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

EE HPC WG @ SC'17

Andreas Gocht – Technische Universität Dresden

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

European Commission
Horizon 2020
European Union funding for Research & Innovation

Overview: Project Partners

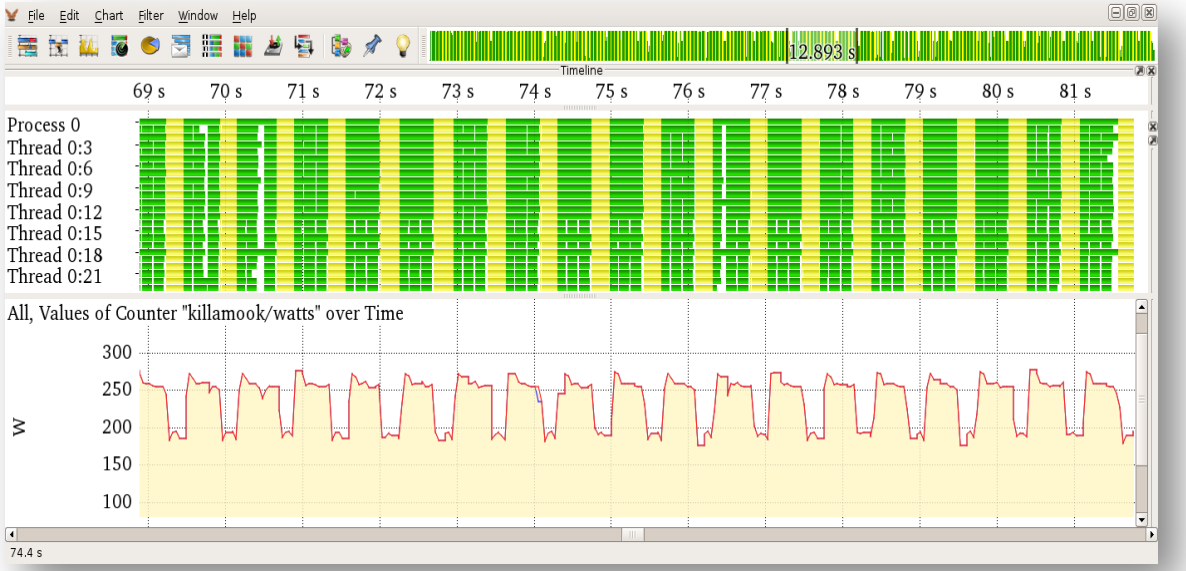
- Grant agreement No 671657
- Officially started September 1st, 2015

- 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

Project Motivation

Applications exhibit dynamic behaviour

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



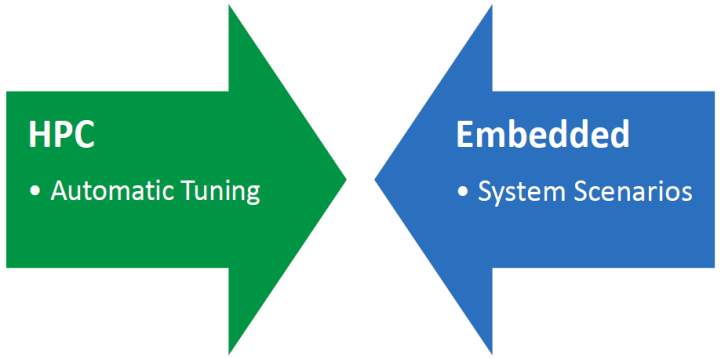
Overview

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
- Embedded Systems: System scenario methodology

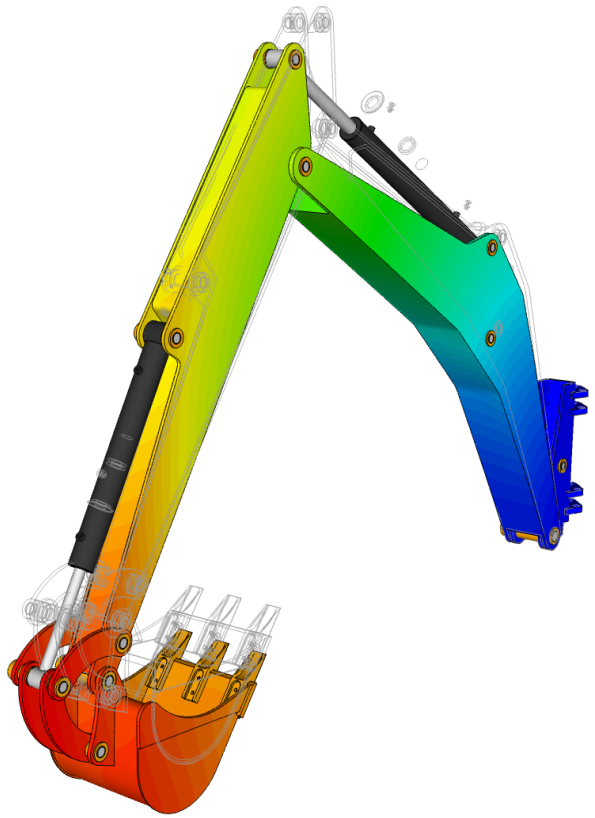


Overview

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



Terminology: Region and Region Instance

```
int main(void)
{
        // Initialize application
        int num_iterationr = 2;
        for (int iter = 1; iter <= num_iterations; iter++)
        {
                laplace3D();
                reduction();
                fftw_execute();
        }
        // Post processing and finalization
        return 0;
}
```

Phase region

Phase

Significant region

Runtime situation

READEX
Runtime Exploitation of Application Dynamism for Energy-efficient eXascale computing
European Commission
Horizon 2020
European Union funding for Research & Innovation

Terminology: Tuning Parameter

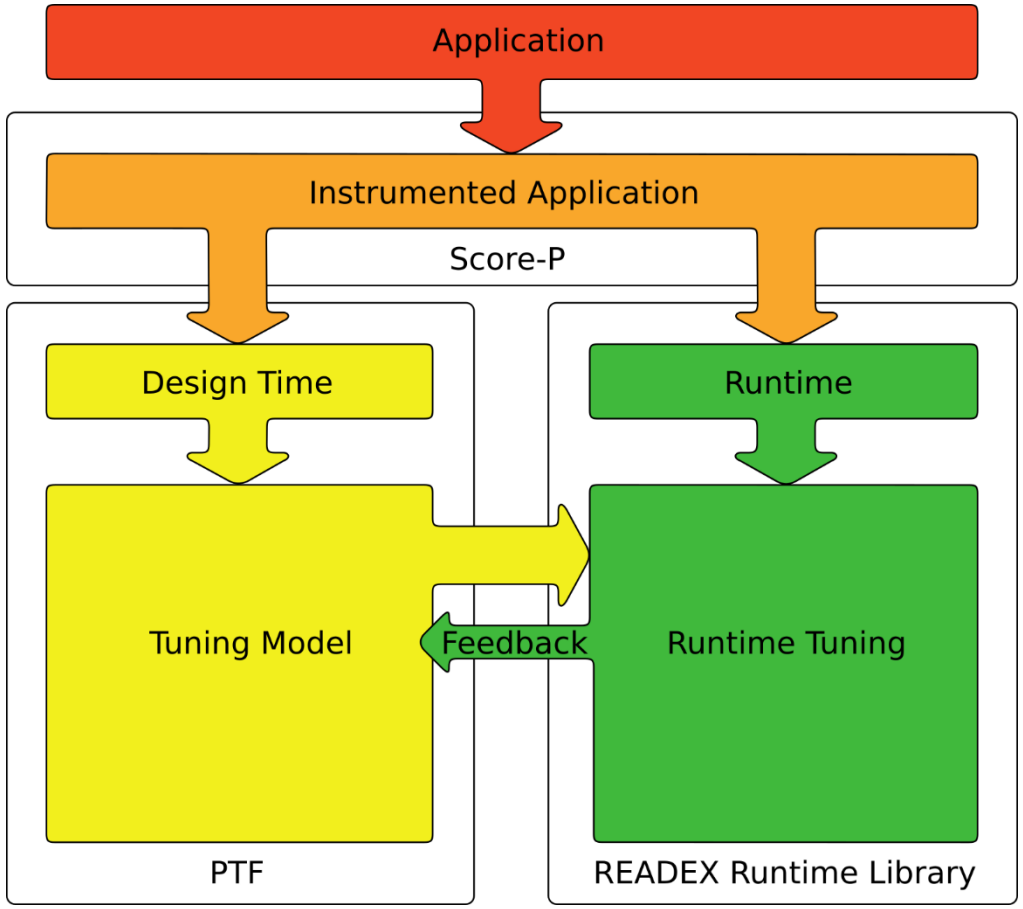
```
int main(void)
{
        // Initialize application
        int num_iterationr = 2;
        for (int iter = 1; iter <= num_iterati
        {
                laplace3D();
                reduction();
                fftw_execute();
        }
        // Post processing and finalization
        return 0;
}
```

Tuning Parameter FREQ=2 GHz

Tuning Parameter FREQ=1.5 GHz

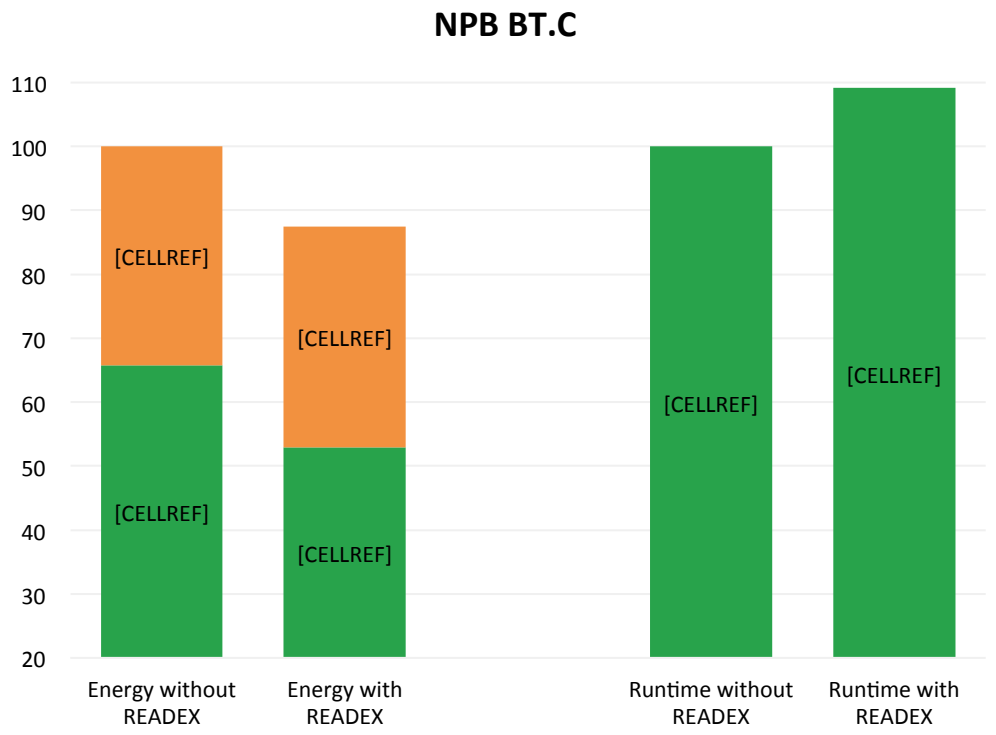
Overview: Workflow

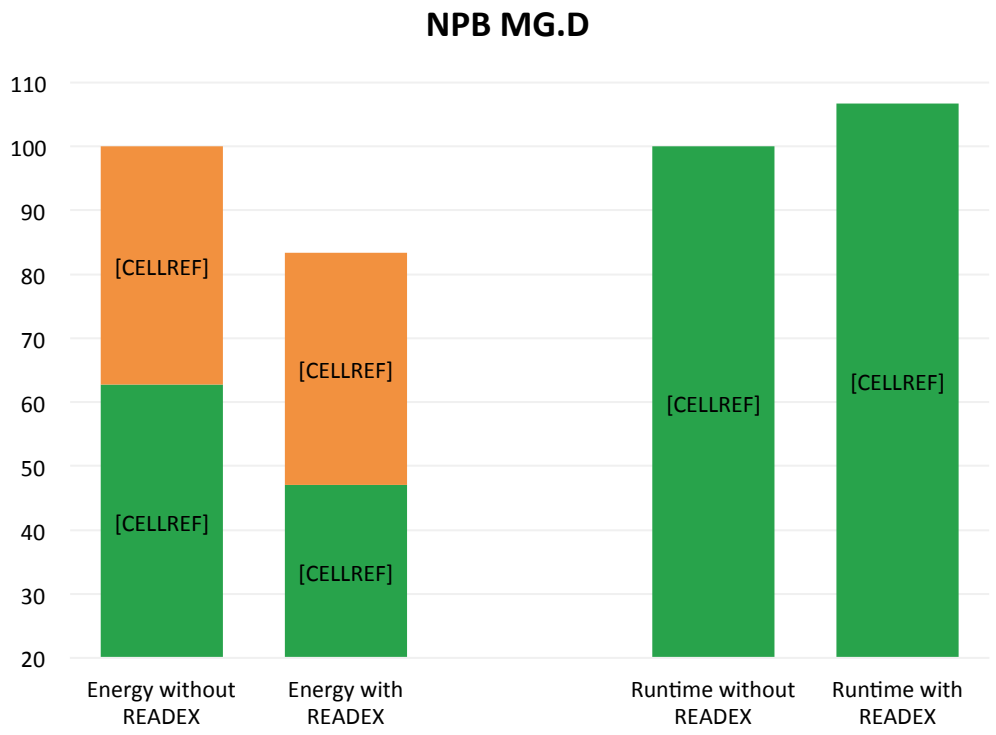
1. Instrument application
   Score-P provides different kinds of instrumentation
2. Detect dynamism
   Check whether runtime situations could benefit from tuning
3. Detect energy saving potential and configurations (DTA)
   Use tuning plugin and power measurement infrastructure to search for optimal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



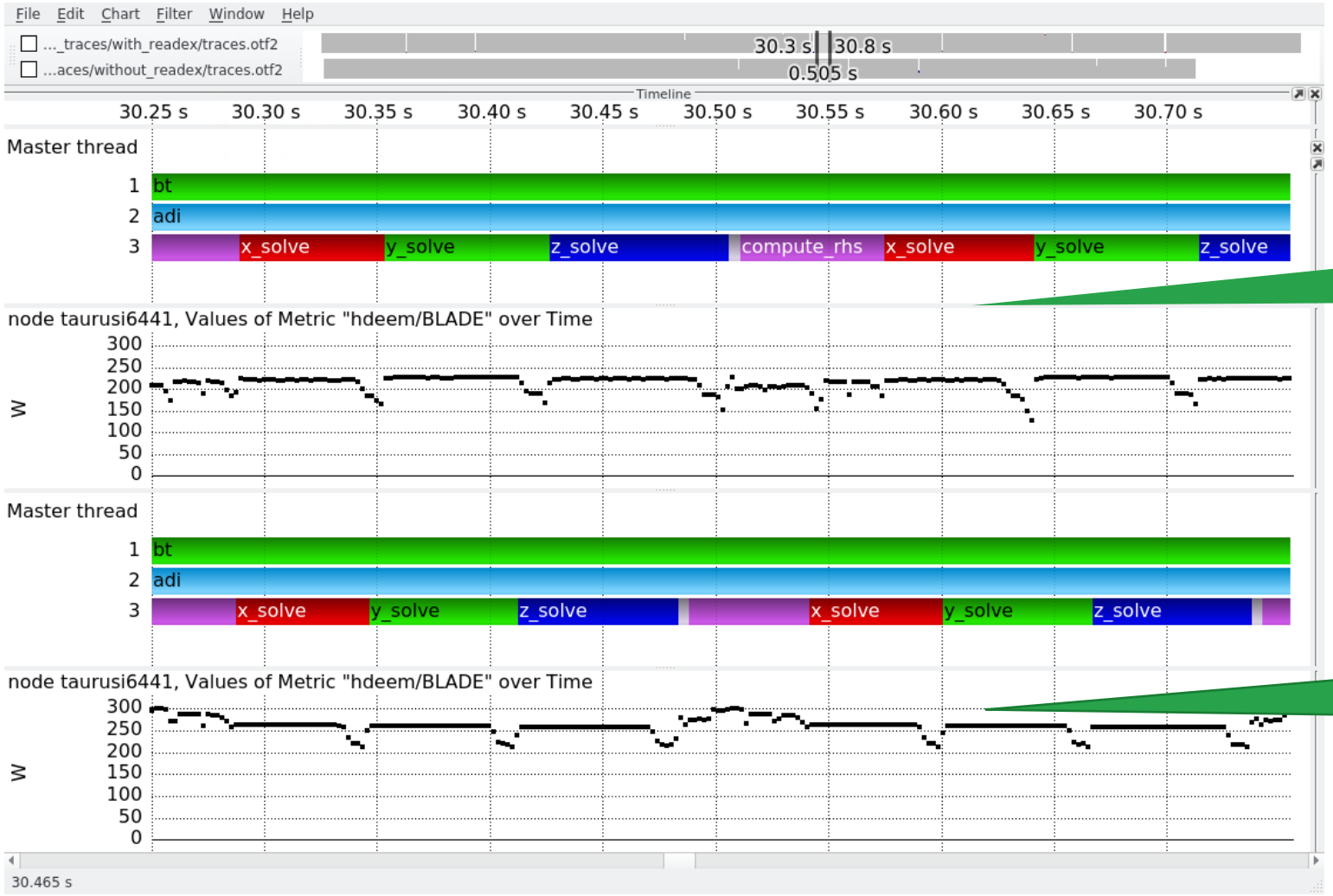
Current Status

- Application instrumentation, dynamism detection, DTA, and RAT are implemented
- Promising results:





Example: NAS-OMP BT.C Benchmark

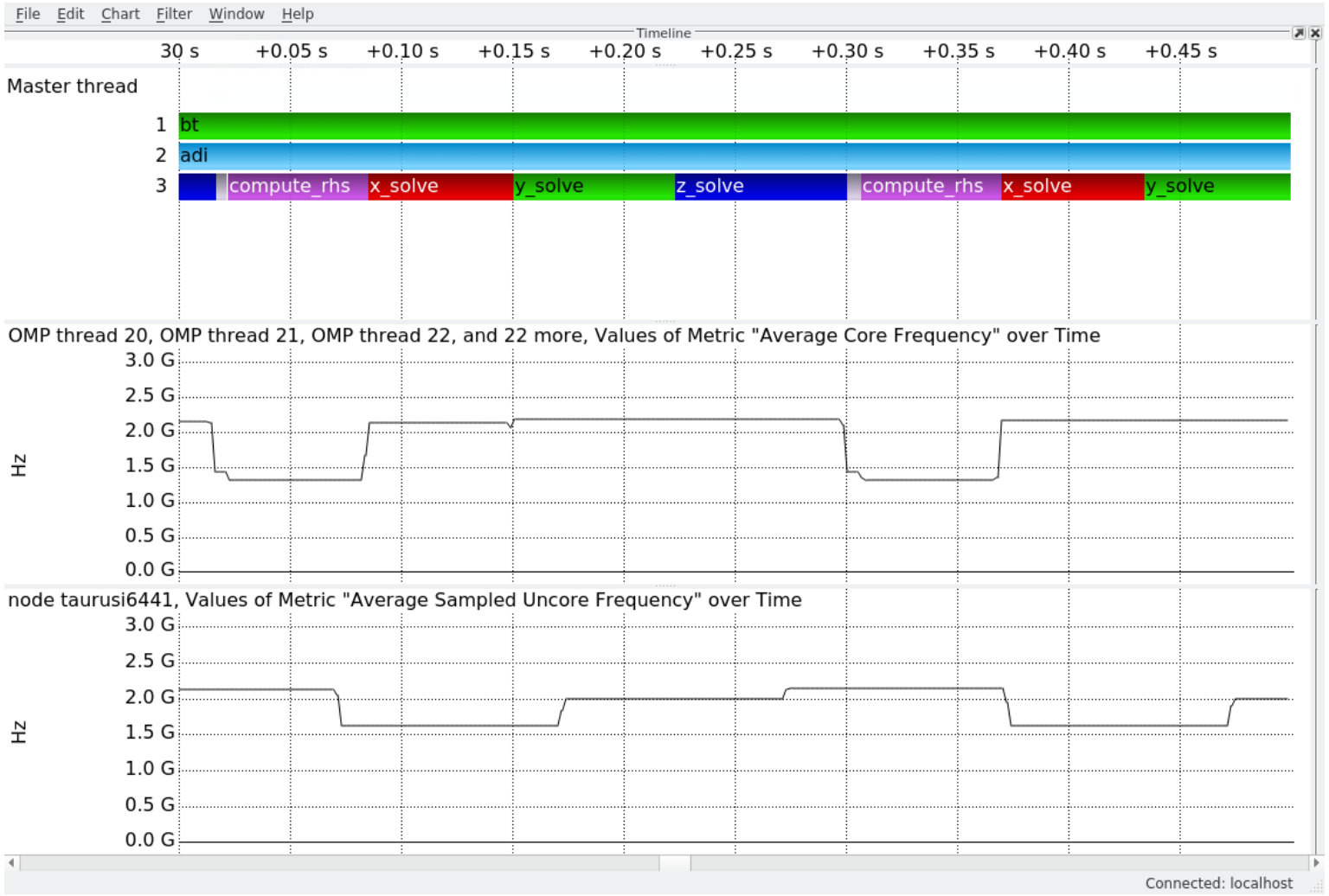


With READEX

Without READEX

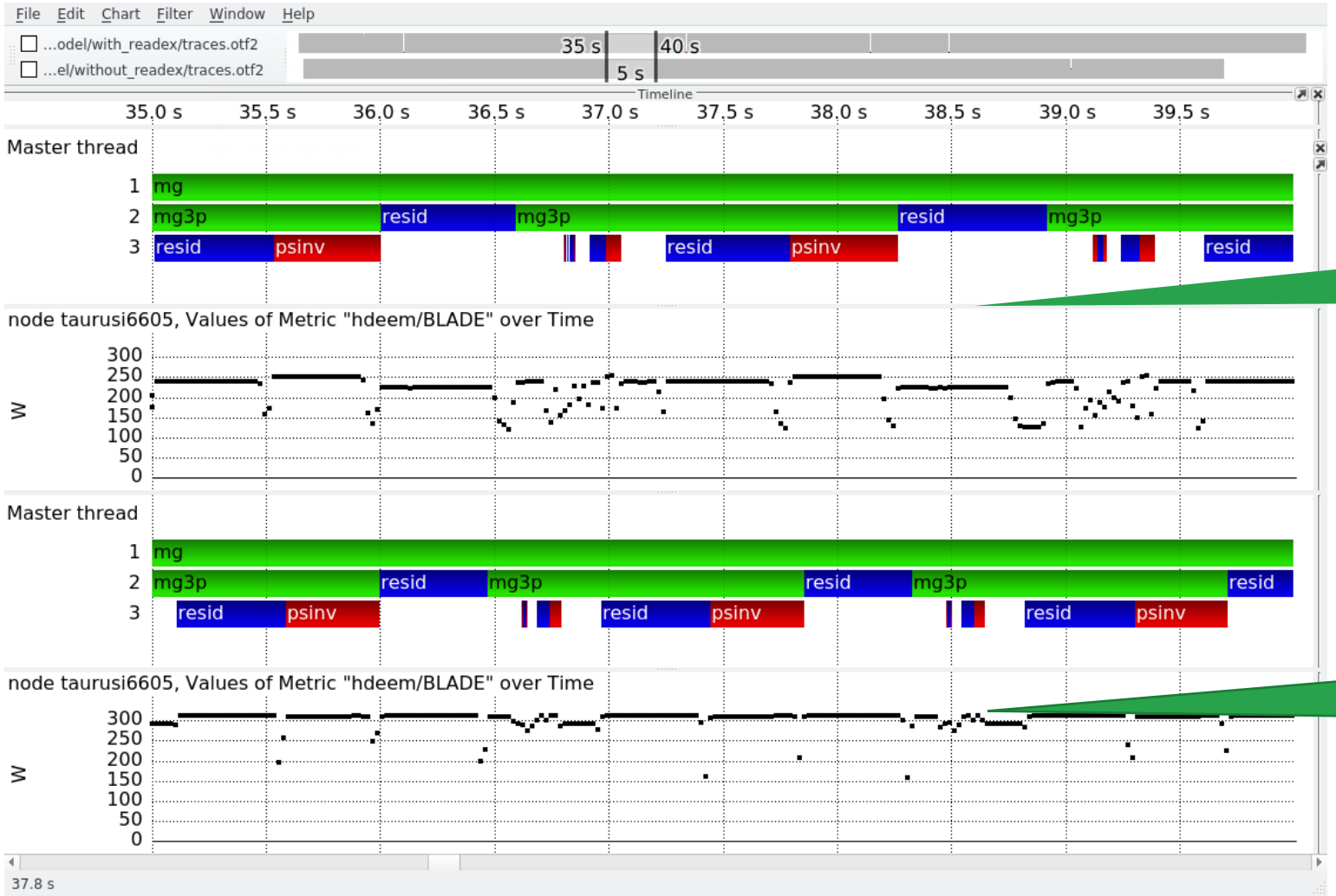
Comparison of power consumption and runtime of the BT benchmark

Example: NAS-OMP BT.C Benchmark



Selected tuning parameter: core and uncore frequency of the BT benchmark

Example: NAS-OMP MG.D Benchmark

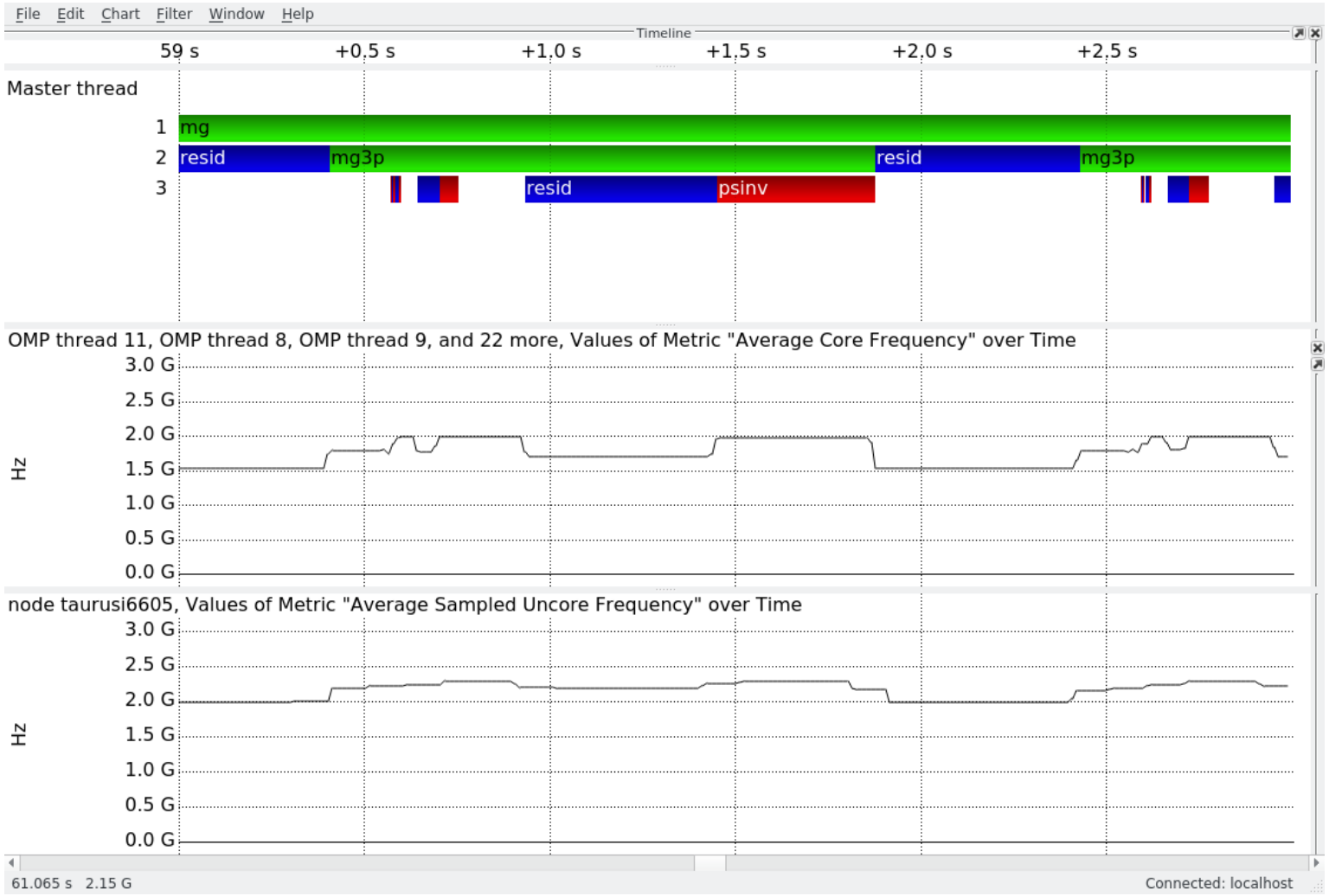


With READEX

Without READEX

Comparison of power consumption and runtime of the MG benchmark

Example: NAS-OMP MG.D Benchmark



Selected tuning parameter: core and uncore frequency of the MG benchmark

Work in Progress

- Evaluating Application Tuning Parameters (ATPs)
  - Allow the tuning of program internal “decisions”
  - Example preconditioner in ESPRESO a solver for FETI problems:

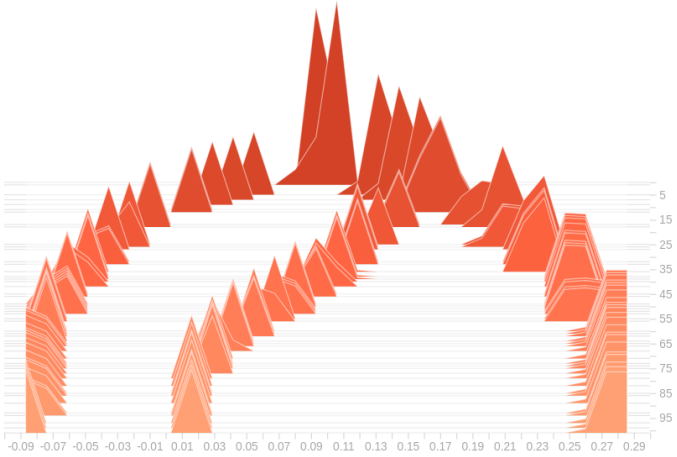
Preconditioner type	Number of iterations	Single iteration cost Time and energy		Total solution cost Time and energy	
No preconditioner	172	130 + 0 ms	32.3 + 0.00 J	21.4 s	5.50 kJ
Weight function	100	130 + 2 ms	32.3 + 0.53 J	12.9 s	3.28 kJ
Lumped	45	130 + 10 ms	32.3 + 3.86 J	6.3 s	1.64 kJ
Light Dirichlet	**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**

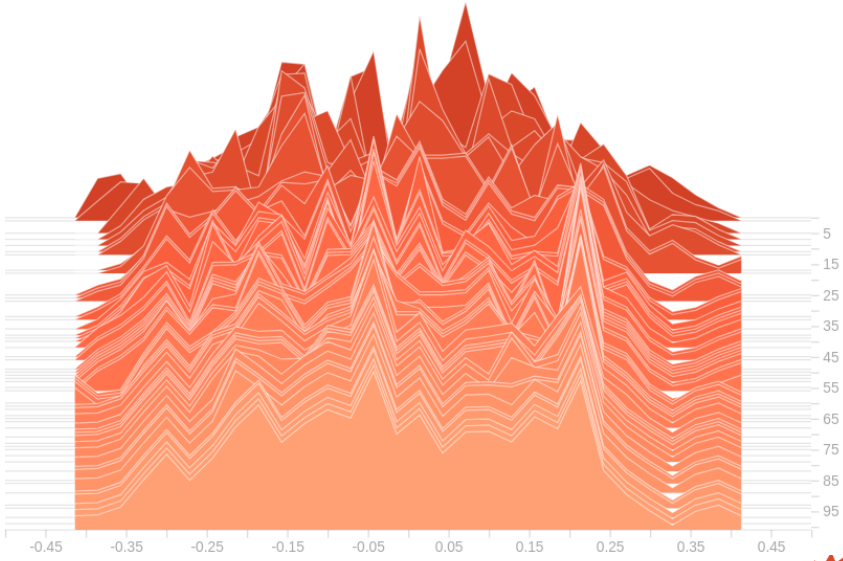
11.3% energy savings against the default full Dirichlet preconditioners

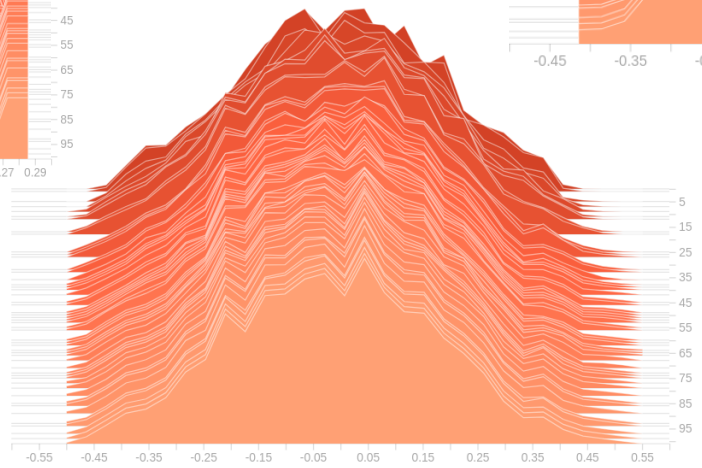
Note: 130 ms and 32.3 J – is a baseline for single iteration cost without preconditioner

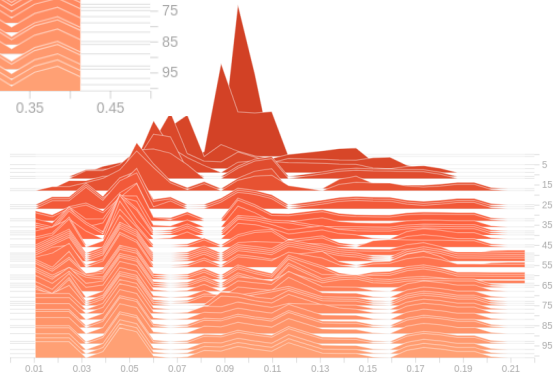
Work in Progress

- Runtime Calibration
  - Automatically finds a good configuration for unseen scenarios
  - Applied at runtime
  - Machine Learning techniques









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

European Commission
Horizon 2020
European Union funding for Research & Innovation

Thank you for your attention

Questions?