Application-level Energy Awareness for OpenMP

Ferdinando Alessi[1], Peter Thoman[1], Giorgis Georgakoudis[2]
Thomas Fahringer[1], and Dimitrios S. Nikolopoulos[2]

[1]University of Innsbruck, [2]Queen's University of Belfast

Agenda

- Introduction
- Language Extensions
  - Objective Clause
  - User-defined Tunable Parameters
- Prototype Implementation
  - Compilation
  - Runtime Optimization
- Evaluation
- Discussion

Introduction

Background

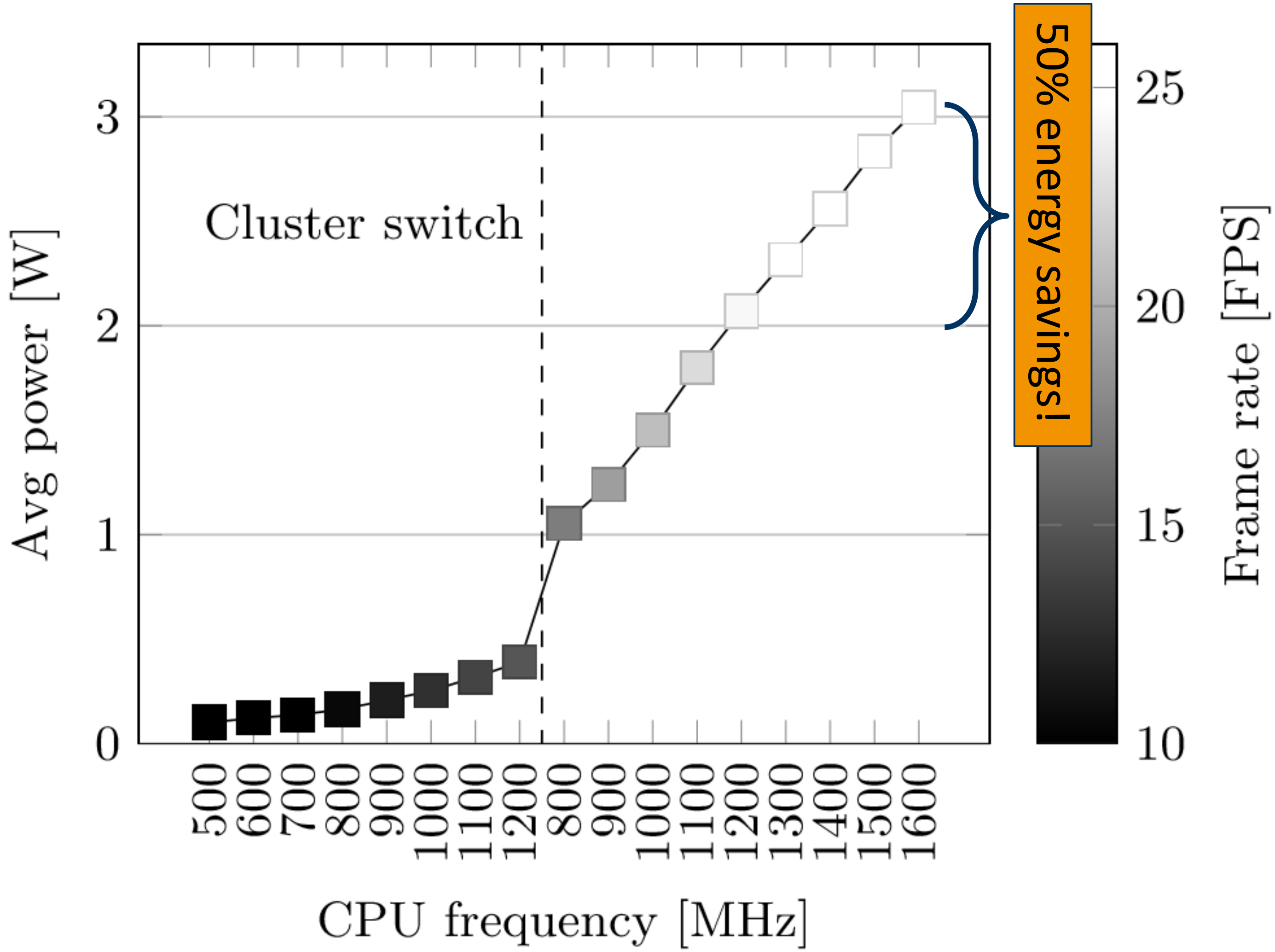
- Modern use cases often feature **multiple optimization goals**
  - Not just execution time, also e.g. energy or power consumption
  - Relevant in embedded and mobile systems, but also HPC!

- Control of **parallelism** is a major tradeoff factor
  - Might differ according to program region
  - Needs application-level knowledge

→ Idea:

Extend OpenMP to allow programmers to steer multi-objective optimization and enable application-level tuning

Motivation
- Video decoder application



Language Extensions

Multi-objective Optimization Clause

- Define a new clause which allows the programmer to specify an optimization *goal function* and *constraints*

- **`objective`**(*weights* [: *constraints*])
  - *weights* = $f_1 * P_1 + f_2 * P_2 + \cdots + f_N * P_N$
  - *constraints* = $\{P_i < c_i;\ constraints\} \mid \emptyset$

Base language expressions of a numeric type

A non-functional parameter, currently:
T (execution time), **P** (power),
E (energy), **Q** (quality of service)

Multi-objective Optimization Clause

- Define a new clause which allows the programmer to specify an optimization *goal function* and *constraints*

- **`objective`**(*weights* [: *constraints*])
  - *weights* $= f_1 * P_1 + f_2 * P_2 + \cdots + f_N * P_N$
  - *constraints* $= \{P_i < c_i;$ *constraints*$\} \mid \emptyset$

- Can be applied to `parallel`, `for` and `task` constructs

Objective Clause Examples

```
#pragma omp parallel objective(E)
```

- Minimize energy consumption in the binding parallel region

```
#pragma omp for objective(0.8*E+0.2*T)
```

- Weighted energy (0.8) & time (0.2) optimization in this for loop

```
double p;
…
#pragma omp task objective(T : P<p)
```

- Complete this task in minimum time while staying below the given power consumption p

Tunable Parameters

- In order to achieve optimization goals,
  the runtime system can adjust various parameters:
  - Degree of parallelism (DOP), dynamic voltage and frequency scaling (DVFS), …
- However, there are also application-specific parameters
  - Either non-functional, e.g. tiling sizes in a numeric algorithm
  - Or influencing the quality of service, e.g. image, audio or video quality in decoders

Tunable Parameter Clause

- A new clause which allows the programmer to expose application-level tunable parameters

- **`param`**(*var*, (range(*value-range* [: *q-range*])
  | enum(*values*, *size* [: *q-values*])))
  - *value-range* = *q-range* = start, end, step

Base language variable

Base language expressions of a numeric type

Base language array of same type as *var*

Tunable Parameter Clause

- A new clause which allows the programmer to expose application-level tunable parameters

- **`param`**(*var*, (`range`(*value-range* [: *q-range*])
  | `enum`(*values*, *size* [: *q-values*])))
  - *value-range* = *q-range* = start, end, step

- Can be applied to `parallel`, `for` and `task` constructs
  - If used without `objective`, assume `objective(T)`

Parameter Clause Examples

```
#pragma … param(rate, range(24, 74, 10))
```

- `rate` can be freely set to 23, 34, …, 74 in the binding region

```
#pragma … param(rate, range(24,74,10 : 5,0,-1))
```

- Associates a quality of service with each setting
  - E.g. `rate=34` ➔ `Q=4`

```
#pragma … param(method, enum(methods, N))
```

- `method` can be set to any entry in the `methods` array
  - E.g. Function pointers implementing different solvers

Region Construct

- For completeness, allow applying different optimization / parameters independent of parallelization

- `#pragma omp **region** [objective(…)] [param(…)]`
  `structured-block`

- Applies the given objective and parameters for the binding structured block

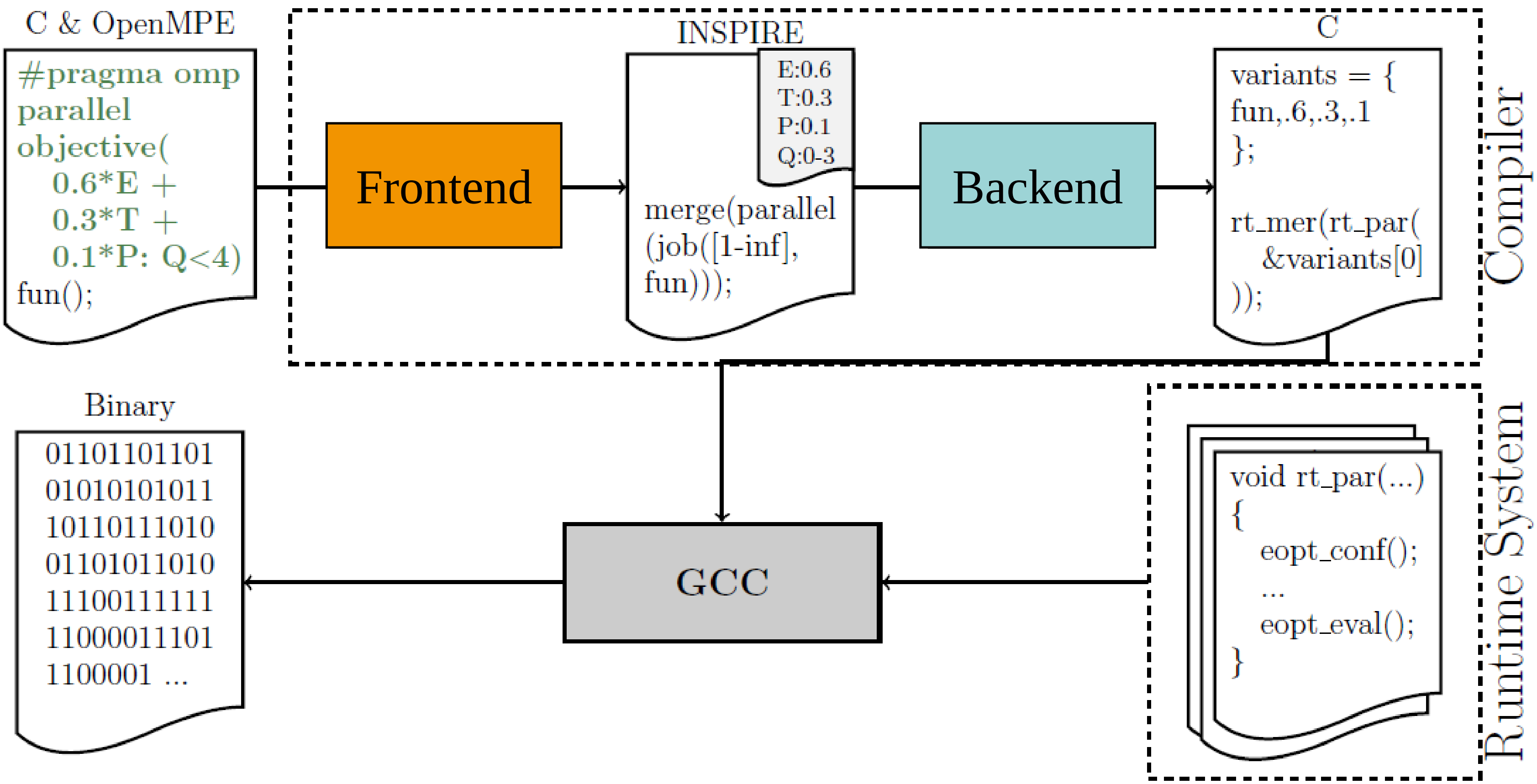
Prototype Implementation

Implementation Background

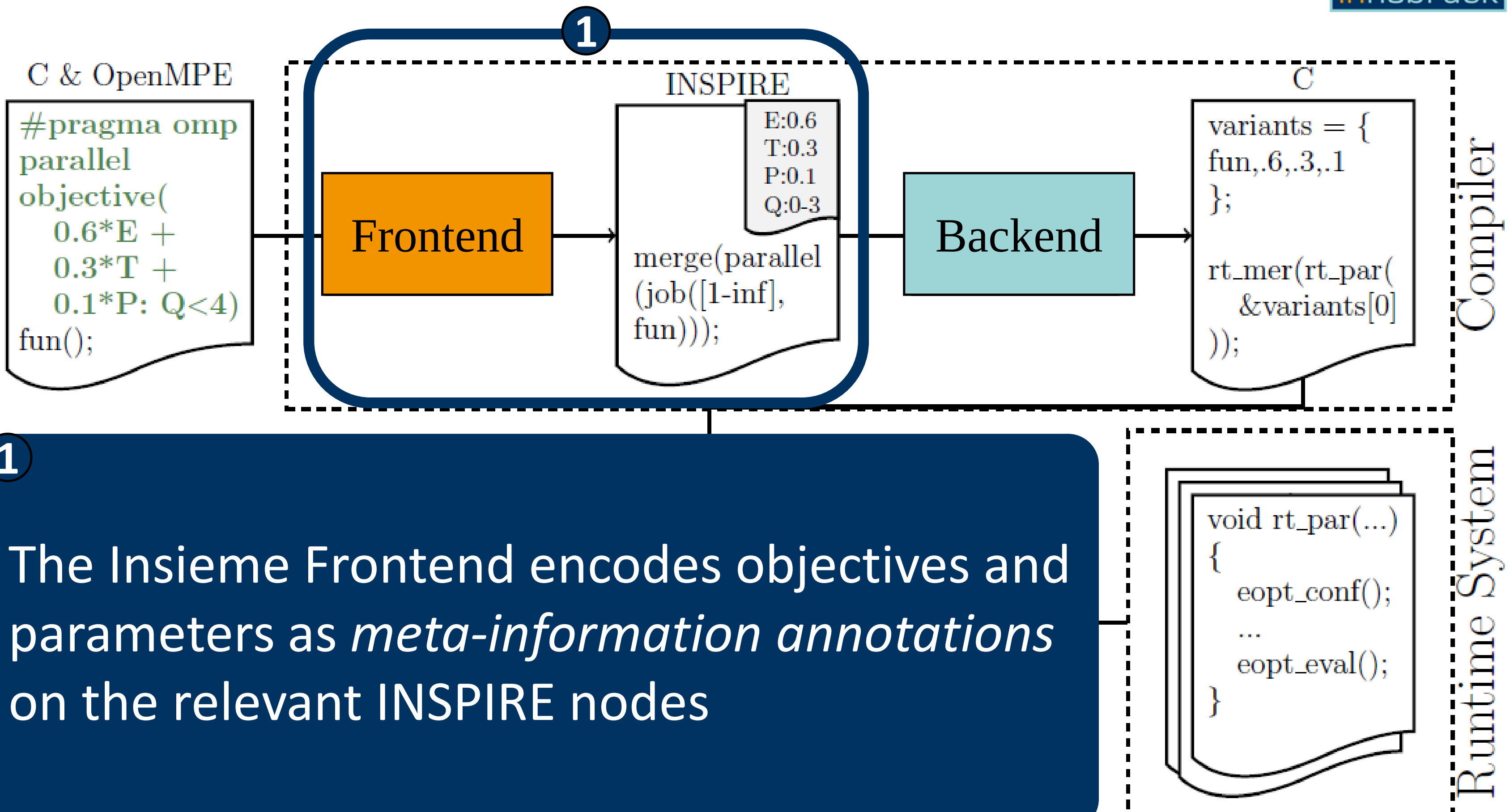
- Implemented in the Insieme Compiler
- A source-to-source C/C++ compiler framework
  - *Frontend* based on Clang
  - *Backend* targets GCC/ICC/llvm + the Insieme runtime system
  - *Core*: Analysis and transformation based on INSPIRE
- Insieme Runtime System
  - Based on user-space task scheduling with work stealing
- Previously used in OpenMP loop and task optimization

Automatic OpenMP Loop Scheduling: A Combined Compiler and Runtime Approach (IWOMP2012)
INSPIRE: The Insieme Parallel Intermediate Representation (PACT2013)
Compiler Multiversioning for Automatic Task Granularity Control (CCPE2014)

Implementation Overview



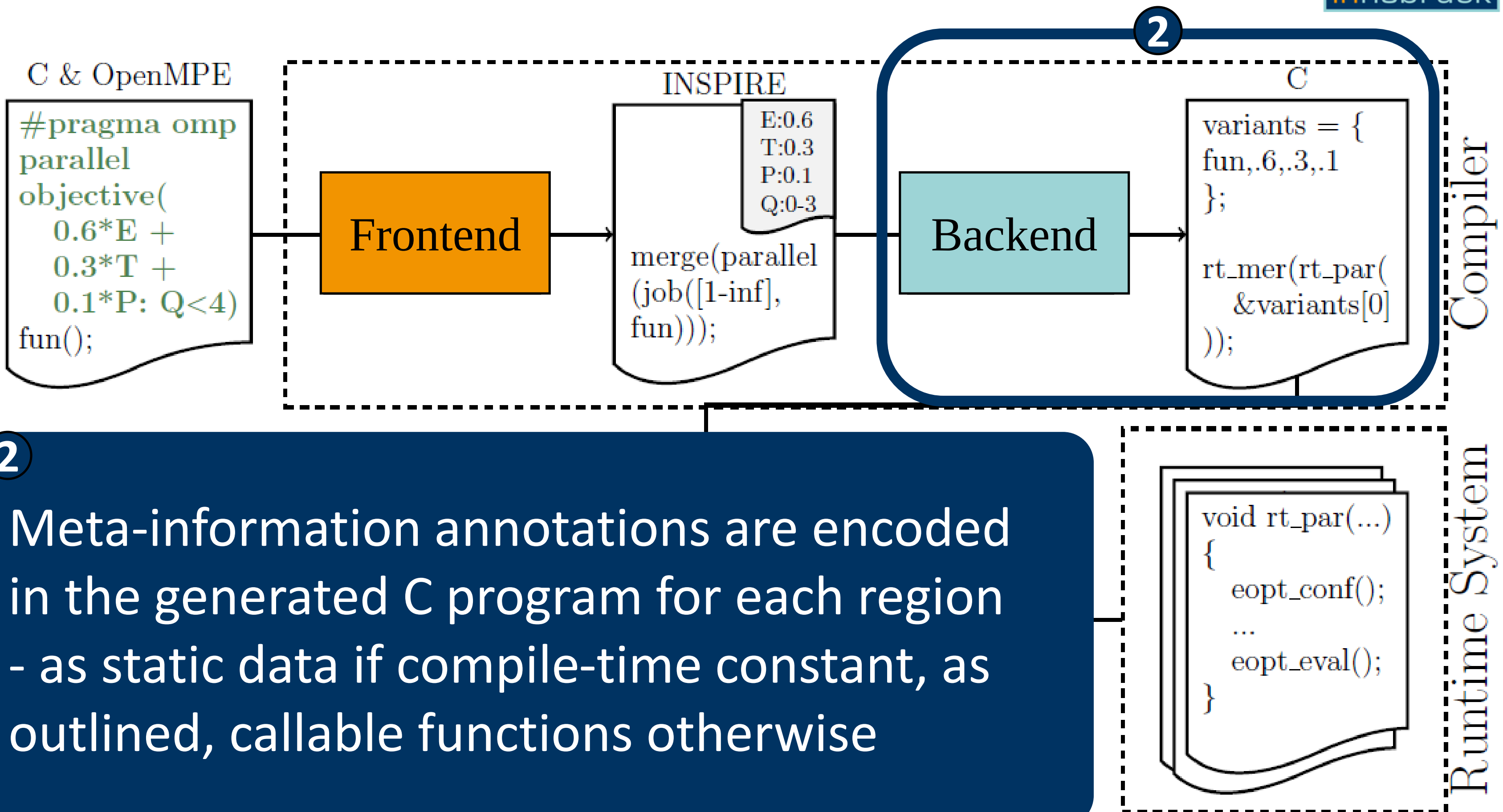
Implementation Overview



1

The Insieme Frontend encodes objectives and parameters as *meta-information annotations* on the relevant INSPIRE nodes

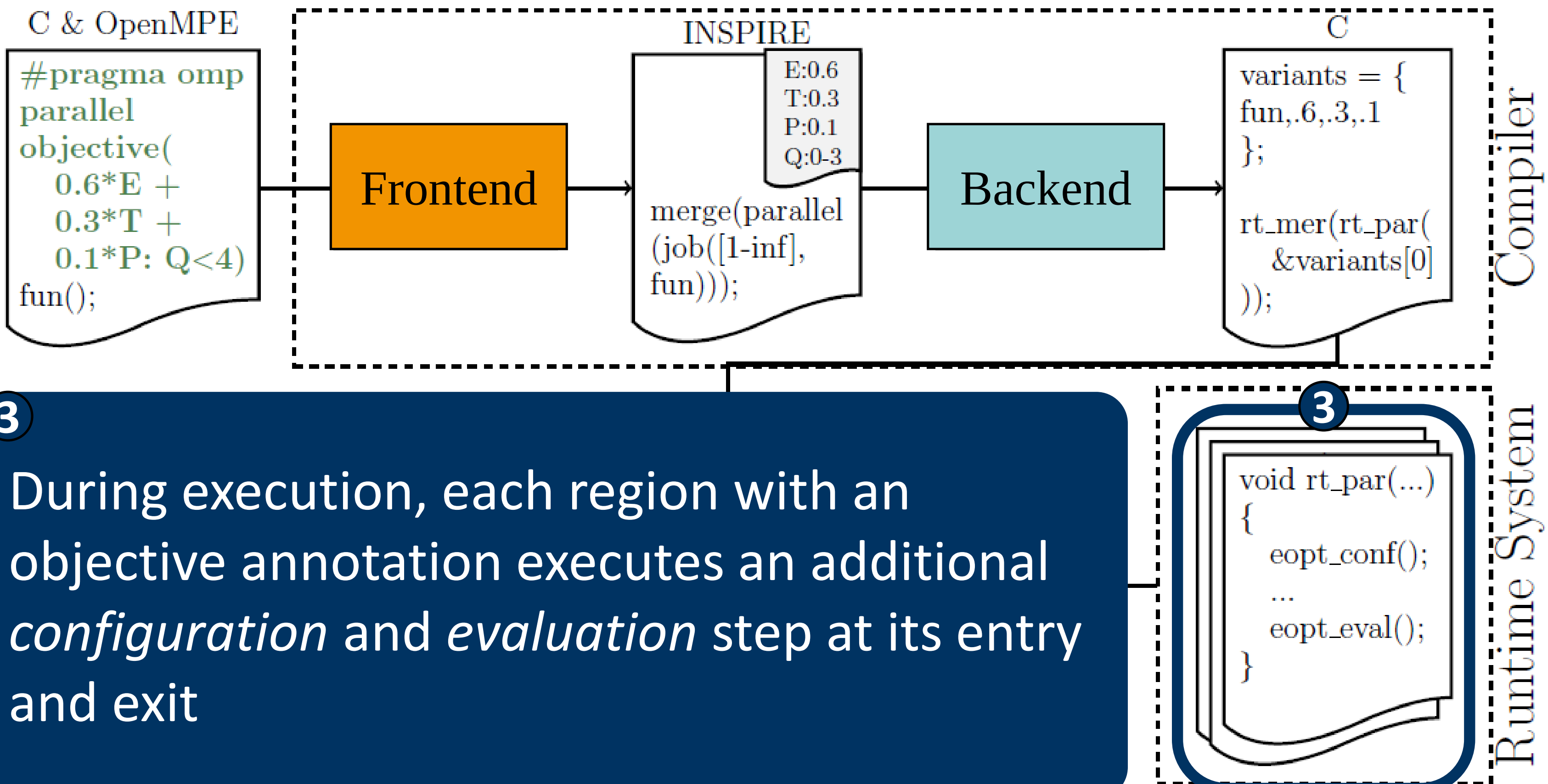
Implementation Overview



2

Meta-information annotations are encoded in the generated C program for each region - as static data if compile-time constant, as outlined, callable functions otherwise

Implementation Overview

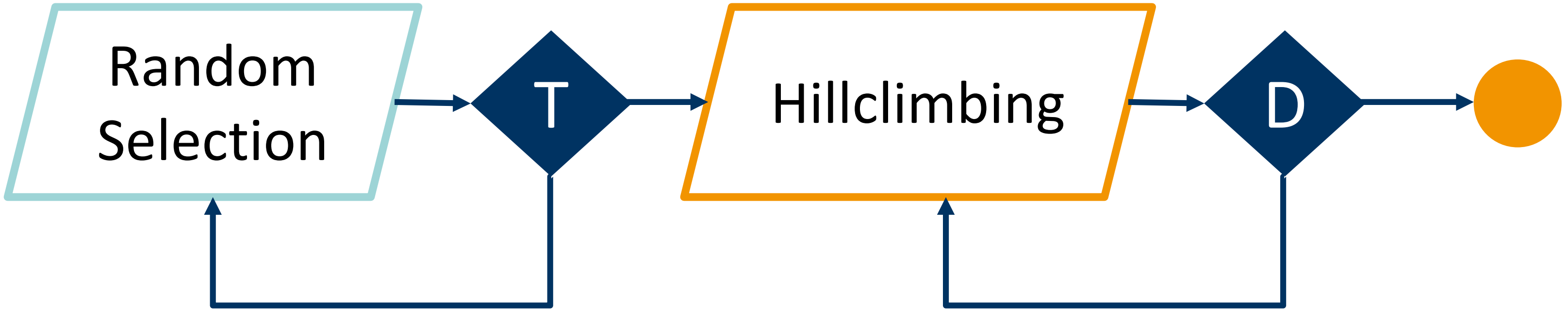


3

During execution, each region with an objective annotation executes an additional *configuration* and *evaluation* step at its entry and exit

Runtime Optimization

- Muti-stage (per-region) search approach
  1. Random selection until a threshold number of times T
  2. Hill climbing until no improvement in any dimension



- Assumptions:
  - Regions to optimize are executed multiple times
  - Region behaviour doesn‘t change significantly

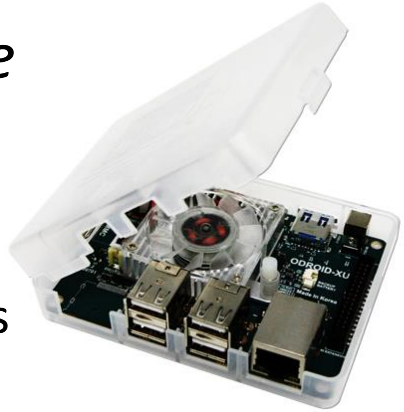
Runtime Optimization

- In evaluation steps:
  - Need to compute goal function and check constraints
  - Obtain values for paramters:
    - **Q**: directly from setting and user-defined mapping
    - **T**: fine-grained per-region measurements in Insieme Runtime System
    - **P** and **E** require hardware-specific support
- Energy/power: we use RAPL on Intel and a custom library (directly accessing HW sensors) on the XU+E ARM board
  - Hardware readout frequency limits minimum time granularity possible during optimization

Evaluation

Experiment Setup

- Hardware: 2 device classes: *desktop* and *mobile*
  - *Desktp:* Intel i7-3770k Ivy Bridge quad core
  - *Mobile:* ODROID XU+E development board
    - Exynos 5 Octa with 2 core clusters: 4 A15 and 4 A7 cores



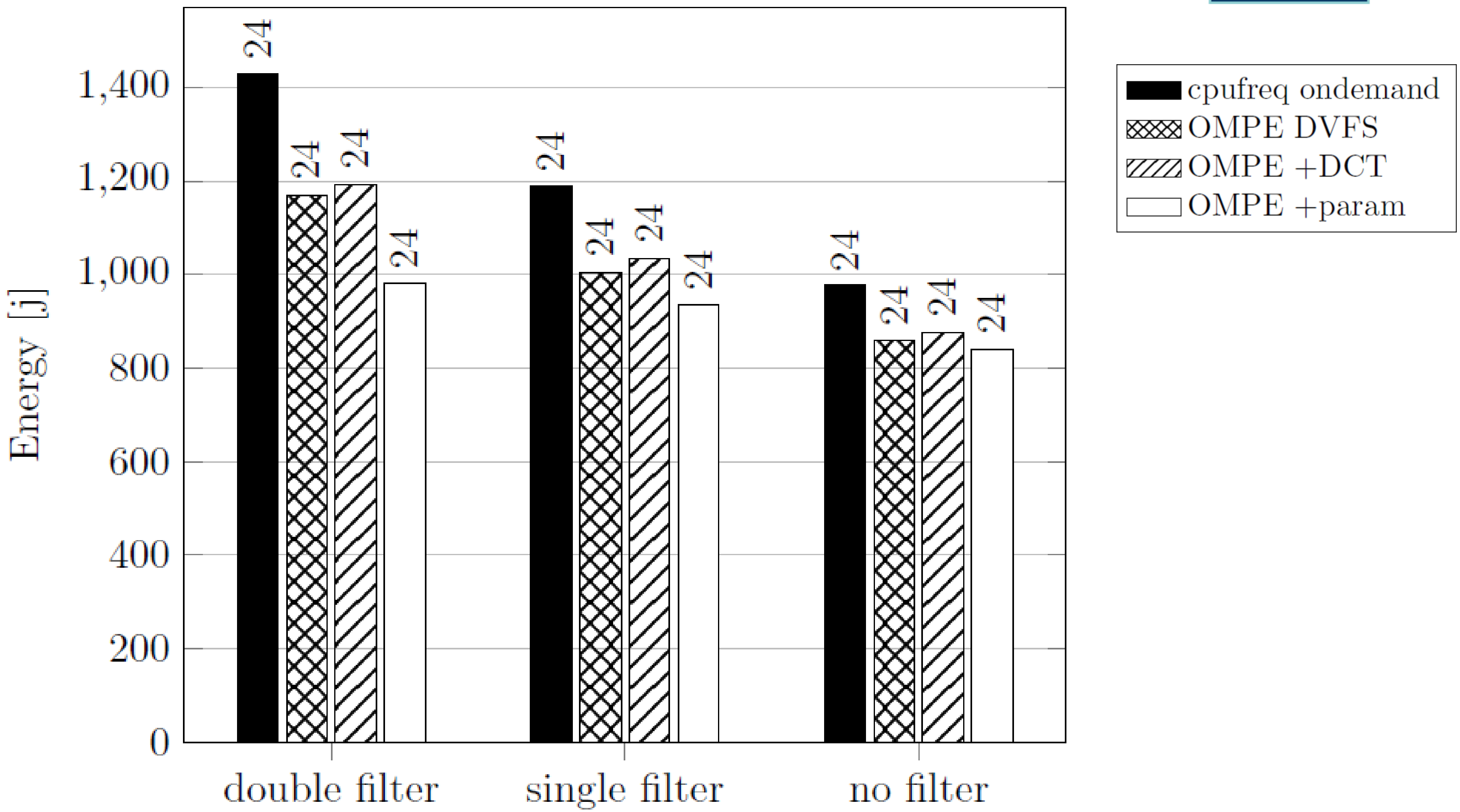
- Application: video decoder from MediaBench
  - Used *tmndec* for simplicity of OpenMP parallelization
  - Added optional horizontal and vertical deposterization filters to test multiple load scenarios
  - Reference run: ondemand CPU frequency governor

Tmndec Main Loop Pseudocode

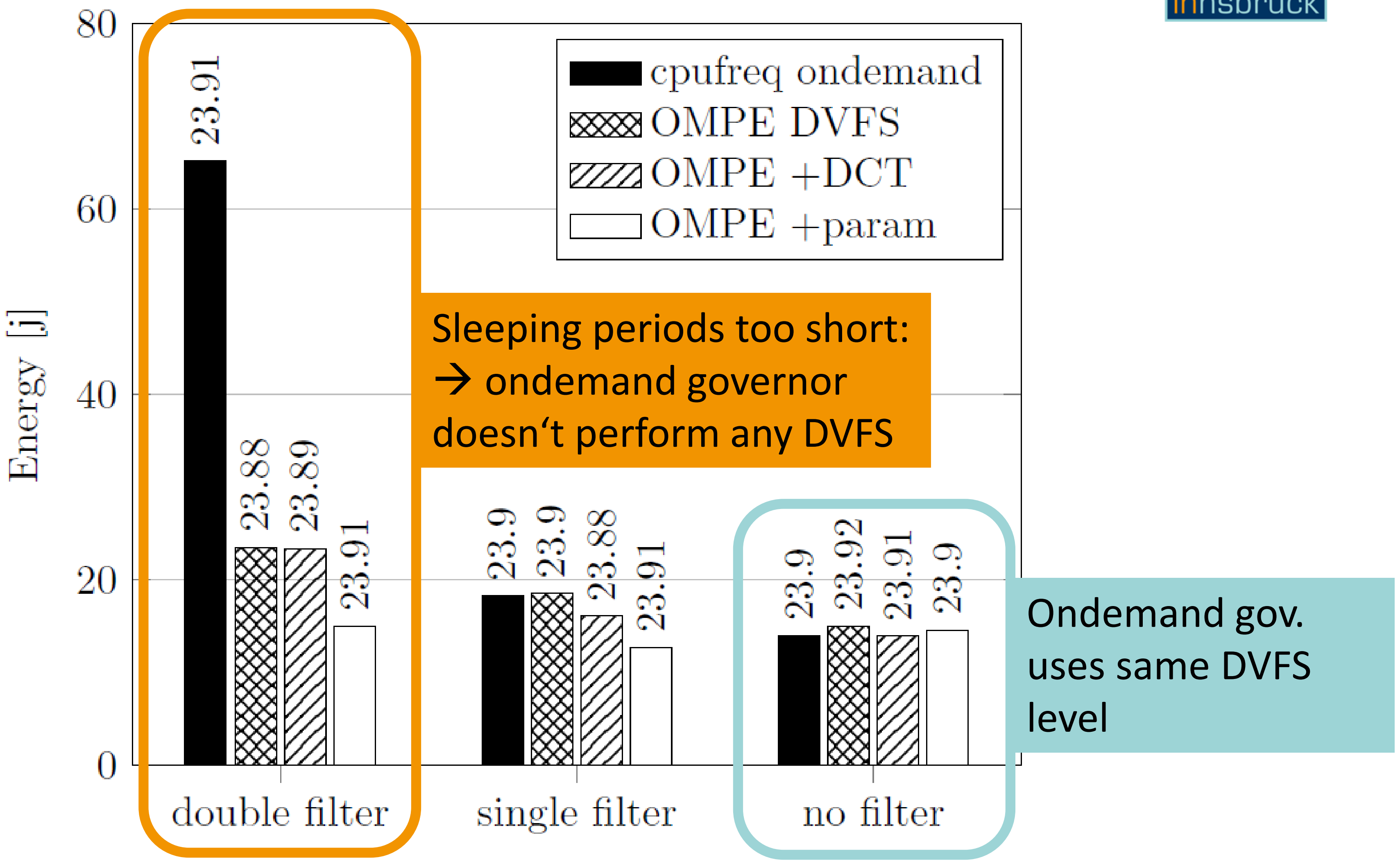
```
#pragma omp parallel for schedule(dynamic)        \
    objective(E : T < 1 / f_rate; Q<3)            \
    param(scaling, range(1,8,1))
for (int y=0; y<rows; y+=2*scaling)
    for (int x=0; x<cols_2; x+=scaling)
        ...
```

- Read as:
  - Optimize for energy
  - While maintaining the target framerate
  - Allow setting the scaling parameter up to 2

Results - *desktop*



Results - *mobile*



Discussion

Related Work

- Energy management has been investigated in and across all components of the system software stack
  - Also in combination with OpenMP and DVFS/DCT
  - Generally does not take into account application-level knowledge!

- Application-level approaches are often domain-specific, and usually per-program rather than per-region

Problems / Future Opportunities

- How to deal with co-running applications?
  - Need OS-level component or user-space orchestrator
  - May have conflicting goals and distinct optimal configurations
    - SLAs?
- Limited optimization strategy
  - How can we optimize regions which only run once or a small number of times?
- Improve user-define tunable parameter support
  - Is a simple mapping to a single quality metric sufficient?
  - Provide guidance to expected impact on metrics

Conclusion

- We propose an interface for...
  - Setting per-region multi-objective *goals* and *constraints*
  - Exposing *application-level tunable parameters* to the runtime system
- Compared to previous work, it‘s more generic and flexible as well as easier to integrate into existing code bases
- Our prototype implementation shows possibilities for energy savings in several scenarios
  - Requiring only two clauses on the main parallel loop

Thank You!

http://www.insieme-compiler.org

https://github.com/insieme/insieme

petert@dps.uibk.ac.at

- Questions?