A Programming Model and Runtime System for Significance-Aware Energy-Efficient Computing

V. Vassiliadis[1,2], K. Parasyris[1,2], C. Chalios[3],

C. Antonopoulos[1,2], S. Lalis[1,2], N. Bellas[1,2],

H. Vandierendonck[3] D. Nikolopoulos[3]

[1]Department of Electrical and Computer Engineering

University of Thessaly

[2]Centre for Research and Technology Hellas

(CE.R.T.H.), Greece

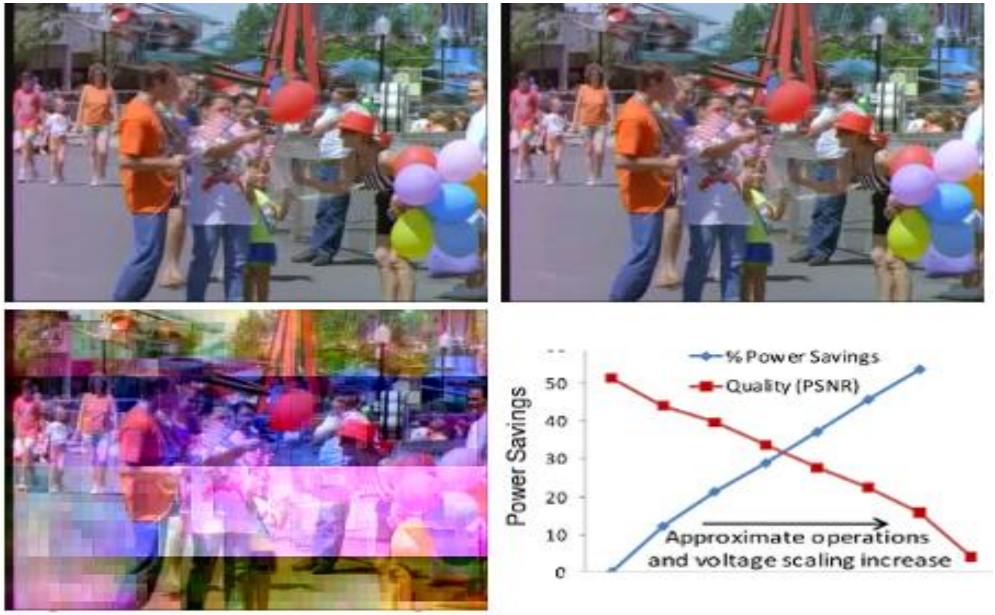
[3]Queen's University Belfast

United Kingdom

Motivation

- Energy consumption has become a major barrier.
- Many applications are adaptive to approximations.
  - ➢Different parts of the same application have different "significance" for the quality of the end-result.
  - ➢Multimedia, scientific computing, communication, visualization apps can be approximated.

iDCT algorithm with varying degree of accuracy.



Objectives

We would like to provide mechanisms that allow the programmer to:

- Express the *significance* of computations in terms of their contribution to the quality of the end result;
- Specify approximate alternatives for selected computations;
- Express parallelism, beyond significance;
- Control the balance between energy consumption and the quality of the end-result.

Programming Model Example

```
void DCT(unsinged char *img, double *dct_out){
        /* Significance look up table for each 2x4 sub-block */
        float sgnf_lut[] = {1.0, 0.9, 0.7, 0.3, 0.8, 0.4, 0.3, 0.1};
        for each 2x4 sub-block K {
                #pragma omp task label(dct) in(img) out(dct_out)
                significance(expr(sgnf_lut[K])) approxfun(NULL)
                dct_task(...);
        }
        #pragma taskwait label(dct) ratio(0.8)
}
```

Express Parallelism

```
void DCT(unsinged char *img, double *dct_out) {

        float sgnf_lut[] = {1.0, 0.9, 0.7, 0.3, 0.4, 0.8, 0.4, 0.3, 0.1};
        for each 2x4 sub-block K {
                #pragma omp task label(dct) in(img) out(dct_out)
                significance(expr(sgnf_lut[K])) approxfun(NULL)
                dct_task(...);
        }
        #pragma taskwait label(dct) ratio(0.8)
}
```

1) **Task based programming model.**
2) **Parallelism is implicitly declared by annotating a tasks memory footprint**

Approximation Extensions

```
void DCT(unsinged char *img, double *dct_out) {
        float sgnf_lut[] = {1.0, 0.9, 0.7, 0.3, 0.8, 0.4, 0.3, 0.1};
        for each 2x4 sub-block K {
                #pragma omp task label(dct) in(img) out(dct_out)
                significance(expr(sgnf_lut[K])) approxfun(NULL)
                dct_task(...);
        }
        #pragma taskwait label(dct) ratio(0.8)
}
```

Subscribe a task into a group of tasks identified by a string

Approximate alternative for selected functions.

Define the *significance* of computations based on their impact on the output's quality.

Synchronization Extensions

```
void DCT(unsinged char *img, double *dct_out){
        /* Significance look up table for each 2x4 sub-block */
        float sgnf_lut[] = {1.0, 0.9, 0.7, 0.3, 0.8, 0.4, 0.3, 0.1};
        for each 2x4 sub-block K {
                #pragma omp task label(dct) in(img) out(dct_out)
                significance(expr(sgnf_lut[K])) approxfun(NULL)
                dct_task(...);
        }
        #pragma taskwait label(dct) ratio(0.8)
}
```

Control the balance between energy consumption and the quality of the end-result using a single clause.

Wait for all tasks subscribed in the *"dct" group*

Runtime Support Approximate Computing

The runtime should respect:

- **The significance of each task.**
- **The fraction of tasks that may be executed approximately for each task group.**

Obstacles:

- **No information on how many tasks will be issued in a task group.**
- **Unknown distribution of significance levels in each task group.**

Significance Aware Scheduling Policies

Global Task Buffering (GTB):

- ➢ Buffers issued tasks and analyzes their properties

Local Queue History (LQH):

- ➢ Estimates the distribution of significance levels using per-worker local information.

Policy	1st Concern	Execution Decision
GTB	Quality	Main Thread
LQH	Performance	Worker Thread

Experimental Evaluation

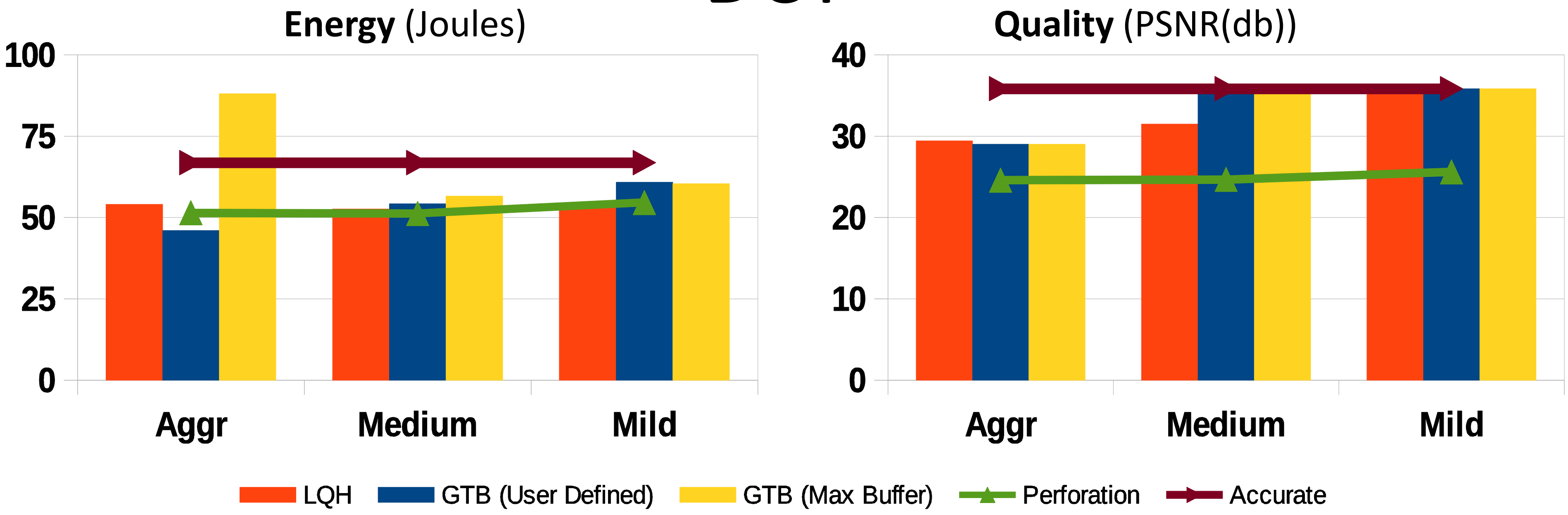
Benchmark	Quality	Approximation Degree		
		Mild	Mid	Aggressive
Sobel	PSNR(db)	10%	30%	80%
DCT	PSNR(db)	10%	40%	80%

Benchmarks used for the evaluation. For all cases, the degree of approximation is given by the percentage of tasks executed approximately.

We compare our results with executions using perforation:

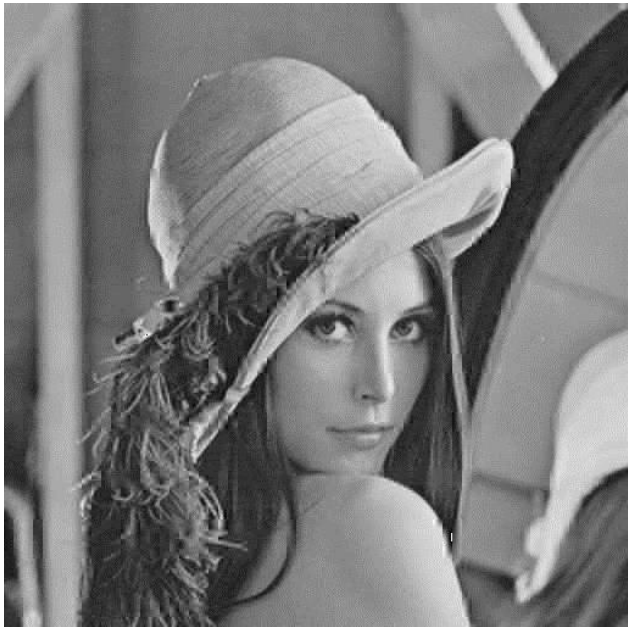
- Perforation is a compiler technique that removes loop-steps.

DCT

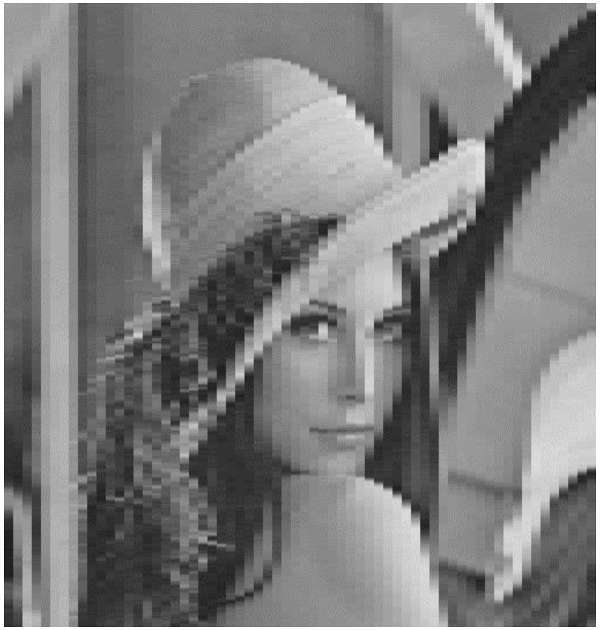


Energy (Joules)

Quality (PSNR(db))

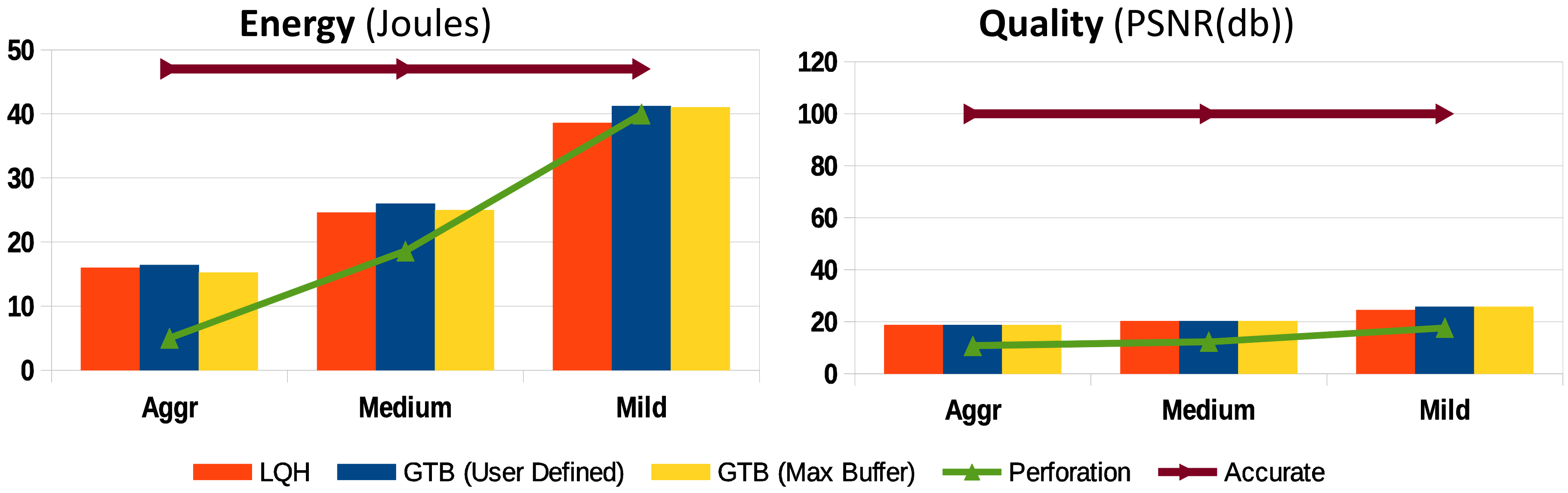


Aggressive significance aware output



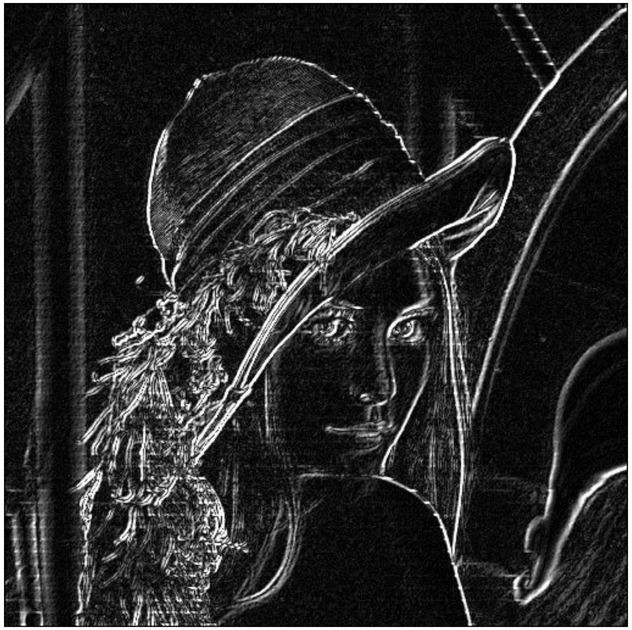
Aggressive perforated output

Sobel

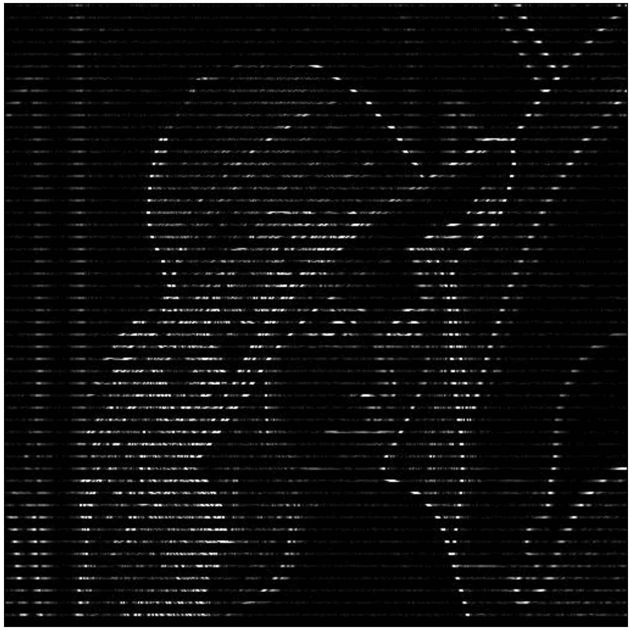


Energy (Joules)

Quality (PSNR(db))



Aggressive significance aware output



Aggressive perforated output

Conclusions

- **Developed a programming model that supports approximate computing at the granularity of tasks.**

- **Introduced extensions to a task-based runtime system to exploit significance information.**

- **Presented Significance-centric scheduling policies**

Questions

Acknowledgements

This work has been partially supported by the "Aristeia II" action (Project "Centaurus") of the operational program Education and Lifelong Learning and is co-funded by the European Social Fund and Greek national resources.

European Union
European Social Fund

OPERATIONAL PROGRAMME
EDUCATION AND LIFELONG LEARNING
investing in knowledge society
MINISTRY OF EDUCATION & RELIGIOUS AFFAIRS, CULTURE & SPORTS
MANAGING AUTHORITY

NSRF
2007-2013
programme for development
EUROPEAN SOCIAL FUND

Co-financed by Greece and the European Union

This work has been partially supported by the EC within the 7th Framework Program under the FET-Open grant agreement SCoRPiO, grant number 323872.

SCoRPiO
Significance-Based Computing for
Reliability and Power Optimization

SEVENTH FRAMEWORK
PROGRAMME