Energy Efficiency Tuning: From Autotune to READEX

Michael Gerndt

Technische Universität München

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

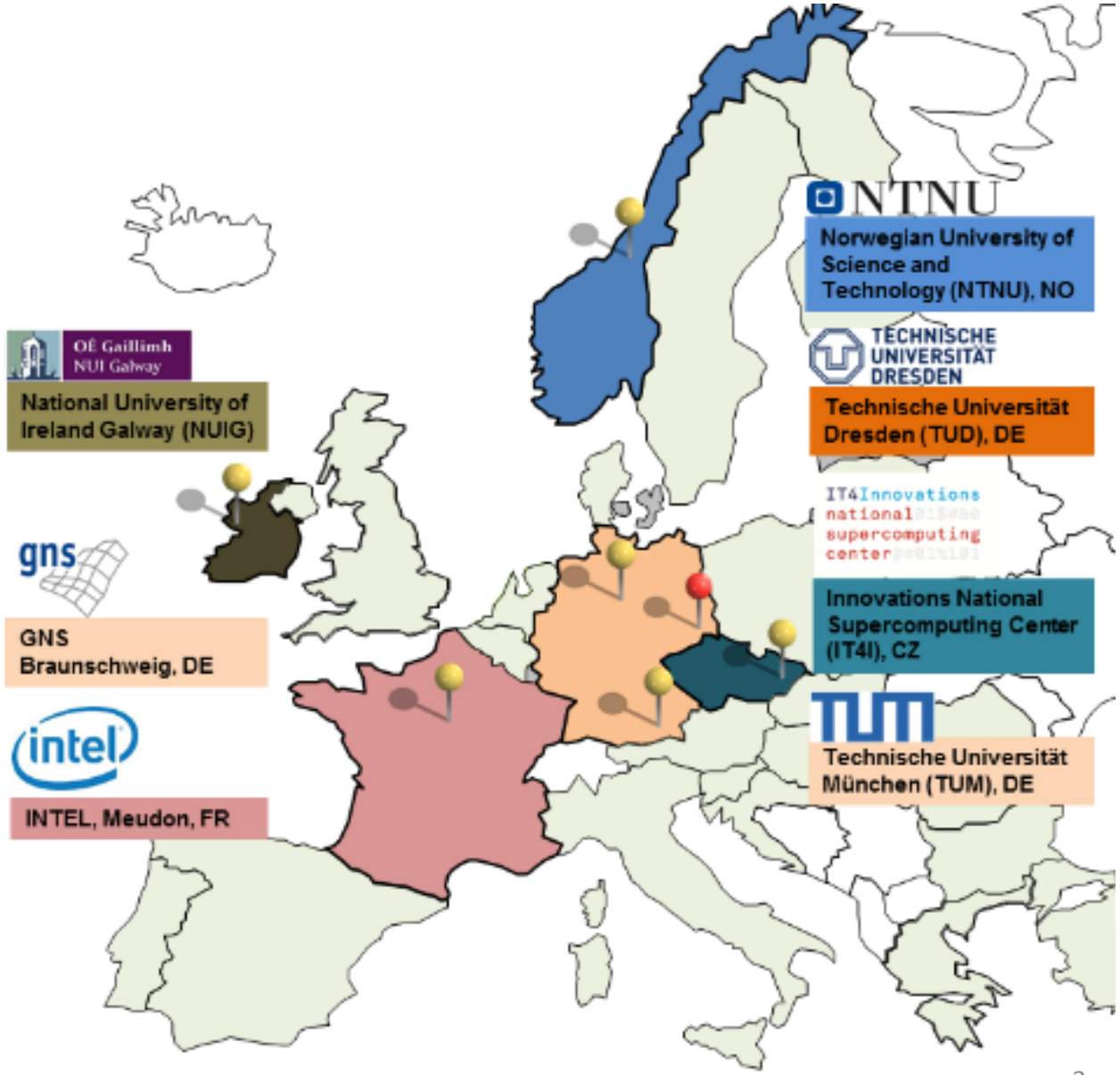
- Duration:

  3 years

- Funding:

  European Commission Horizon 2020 grant agreement 671657

Project partners



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

European Commission
Horizon 2020
European Union funding for Research & Innovation

Motivation

Challenges

- Energy consumption
- Extreme scale
- Dynamism

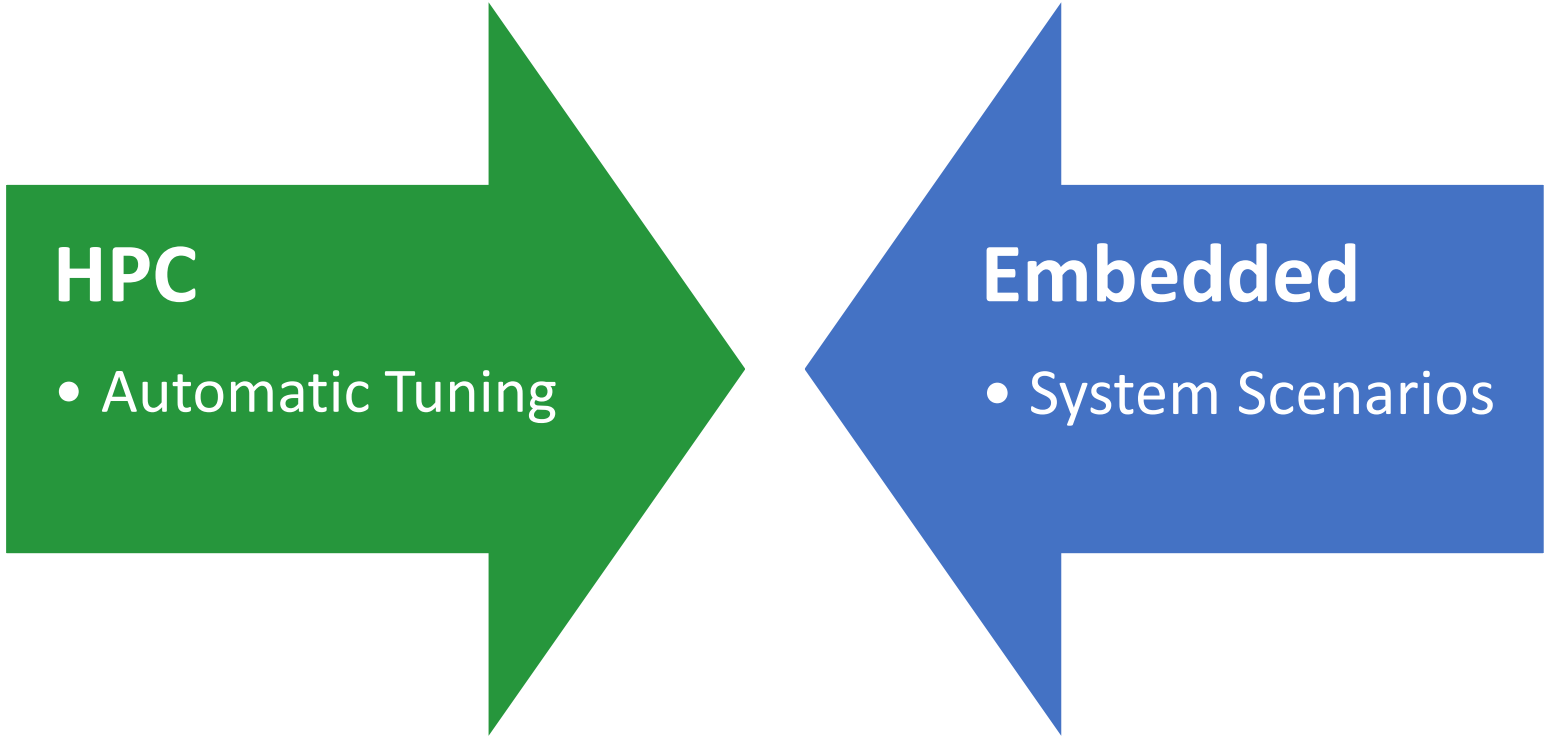
Problems

- Awareness
- Ability
- Effort

Solution

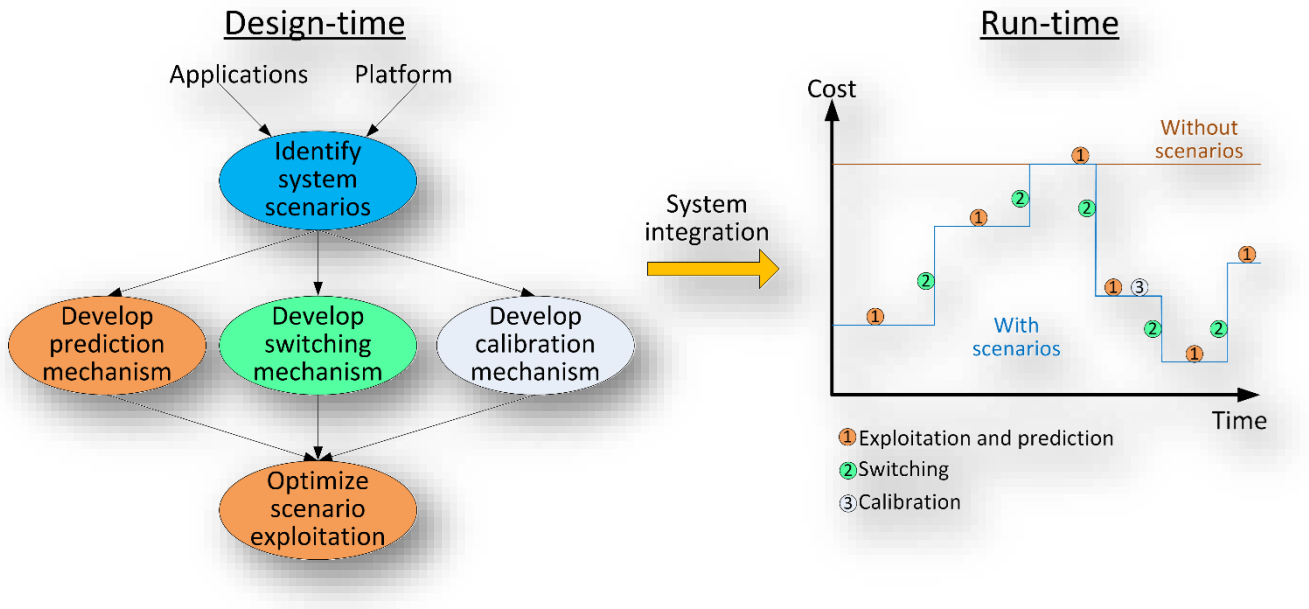
- Dynamism
- Automatic tuning
- Design-/Run-time

General idea



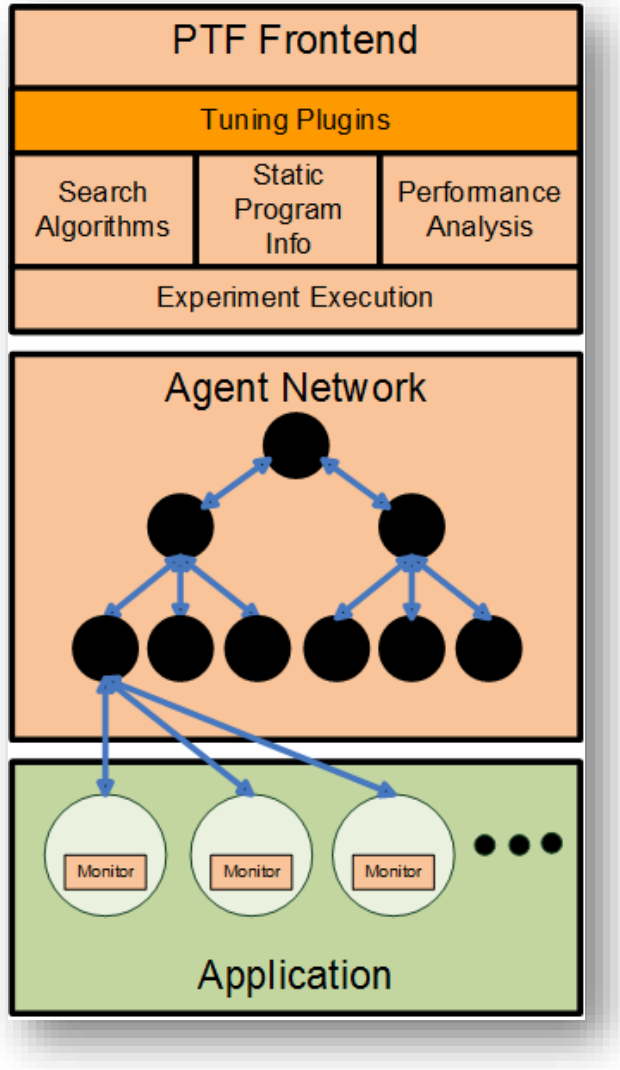
Systems scenarios

- **System Scenario based Methodology**
  - Formalism for dynamic auto-tuning in the embedded systems world
  - Detect and analyze dynamism in applications at design-time
  - Switch parameters at run-time based on detected scenarios



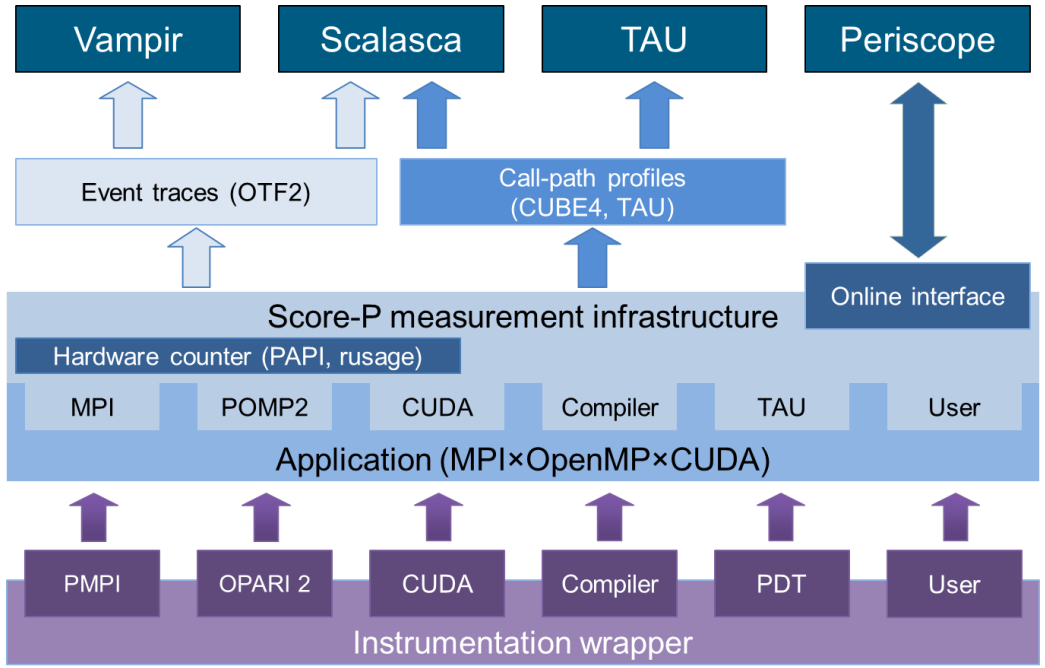
Periscope Tuning Framework

- 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

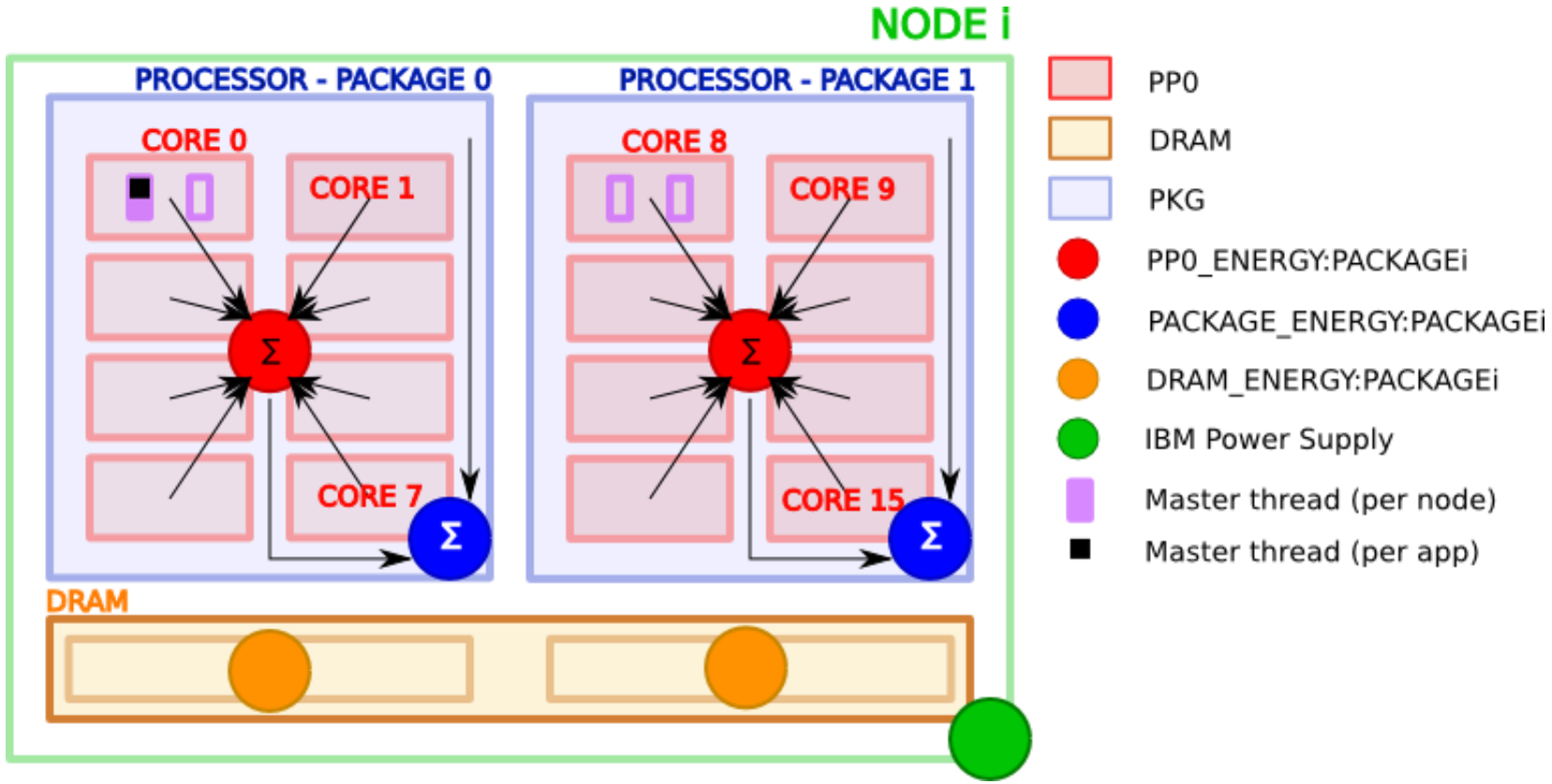


Score-P

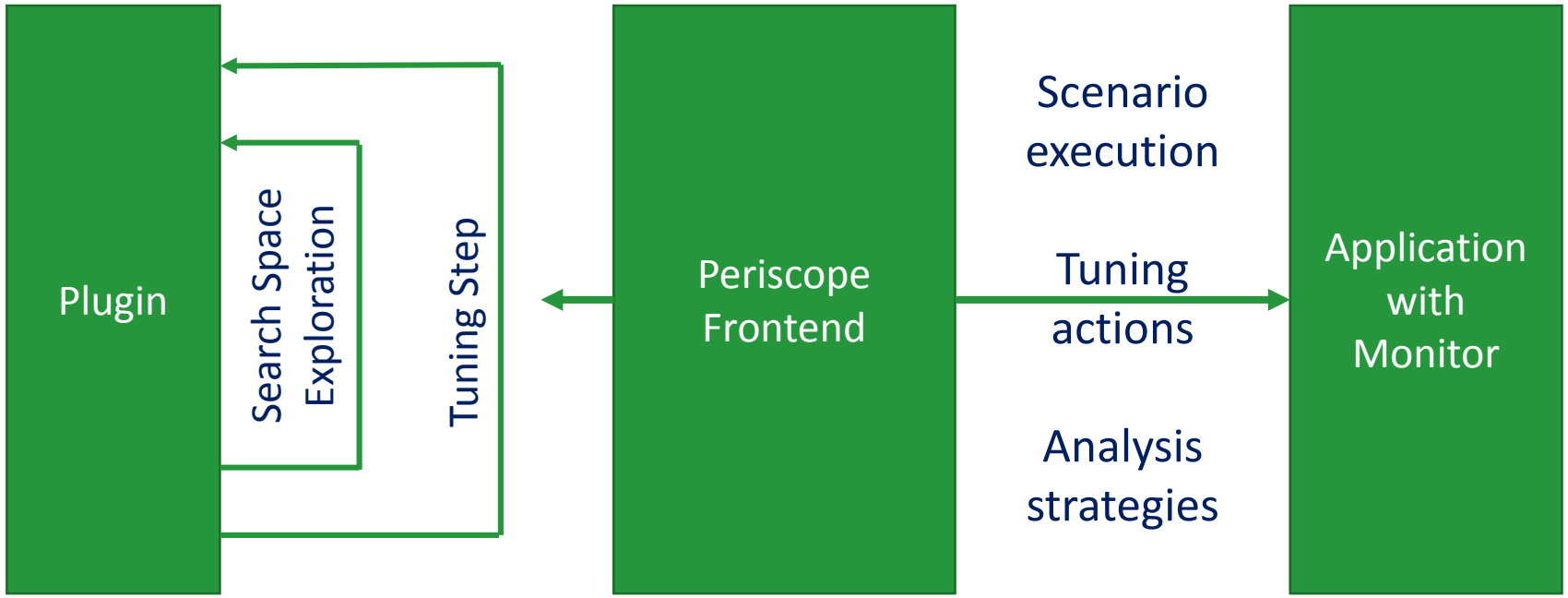
- **Scalable Performance Measurement Infrastructure for Parallel Codes**
  - Common instrumentation and measurement infrastructure



ENOPT library implemented by LRZ



Tuning Plugin Interface



Tuning Plugins

- MPI parameters
  - Eager Limit, Buffer space, collective algorithms
  - Application restart or MPIT Tools Interface
- DVFS
  - Frequency tuning for energy delay product
  - Model-based prediction of frequency
  - Region level tuning
- Parallelism capping
  - Thread number tuning for energy delay product
  - Exhaustive and curve fitting based prediction

European Commission
Horizon 2020
European Union funding for Research & Innovation

Tuning Plugins

- Master/worker
  - Partition factor and number of workers
  - Prediction through performance model based on data measured in pre-analysis
- Parallel Pattern
  - Tuning replication and buffers between pipeline stages
  - Based on component distribution via StarPU
- OpenCL tuning
  - Compiler flags for offline compilation
  - NDRange tuning

European Commission
Horizon 2020
European Union funding for Research & Innovation

Tuning Plugins

- MPI IO
  - Tuning data sieving and number of aggregators
  - Exhaustive and model based
- Compiler Flag Selection
  - Automatic recompilation and execution
  - Selective recompialtion based on pre-analysis
  - Exhaustive and individual search
  - Scenario analysis for significant routines
  - Combination with Pathway

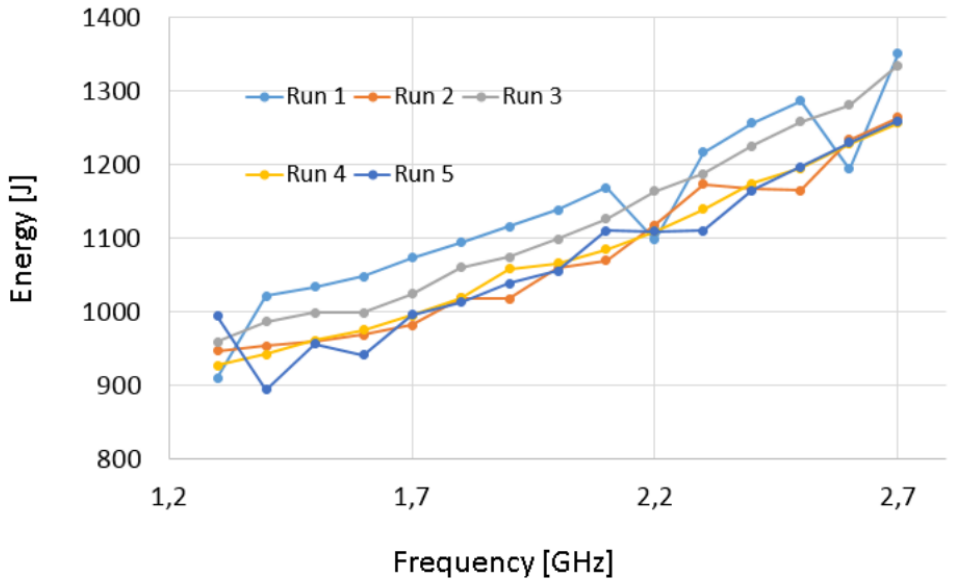
European Commission
Horizon 2020
European Union funding for Research & Innovation

Plugin Evaluation Status

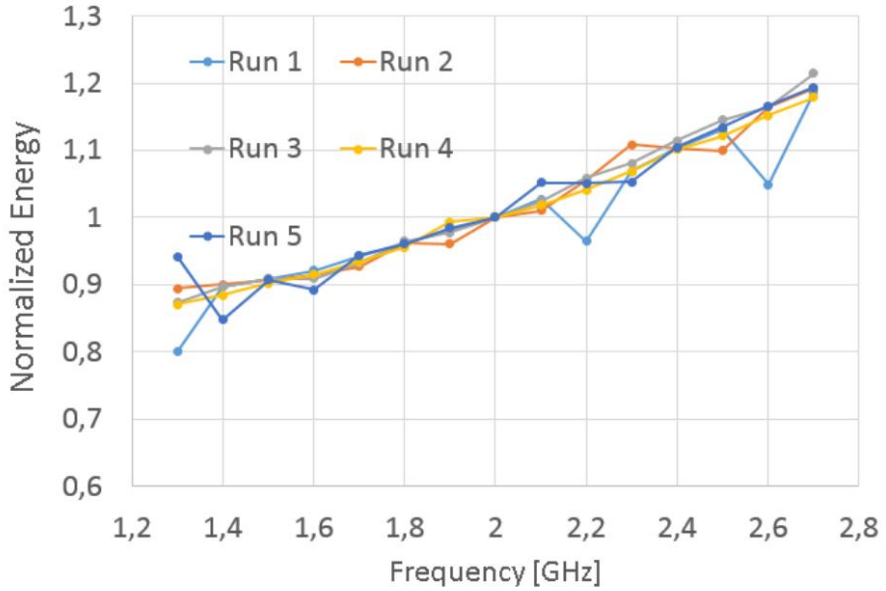
Application Name	CFS	DVFS	MPI Parameters	Patterns	Master Worker
APEX-MAP	✗	✓	✗	✗	✗
BLAST	✗	✗	✗	✗	✓
Convolution	✓	✗	✗	✗	✗
FaceDetect	✗	✗	✗	✓	✗
FSSIM	✓	✓	✓	✗	✗
HydroC	✓	✗	✗	✗	✗
NPB	✓	✓	✓	✗	✗
pmatmul	✓	✓	✓	✗	✗
SeisSol	✓	✓	✓	✗	✗
Sip	✓	✓	✗	✗	✗
S2F2M	✓	✗	✓	✗	✓
Model_primes	✓	✗	✓	✗	✗
nw	✗	✗	✗	✓	✗

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

Variation of Measurements

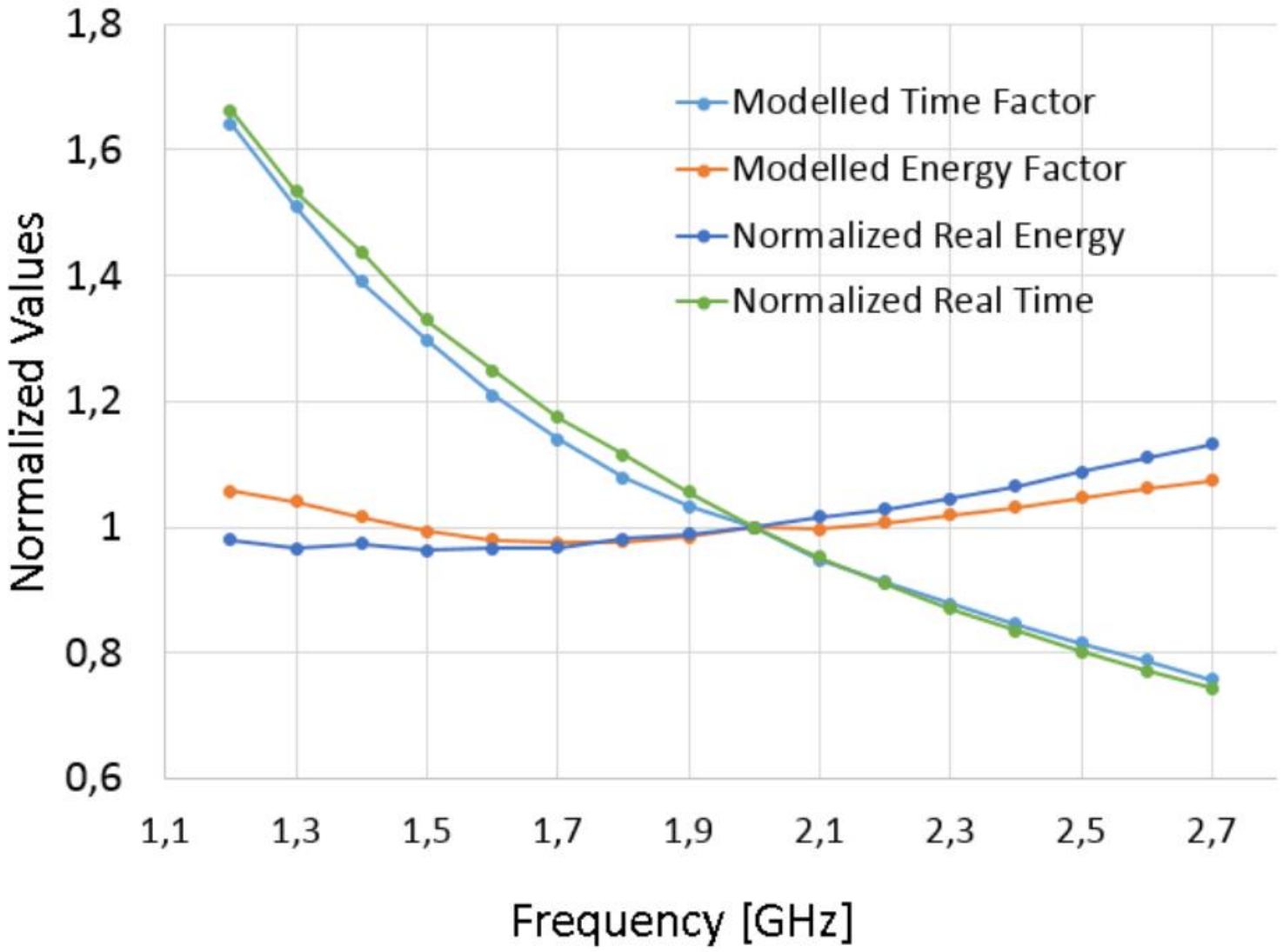


Energy consumption of the SeisSol application at different compute nodes.

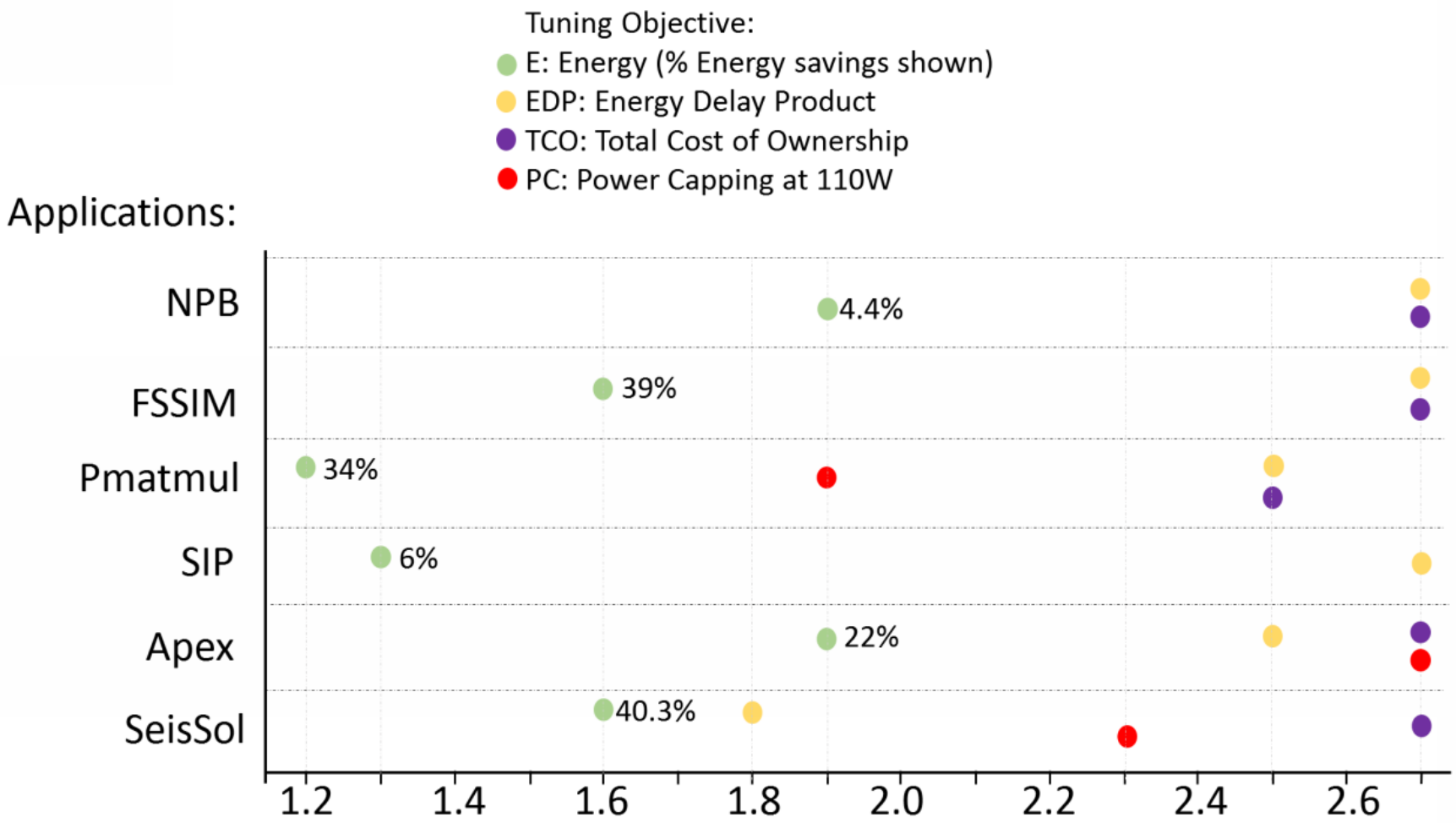


Normalized energy consumption of the SeisSol application at different compute

Predicted vs Measured Time for Seissol



Tuning with Persicope Tuning Framework



Dynamism

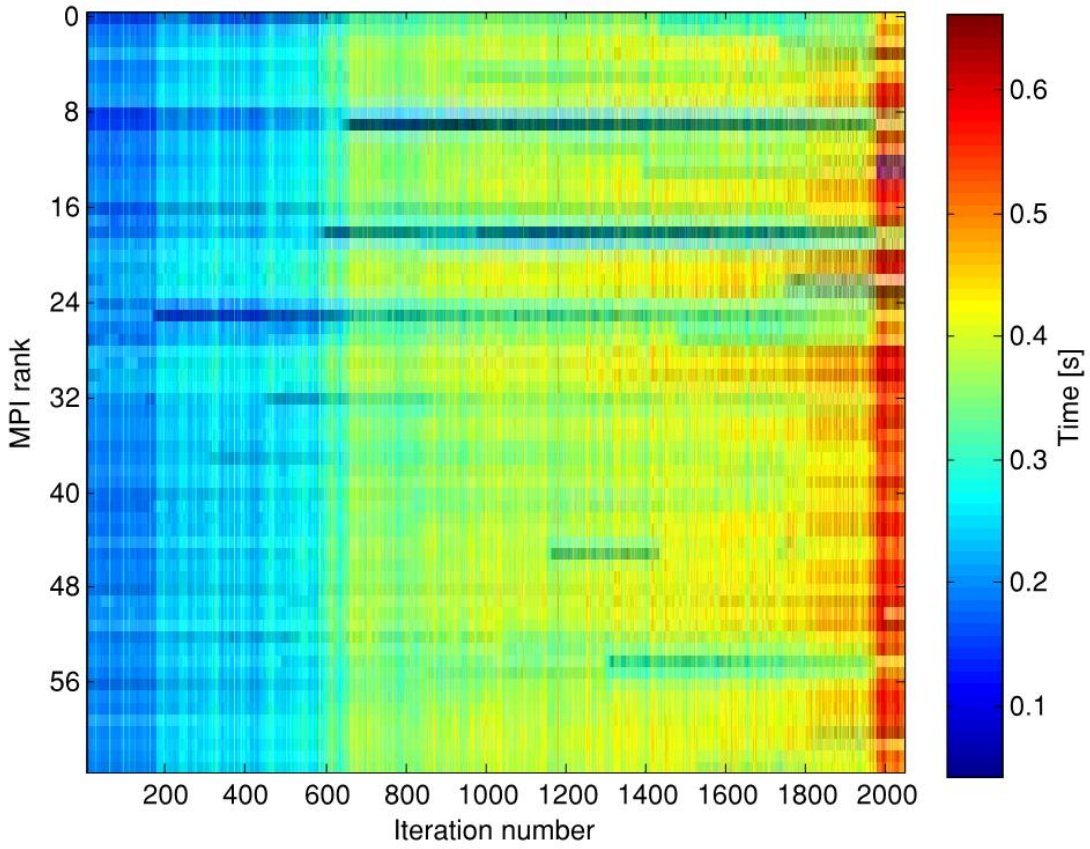
- Intra-phase
- Inter-phase

```
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;
}
```

One iteration of the progress loop

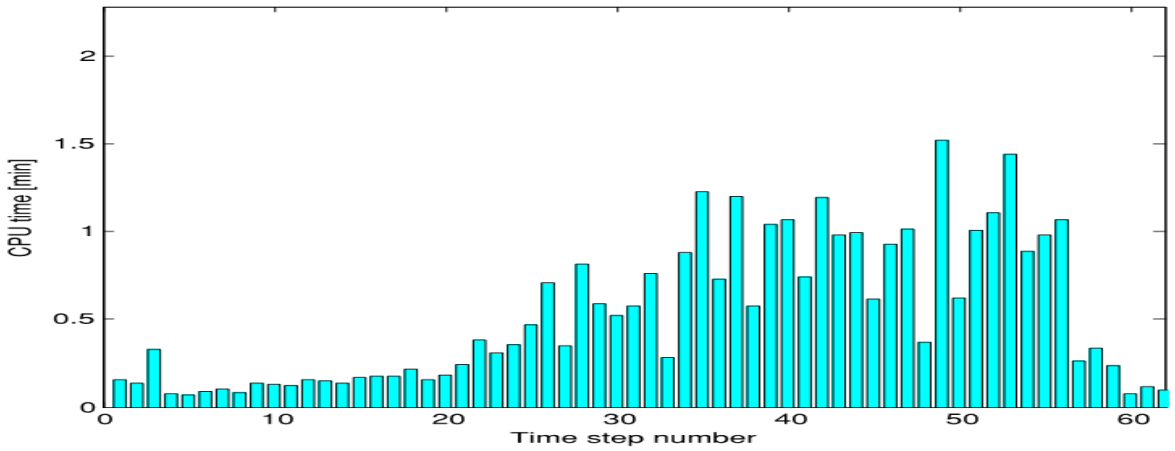
PEPC Benchmark of the DEISA Benchmark Suite

All-to-all
Performance
2048 phases

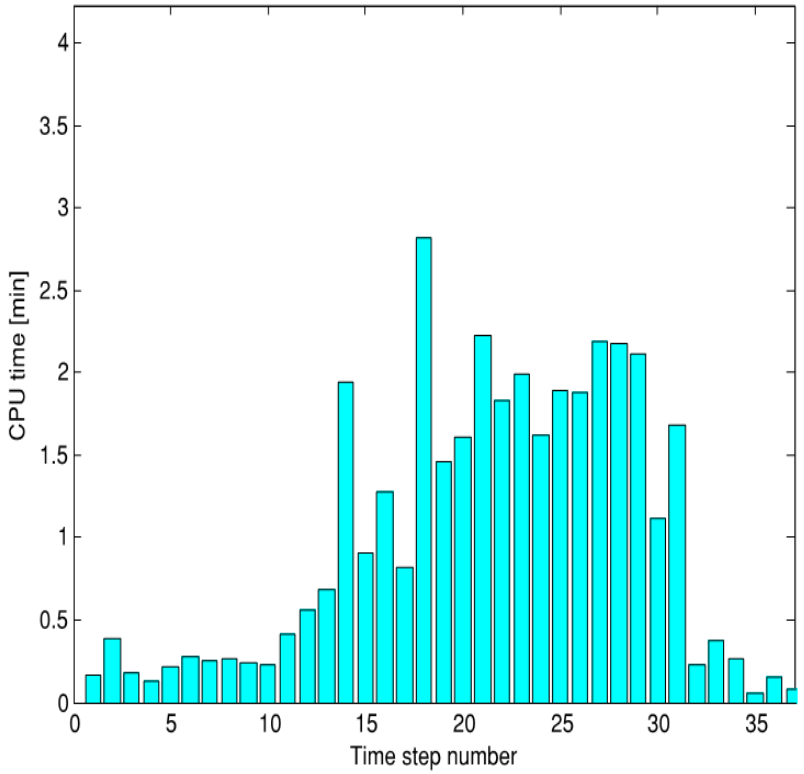


Inter-phase Dynamism

- Indeed application of GNS
- Identifiers for
  - adaptation strategy
  - Valleys vs hills

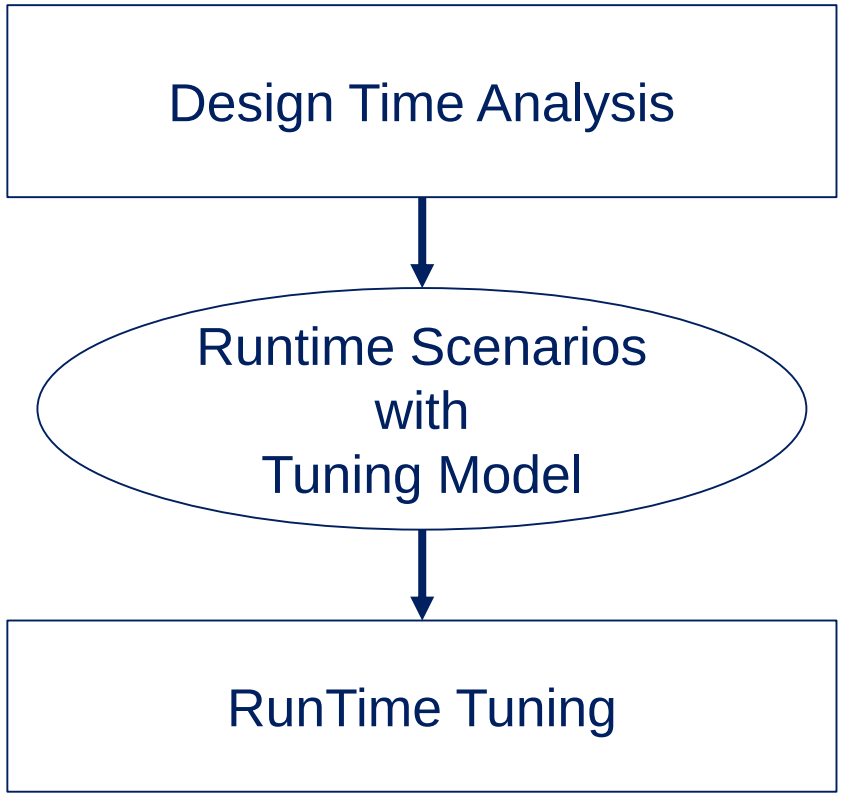


(a) Adaptation strategy 1



(b) Adaptation strategy 2

Scenario-Based Tuning



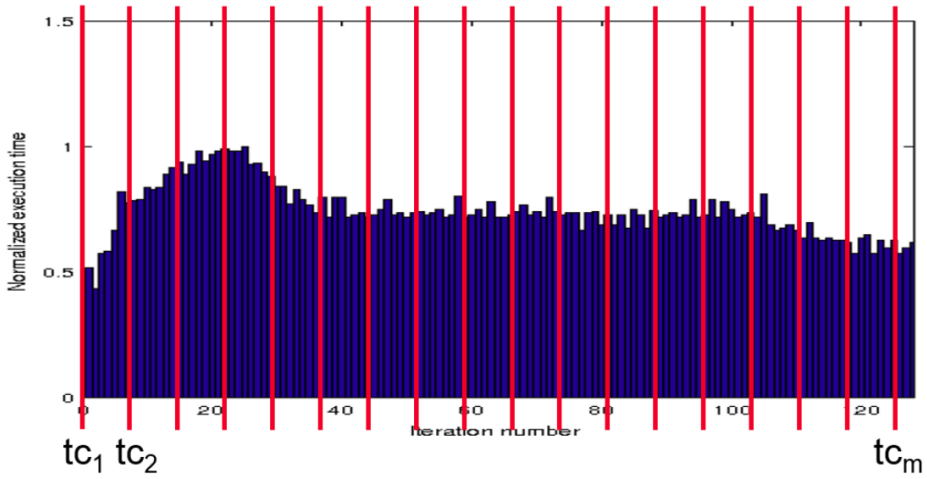
Periscope Tuning Framework (PTF)

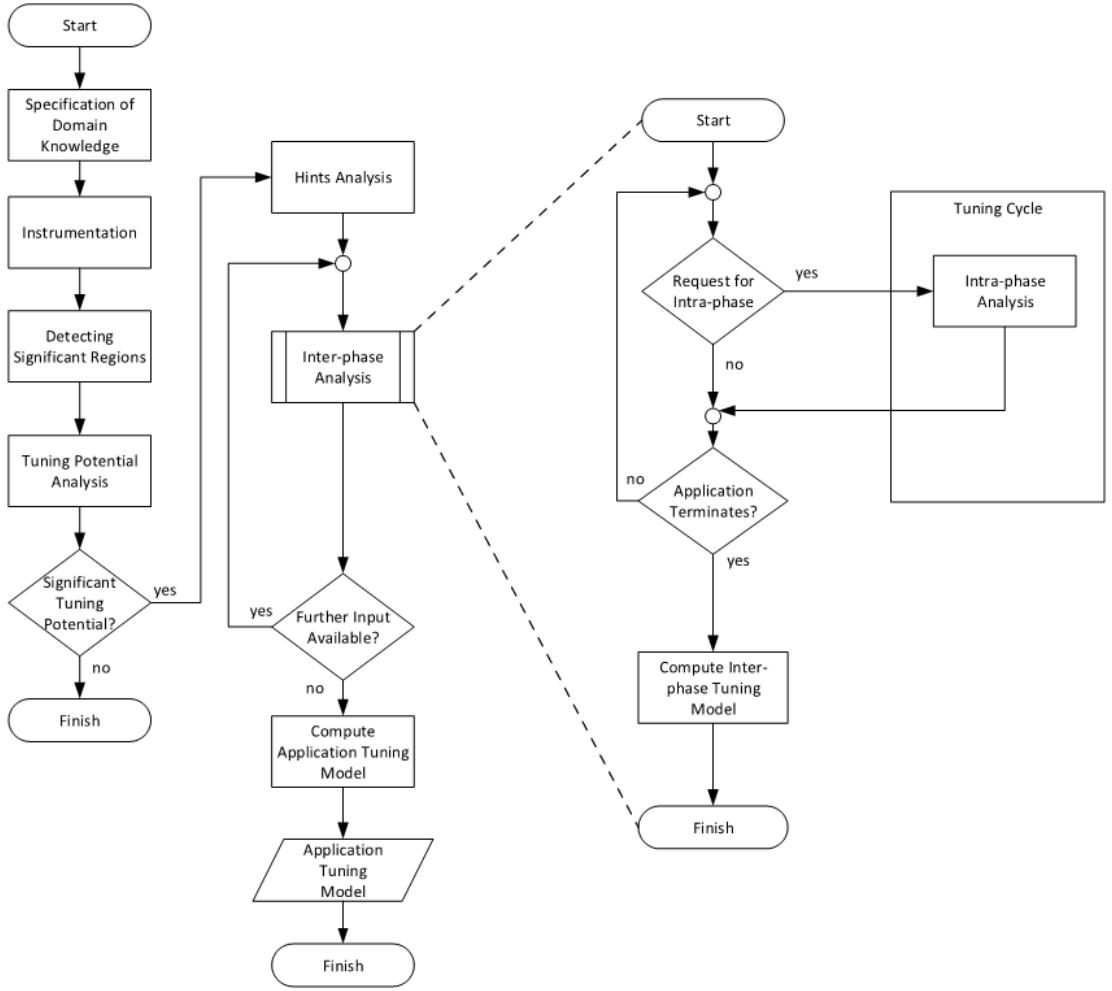
READEX Runtime Library (RRL)

Terminology

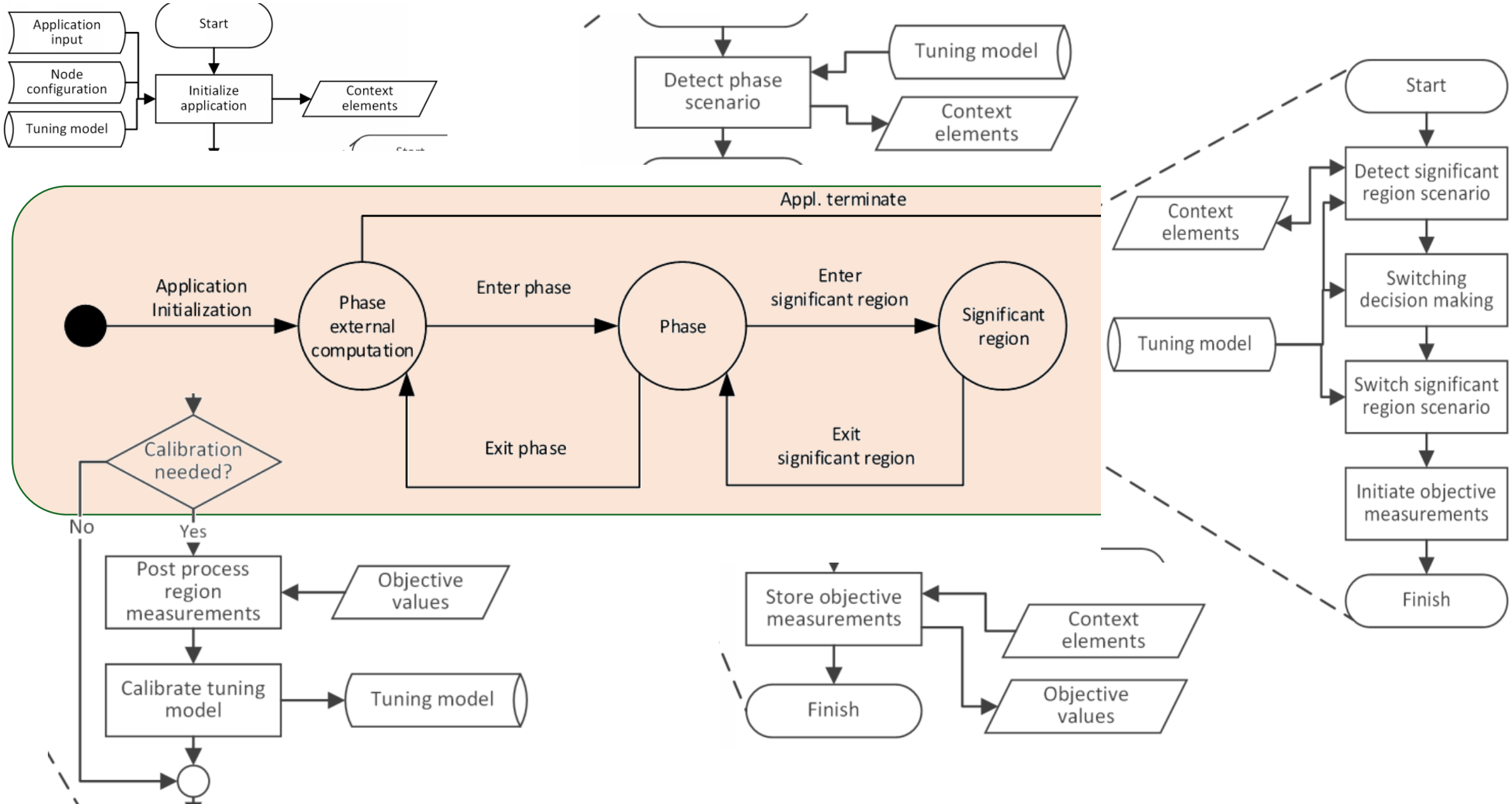
- **Significant Regions**: Coarse-granular code regions
- **Runtime Situations**: Instances of significant regions
- **Identifiers**: Distinguish rts's with different characteristics
  - Region ID, Call path, region parameters, phase identifiers, input identifiers
- **Scenarios**: rts's with same characteristics
- **Tuning Model**:
  - Set of scenarios
  - Classifier based on the identifiers
  - Selector for each scenario

Design Time Analysis with Periscope



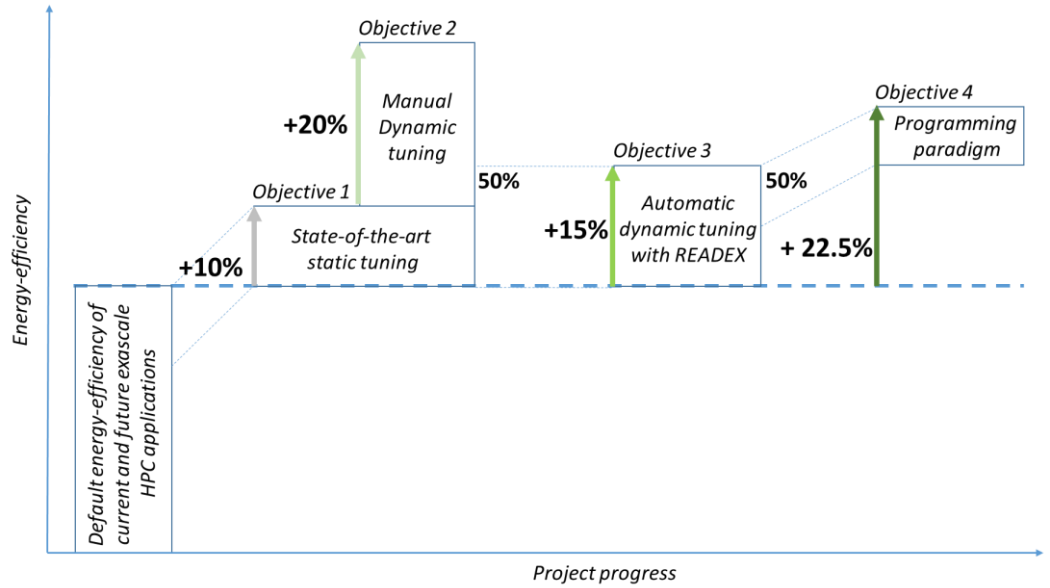


Runtime Tuning with the READEX Runtime Library



Validation and project goals

- **Goal: Validate the effect of READEX using real-world applications**
  - Co-design process:
    - Hand-tune selected applications
    - Compare results with automatic static and dynamic tuning
  - Energy measurements using HDEEM infrastructure



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

European Commission
Horizon 2020
European Union funding for Research & Innovation

Conclusion

- Energy-efficiency at exascale
  - Application developers and users will have to care
- Lack of capabilities
  - Awareness
  - Expertise
  - Resources
- Proposed solution – READEX:
  - Exploit dynamism
  - Detect at design-, exploit at run-time
  - Tools-aided auto-tuning methodology

Thank you! Questions?