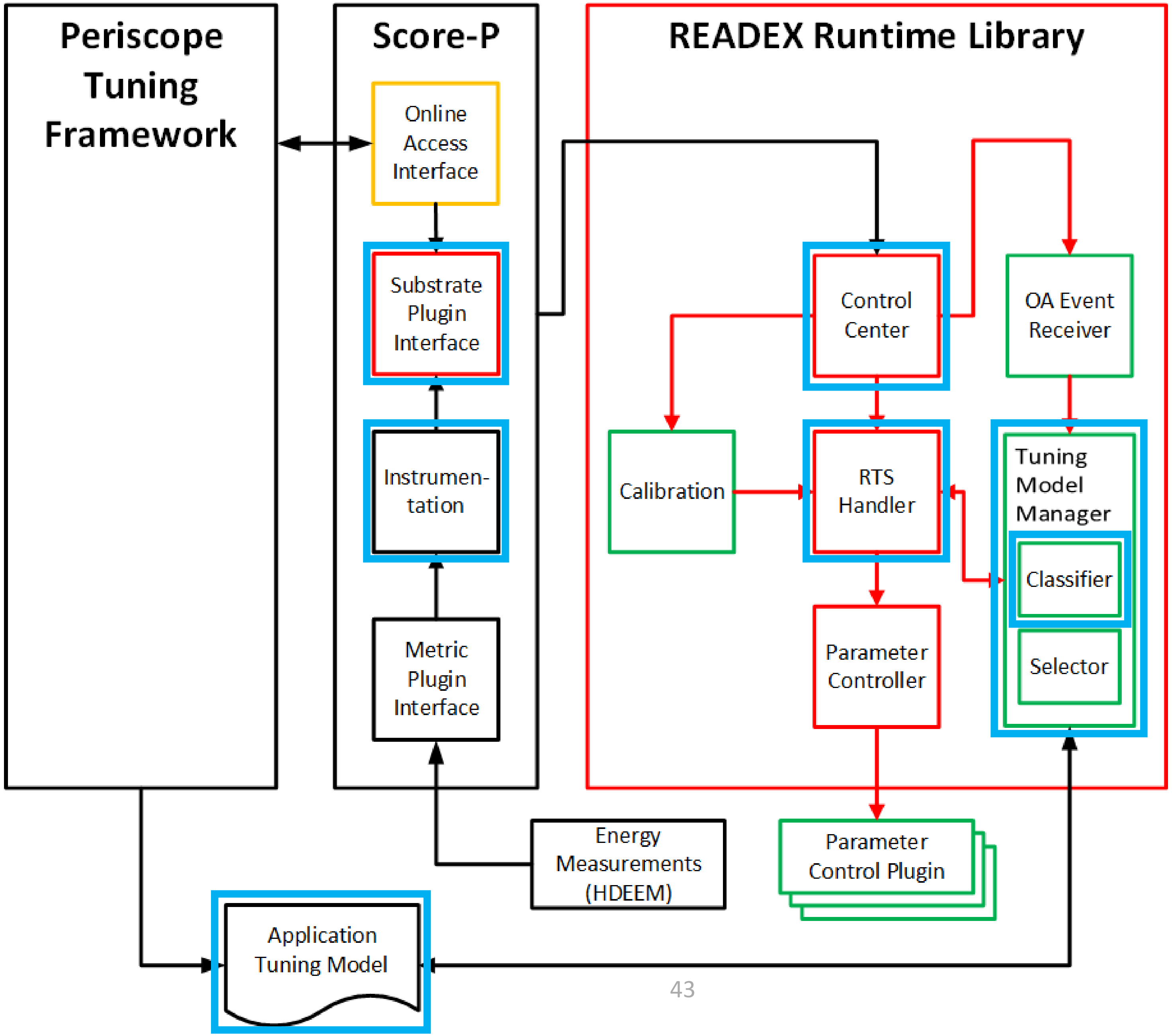
- During DTA
- Tuning request from PTF as JSON string
- Parsed by OA Event Receiver
- Stored by TMM



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

Task 3.2: Runtime Scenario Detection … (3)

- During RAT
- Identifier value detection
- Scenario classification

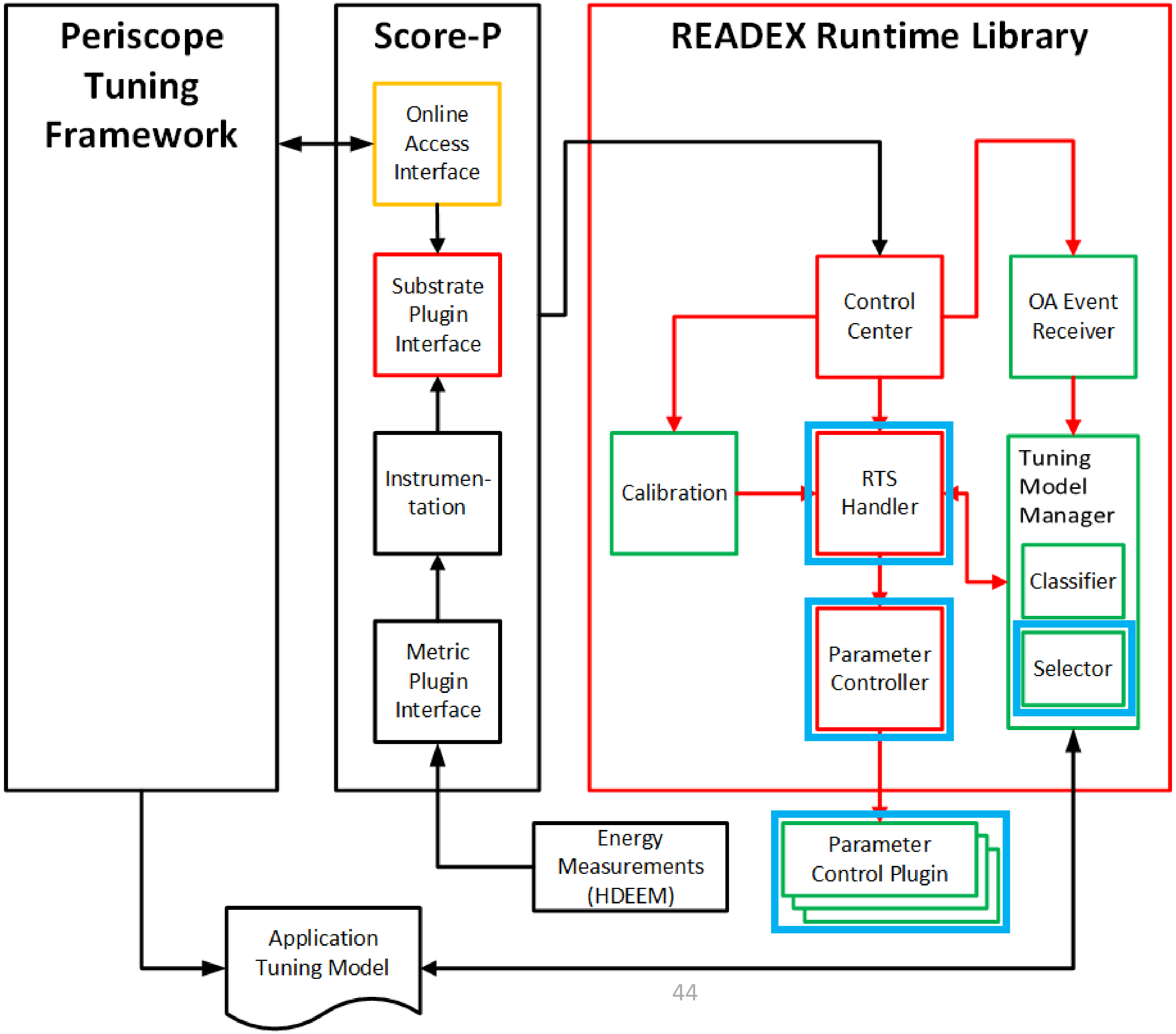


READEX
Runtime Exploitation of Application Dynamism for Energy-efficient eXascale computing

Task 3.4: Efficient Switching Decision making … (2)

Switching decision component

Manipulation of tuning parameters



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

Task 3.3: Runtime Calibration Mechanism … (2)

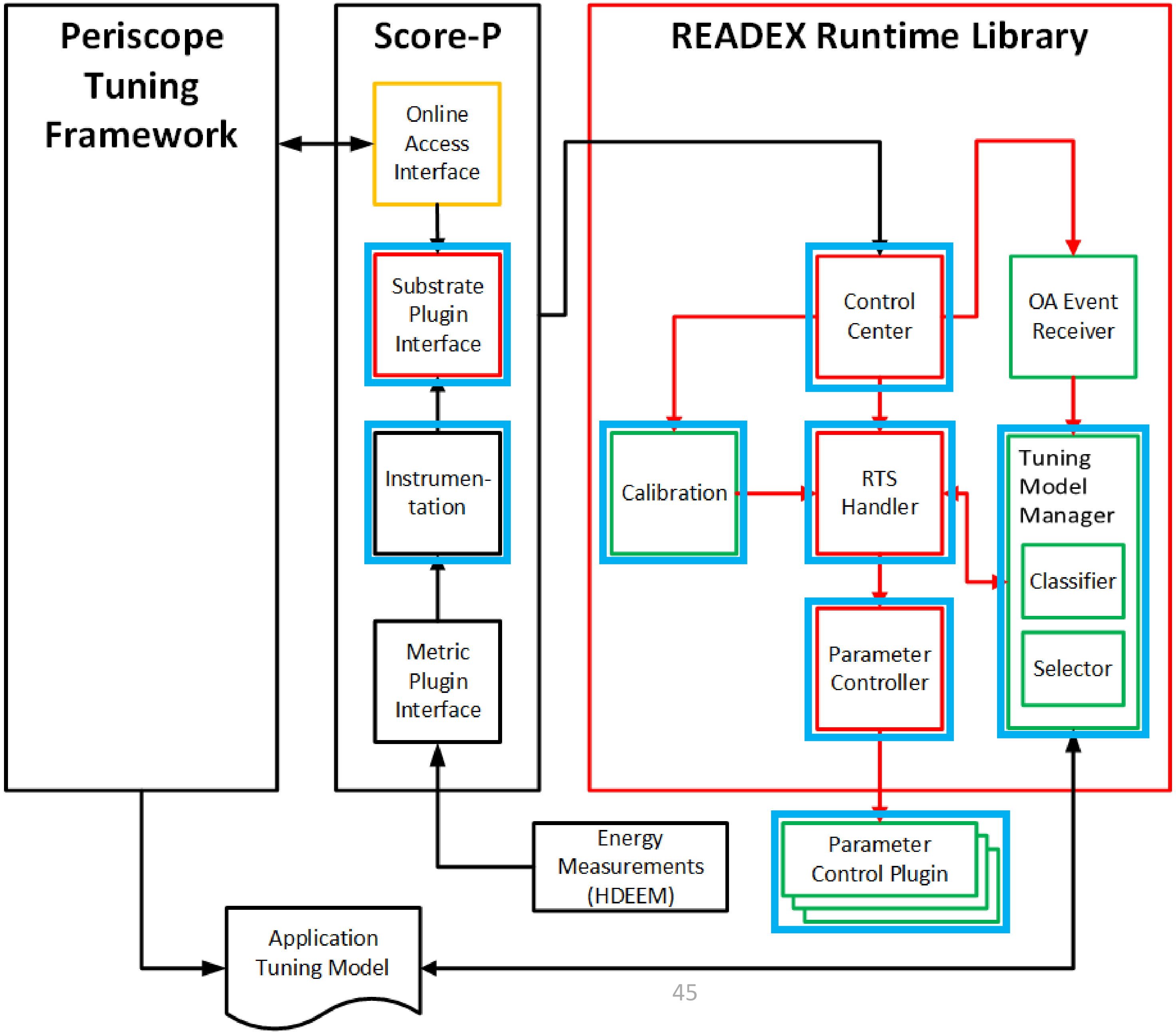
Identifier value detection

Unseen rts

Calibrate

Configure

Update ATM



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

Integration

READEX READEX EAB Meeting

Robert Schöne – TUD

Runtime Exploitation of Application Dynamism
for Energy-efficient eXascale computing

European Commission
Horizon 2020
European Union funding
for Research & Innovation

Initial Results

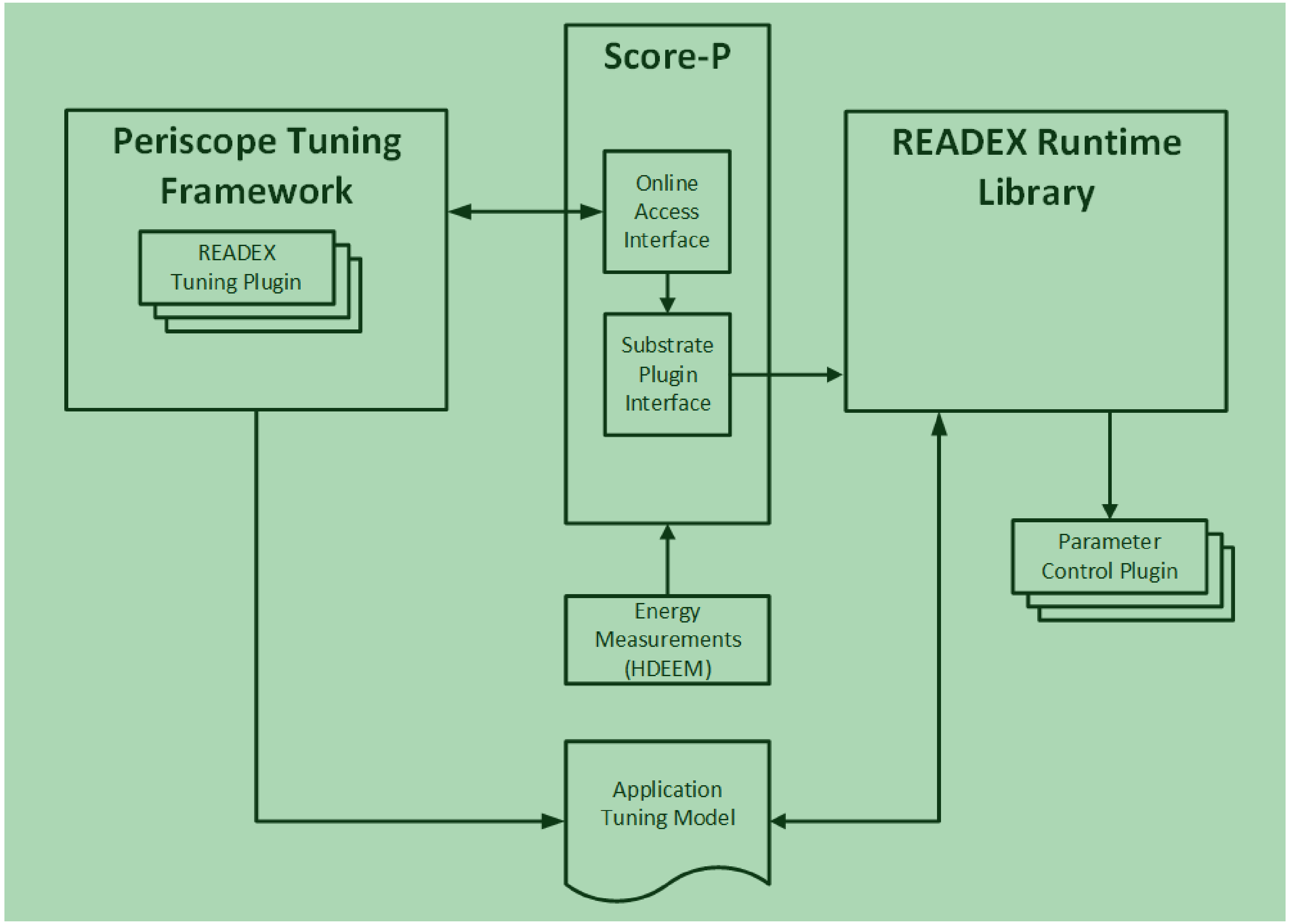
READEX READEX EAB Meeting

Robert Schöne – TUD

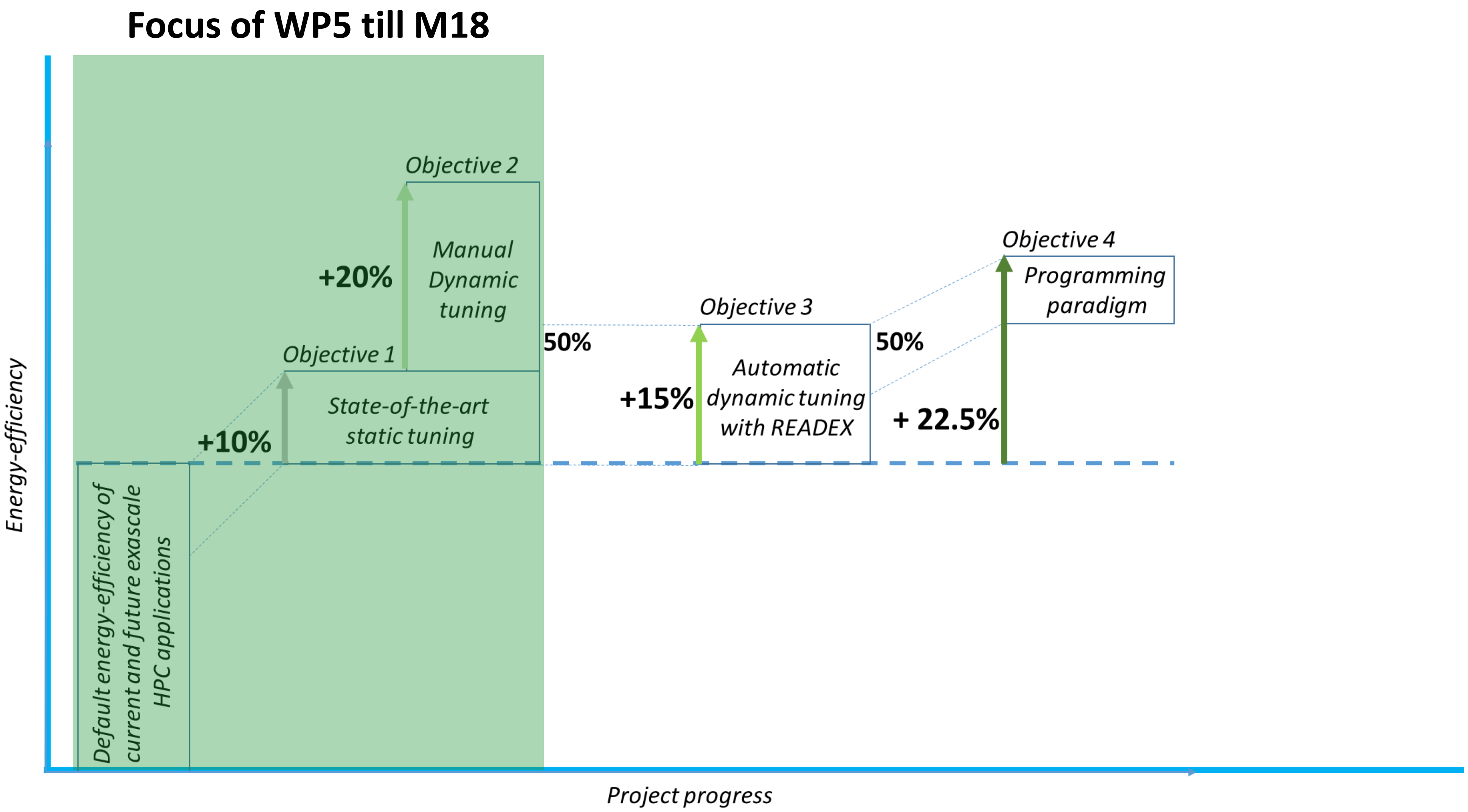
READEX
Runtime Exploitation of Application Dynamism for Energy-efficient eXascale computing

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

WP5 Implementation



WP5 Objectives



Task 5.1: Evaluating dynamism in HPC applications

Tools for Manual Evaluation

- MERIC tool
  - Based on manual annotation of significant regions
  - Search the space of tuning parameters to find optimal settings for each significant region
  - Support tool for energy measurements from HDEEM and RAPL
- **R**EADEX **A**pplication **D**ynamism **A**nalysis **R**eport (RADAR) & RADAR generator
  - Evaluates and reports the dynamic behaviour of the application

The dynamism observed in applications can be caused by the following factors:

- Floating point computations – variation in computational intensity (example - next slide)
- Memory read/write access patterns - variation in the sparsity of matrices in sparse linear algebra
- Inter-process communication patterns
- I/O operations performed during the application's execution
- Different inputs to regions in the application

Task 5.1: Evaluating dynamism in HPC applications

Goal: Investigate techniques to detect and evaluate dynamic behaviour in HPC applications

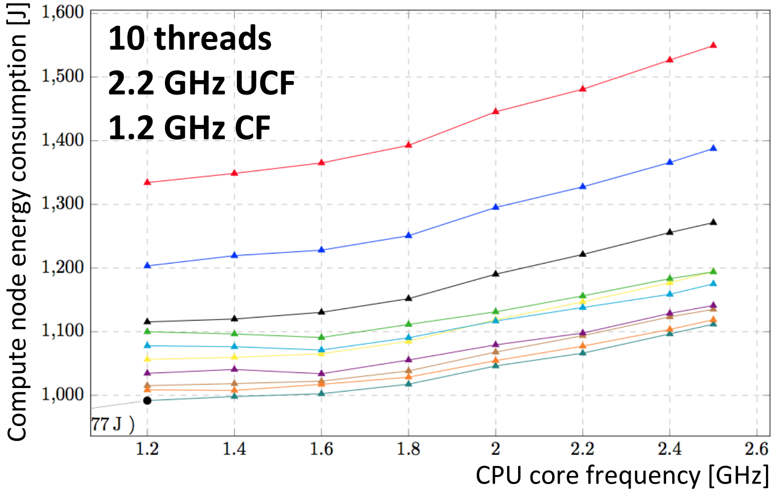
Hardware tuning parameters:

- core frequency (DVFS), uncore frequency (UFS)
- number of OpenMP threads

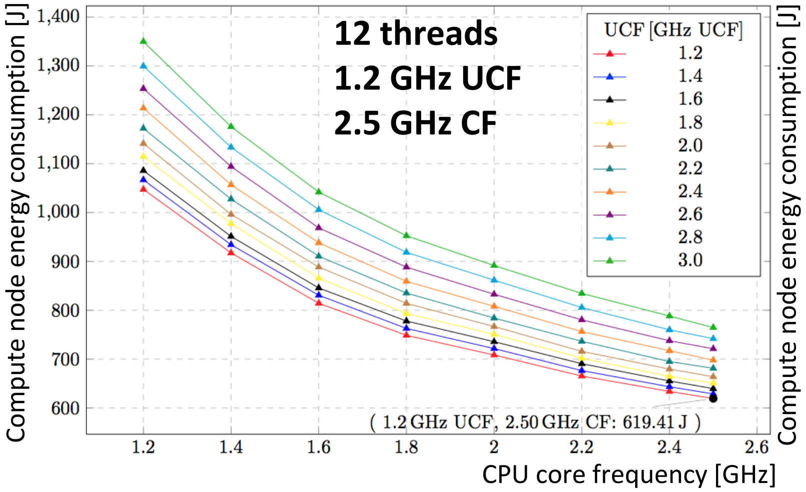
Example: Effect of Computational Intensity (CI)

Two kernels with 1:1 workload ratio	Energy consumption	Energy savings	
Default settings	2017J	-	-
Static optimal	1833J	179J	9%
Dynamic optimal	1612J	221J	12%
Total savings	-	400J	20%

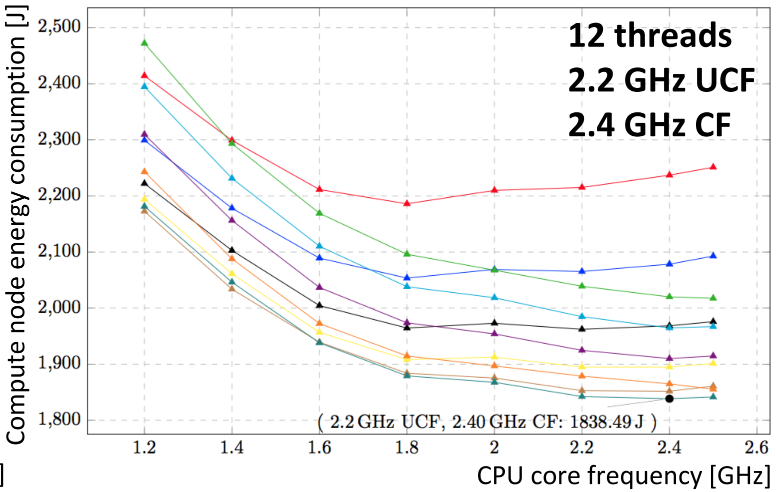
Low CI (DGEMV)



High CI



Static tuning for both kernels



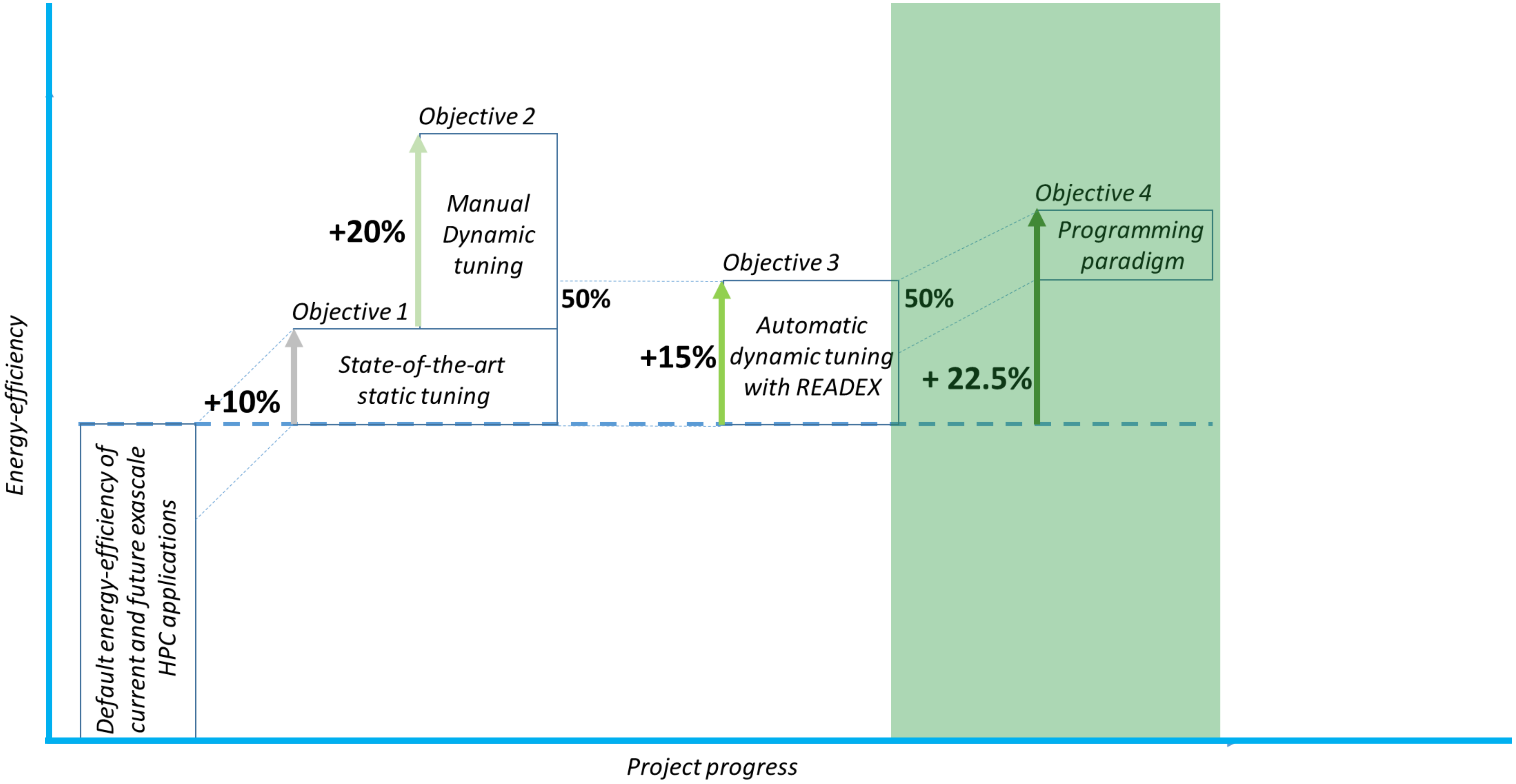
Note: runtime of both kernels was equal for default settings

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

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

Task 5.1: Evaluating dynamism in HPC applications

Investigation of application parameters



Task 5.2 Manually exploiting application dynamism

Approach and methodology for manual dynamism evaluation

1. **Identify significant regions** as the most time consuming (profiler – Alinea MAP)
2. **Manually annotate the significant regions** in the code – no compiler instrumentation overhead
3. **Apply tools developed in T5.1** to detect potential savings of an application
4. Using MERIC **run application for all possible configurations of tuning parameters** (parameters are set statically before each execution) and measure energy consumption
5. Using RADAR **find the best configuration for entire application** – static tuning potential
6. Using RADAR **find the best configuration for each significant region** – dynamic tuning potential
7. **Combine both static and dynamic savings** to define the potential for total energy savings

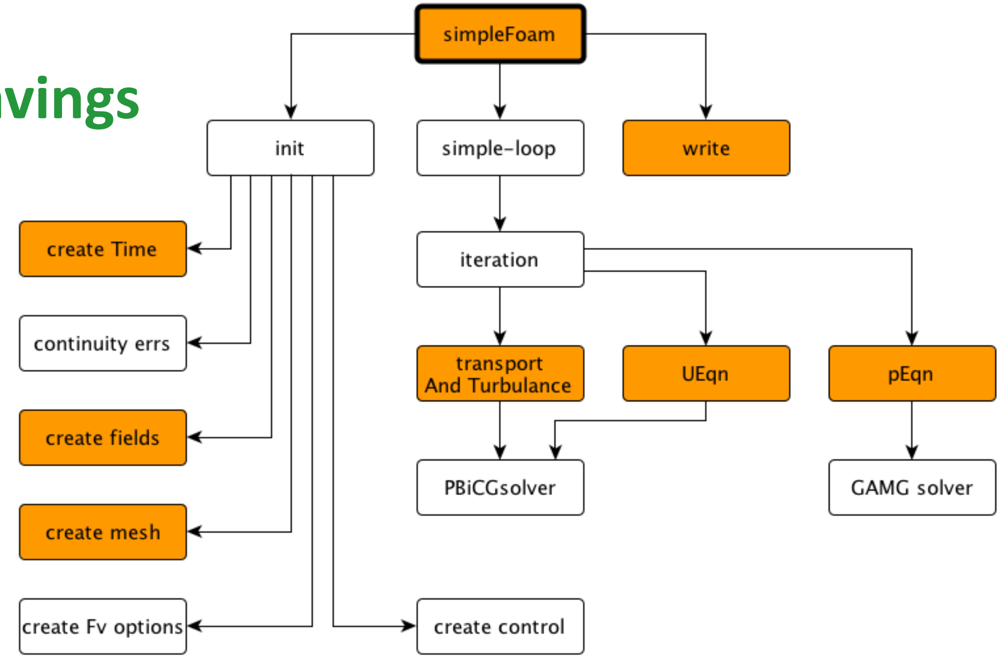
Task 5.2 Manually exploiting application dynamism

static **dynamic** **total**

OpenFOAM: 15.9% + 1.8% = 17.4% energy savings

- Computational fluid dynamics
- Finite volume + multigrid solver

Region	% of 1 phase	Best static configuration	Value	Best dynamic configuration	Value	Dynamic savings
init-createTime	0.03	2.0 GHz UCF, 1.6 GHz CF	3.35 J	1.4 GHz UCF, 1.4 GHz CF	2.64 J	0.71 J (21.06%)
init-createFields	4.28	2.0 GHz UCF, 1.6 GHz CF	506.91 J	2.4 GHz UCF, 2.0 GHz CF	474.80 J	32.11 J (6.33%)
init-createMesh	2.26	2.0 GHz UCF, 1.6 GHz CF	267.33 J	1.4 GHz UCF, 1.4 GHz CF	194.38 J	72.96 J (27.29%)
UEqn	40.71	2.0 GHz UCF, 1.6 GHz CF	4820.82 J	2.2 GHz UCF, 1.6 GHz CF	4810.03 J	10.80 J (0.22%)
pEqn	19.15	2.0 GHz UCF, 1.6 GHz CF	2268.19 J	2.0 GHz UCF, 1.6 GHz CF	2268.19 J	0.00 J (0.00%)
trans-portAnd-Turbulence	25.70	2.0 GHz UCF, 1.6 GHz CF	3042.91 J	2.0 GHz UCF, 1.6 GHz CF	3042.91 J	0.00 J (0.00%)
write	7.88	2.0 GHz UCF, 1.6 GHz CF	932.59 J	1.2 GHz UCF, 1.4 GHz CF	841.62 J	90.97 J (9.75%)
Total value for static tuning for significant regions			3.35 + 506.91 + 267.33 + 4820.82 + 2268.19 + 3042.91 + 932.59 = 11842.12 J			
Total savings for dynamic tuning for significant regions			0.71 + 32.11 + 72.96 + 10.80 + 0.00 + 0.00 + 90.97 = 207.54 J of 11842.12 J (1.75 %)			
Dynamic savings for application runtime			207.54 J of 11966.36 J (1.73 %)			
Total value after savings			11758.82 J (82.63 % of 14231.30 J)			



Uncore freq [GHz] / Core freq [GHz]	1.2	1.4	1.6	1.8	2.0	2.2	2.4	2.6	2.8	3.0
1.2	13,200.02	12,717.1	12,621.78	12,410.62	12,380.68	12,507.38	12,774.16	13,108.6	13,604.2	14,040.8
1.4	13,161.9	12,597.78	12,125.18	12,065.52	12,074.54	12,173.36	12,312.24	12,802.26	13,095.84	13,450.8
1.6	13,320.66	12,640.76	12,256.22	12,033.62	11,966.36	11,992.7	12,372.04	12,579.22	13,126.44	13,370.24
1.8	13,878.04	13,082.66	12,700.92	12,457.08	12,373.86	12,445.98	12,574.6	12,831.82	13,081.62	13,296.04
2	14,218.58	13,327.12	12,902.62	12,544.82	12,456.82	12,494.8	12,680.32	13,038.86	13,207.38	13,474.8
2.2	14,625.62	13,849.58	13,240.14	12,851	12,760.98	12,802.24	12,993.44	13,260.38	13,497.6	13,767.62
2.4	15,083.2	14,412.62	13,568.68	13,447.18	12,973.38	13,238.6	13,332.7	13,388.7	13,777.68	14,030.66
2.5	15,554.96	14,465.2	13,991	13,553.84	13,300.24	13,354.46	13,472.36	14,179.16	14,083.06	14,231.3

	Default settings	Default values	Best static configuration	Static Savings	Dynamic Savings
Energy consumption [J], Blade summary	3.0 GHz UCF, 2.5 GHz CF	14231.30 J	2.0 GHz UCF, 1.6 GHz CF	2264.94 J (15.92%)	207.54 J of 11966.36 J (1.73 %)
Runtime of function [s], Job info - hdeem	3.0 GHz UCF, 2.5 GHz CF	56.45 s	2.6 GHz UCF, 2.4 GHz CF	0.37 s (0.66%)	2.36 s of 56.08 s (4.20 %)

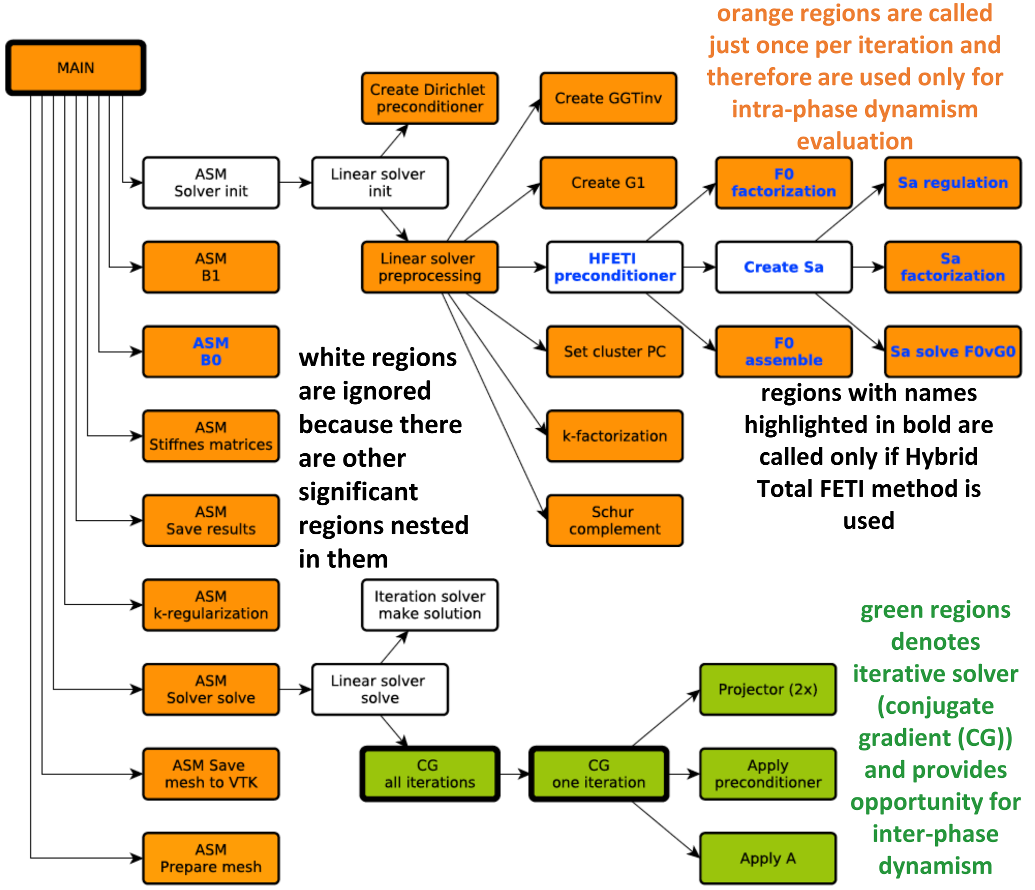
Task 5.2 Manually exploiting application dynamism

static dynamic total

ESPRESO: 12.3% + 9.1% = 20.3%

- Structural mechanics code
- Finite element + sparse FETI solver

Region	% of 1 phase	Best static configuration	Value	Best dynamic configuration	Value	Dynamic savings
Assembler–AssembleStiffMat	14.32	18 threads, 1.8 GHz UCF, 2.5 GHz CF	733.73 J	20 threads, 2.0 GHz UCF, 2.5 GHz CF	731.22 J	2.51 J (0.34%)
Assembler–Assemble-B1	2.23	18 threads, 1.8 GHz UCF, 2.5 GHz CF	114.30 J	2 threads, 2.2 GHz UCF, 2.5 GHz CF	94.15 J	20.15 J (17.63%)
Cluster–CreateF0-FactF0	0.17	18 threads, 1.8 GHz UCF, 2.5 GHz CF	8.71 J	6 threads, 1.6 GHz UCF, 2.5 GHz CF	6.90 J	1.80 J (20.73%)
Assembler–SaveResults	3.10	18 threads, 1.8 GHz UCF, 2.5 GHz CF	158.81 J	2 threads, 1.2 GHz UCF, 2.5 GHz CF	147.66 J	11.16 J (7.03%)
			…			
Cluster–CreateSa-SaReg	0.17	18 threads, 1.8 GHz UCF, 2.5 GHz CF	8.59 J	8 threads, 2.0 GHz UCF, 2.5 GHz CF	7.03 J	1.56 J (18.15%)
Total value for static tuning for significant regions			733.73 + 114.30 + 8.71 + 158.81 + 278.39 + 113.87 + 14.23 + 658.07 + 325.69 + 99.93 + 74.70 + 641.88 + 1578.06 + 13.28 + 24.20 + 278.22 + 8.59 = 5124.66 J			
Total savings for dynamic tuning for significant regions			2.51 + 20.15 + 1.80 + 11.16 + 47.01 + 16.41 + 5.31 + 28.45 + 29.03 + 19.08 + 0.16 + 2.49 + 288.21 + 0.77 + 1.88 + 23.24 + 1.56 = 499.22 J of 5124.66 J (9.74 %)			
Dynamic savings for application runtime			499.22 J of 5493.55 J (9.09 %)			
Total value after savings			4994.33 J (79.72 % of 6265.18 J)			



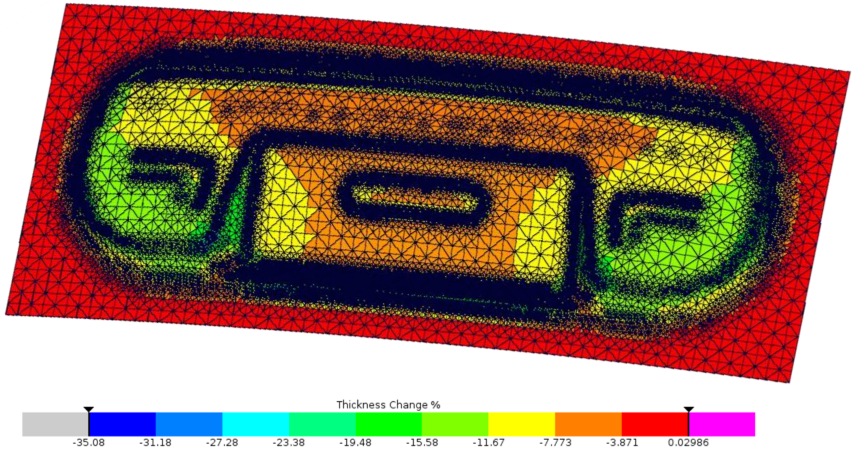
	Default settings	Default values	Best static configuration	Static Savings	Dynamic Savings
Energy consumption [J] , Blade summary	24 threads, 3.0 GHz UCF, 2.5 GHz CF	6265.18 J	18 threads, 1.8 GHz UCF, 2.5 GHz CF	771.63 J (12.32%)	499.2 J of 5493.6 J (9.09 %)
Runtime of function [s], Job info - hdeem	24 threads, 3.0 GHz UCF, 2.5 GHz CF	29.55 s	22 threads, 3.0 GHz UCF, 2.5 GHz CF	0.01 s (0.04%)	0.82 s of 29.54 s (2.76 %)

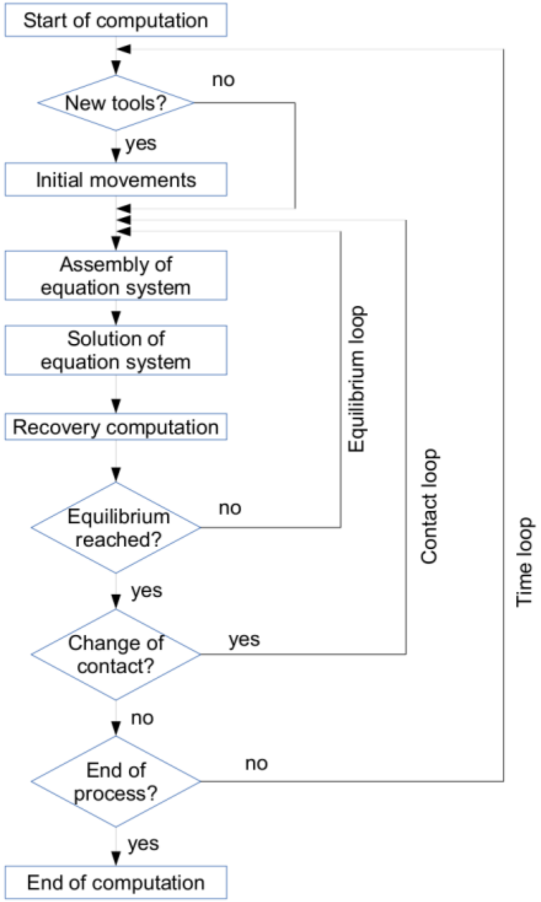
Task 5.2 Manually exploiting application dynamism

static **dynamic** **total**

Indeed: 17.6% + 3.9% = 20.9%

- Structural mechanics code (Simulation of sheet metal forming)
- Finite Element solver



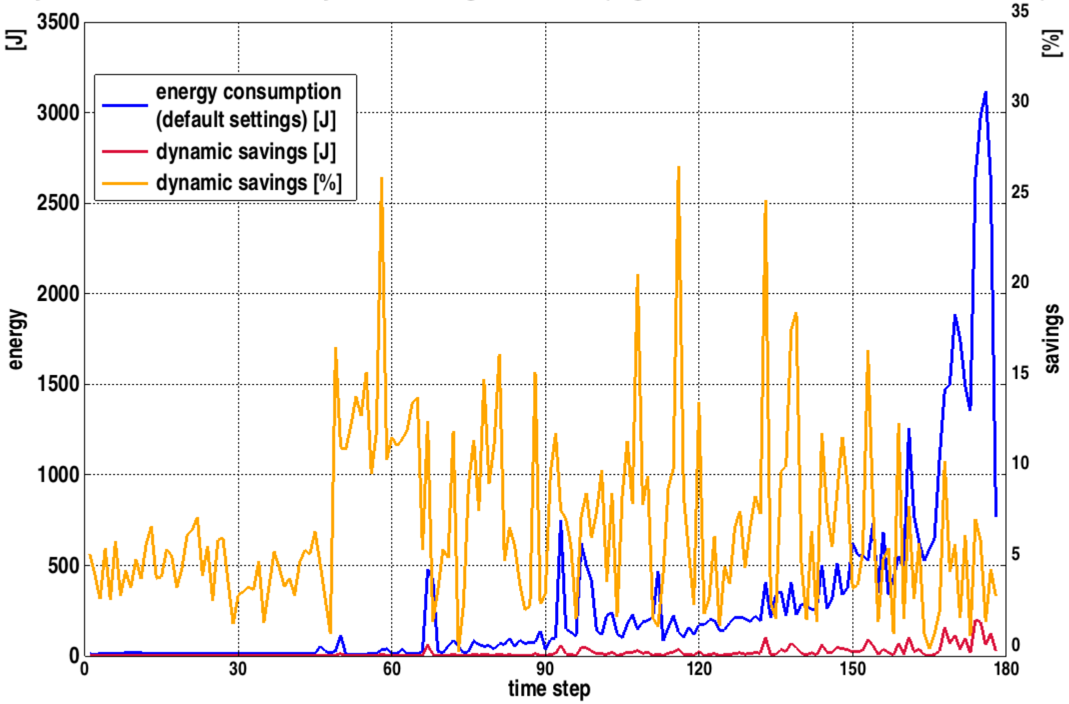


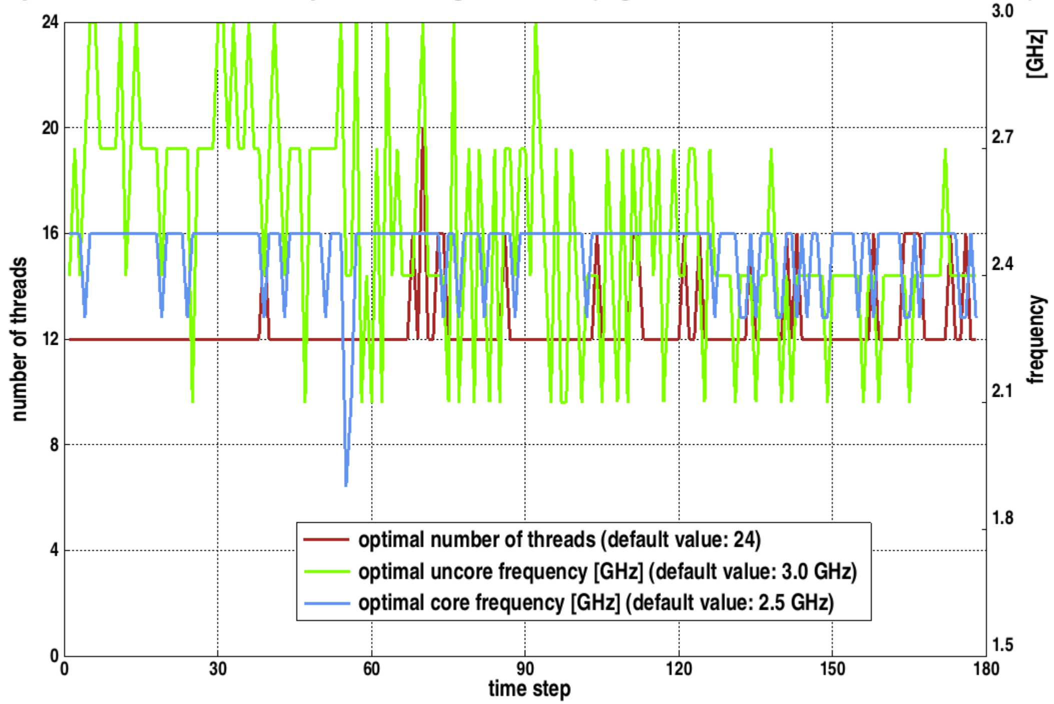
Region	% of 1 phase	Best static configuration	Value	Best dynamic configuration	Value	Dynamic savings
asm-matrix	17.07	16 th, 2.1 GHz UCF, 2.5 GHz CF	119.19 J	24 th, 2.1 GHz UCF, 2.3 GHz CF	113.18 J	6.00 J (5.04%)
output	32.95	16 th, 2.1 GHz UCF, 2.5 GHz CF	230.05 J	16 th, 1.2 GHz UCF, 2.5 GHz CF	226.61 J	3.43 J (1.49%)
recovery	9.16	16 th, 2.1 GHz UCF, 2.5 GHz CF	63.95 J	24 th, 2.1 GHz UCF, 2.3 GHz CF	61.88 J	2.06 J (3.23%)
solver	40.81	16 th, 2.1 GHz UCF, 2.5 GHz CF	284.91 J	12 th, 2.4 GHz UCF, 2.5 GHz CF	269.36 J	15.55 J (5.46%)
Total value for static tuning for significant regions			119.19 + 230.05 + 63.95 + 284.91 = 698.09 J			
Total savings for dynamic tuning for significant regions			6.00 + 3.43 + 2.06 + 15.55 = 27.05 J of 698.09 J (3.87 %)			

Uncore freq [GHz UCF] / Frequency [GHz CF]	1.2	1.5	1.8	2.1	2.4	2.7	3.0
1.3	200,368	195,085	196,079	195,955	200,174	207,545	215,342
1.5	180,762	175,571	174,333	178,460	181,680	188,976	198,247
1.7	171,434	170,215	165,811	175,032	180,476	183,092	184,898
1.9	170,399	162,346	158,271	160,613	162,739	167,129	172,210
2.1	165,392	156,934	154,319	152,769	155,148	158,732	163,338
2.3	163,137	153,630	150,165	149,788	150,529	153,942	157,621
2.5	161,606	153,326	149,027	147,150	147,886	150,857	153,475

Task 5.2 Manually exploiting application dynamism

Indeed: Specific look at solver region – intra-phase dynamism





Total sum of values from dynamic savings from all phases

Energy consumption: from 50.7kJ to 47.6kJ

6.3% energy savings

M18 - Deliverable 5.1: Report on application dynamism

Evaluation of HPC codes ranging from basic kernels to very complex applications

Key results

- Highly optimized applications tend to provide higher static and lower dynamic savings
- Complex applications, such as ESPRESO, which contains variation on workload (not only compute) shows opportunity for dynamic tuning

Application	Static savings [%]	Dynam. savings [%]	Total Savings [%]
Parallel OpenMP I/O	56	—	56
Dense BLAS - DGEMV - without NUMA	5.6	—	5.6
Dense BLAS - DGEMM - without NUMA	10.4	—	10.4
Compute only kernel	12.8	—	12.8
Sparse BLAS Routines - without NUMA	3.1-12.3	—	3.1 – 12.3
Sparse BLAS Routines - with NUMA	4.2-66.2	—	4.2 – 66.2
ProxyApps 1 - AMG2013, configuration 1	6.53	2.89	9.23
ProxyApps 1 - AMG2013, configuration 2	25.66	2.80	27.74
ProxyApps 2 - Kripke, configuration 1	28.16	1.56	29.28
ProxyApps 2 - Kripke, configuration 2	12.63	7.04	18.78
ProxyApps 3 - LULESH, configuration 1	28.58	0.55	28.88
ProxyApps 3 - LULESH, configuration 2	25.81	1.23	26.72
ProxyApps 4 - MCB, configuration 1	4.13	1.42	5.51
ProxyApps 4 - MCB, configuration 2	3.40	4.18	7.44
ESPRESO - configuration 0	5.6	8.7	14.3
ESPRESO - configuration 1	12.3	9.1	21.4
ESPRESO - configuration 2	7.8	4.7	12.5
ESPRESO - configuration 3	7.8	5.4	13.1
OpenFOAM (Motorbike benchmark)	15.9	1.8	17.7
Indeed	17.6	to be evaluated	17.6
MiniMD	21.92	0.00	21.92

WP5: New applications

static dynamic total

BEM4I - Boundary element method: 12.4% + 11.7% = 22.6%

- High CI for matrix assembling & Low CI for linear solver

Overall application evaluation

	Default settings	Default values	Best static configuration	Static Savings	Dynamic Savings
Energy consumption [J] (Samples), Blade summary	24 threads, 3.0 GHz UCF, 2.5 GHz CF	5011.13 J	24 threads, 1.6 GHz UCF, 2.1 GHz CF	619.81 J (12.37%)	512.81 J of 4391.32 J (11.68 %)
Runtime of function [s], Job info - hdeem	24 threads, 3.0 GHz UCF, 2.5 GHz CF	17.87 s	24 threads, 3.0 GHz UCF, 2.5 GHz CF	0.00 s (0.00%)	1.37 s of 17.87 s (7.64 %)

Region	% of 1 phase	Best static configuration	Value	Best dynamic configuration	Value	Dynamic savings
gmres_solve	16.82	24 threads, 1.6 GHz UCF, 2.1 GHz CF	737J	12 threads, 2.2 GHz UCF, 1.3 GHz CF	326.87 J	410.15 J (55.65%)
print_vtu	0.51	24 threads, 1.6 GHz UCF, 2.1 GHz CF	22.37 J	12 threads, 1.2 GHz UCF, 2.3 GHz CF	16.28 J	6.09 J (27.24%)
assemble_K	40.81	24 threads, 1.6 GHz UCF, 2.1 GHz CF	1788J	24 threads, 1.2 GHz UCF, 2.3 GHz CF	1747.57 J	40.59 J (2.27%)
assemble_V	41.85	24 threads, 1.6 GHz UCF, 2.1 GHz CF	1833J	24 threads, 1.2 GHz UCF, 2.3 GHz CF	1777.74 J	55.98 J (3.05%)
Dynamic savings for application runtime			512.81 J of 4391.32 J (11.68 %)			
Total value after savings			3878.51 J (77.40 % of 5011.13 J)			

Low CI - 17%

High CI is dominant 83% of RT