Time-Predictable Task-to-Thread Mapping in Multi-Core Processors

Presenter:

Mohammad Samadi

Co-authors:

Sara Royuela, Luis Miguel Pinho, Tiago Carvalho, Eduardo Quiñones

School of Engineering, Polytechnic Institute of Porto, Portugal

Barcelona Supercomputing Center, Barcelona, Spain

Universitat Politècnica de Catalunya, Barcelona, Spain

June 2023

isep Instituto Superior de Engenharia do Porto

BSC Barcelona Supercomputing Center Centro Nacional de Supercomputación

UPC UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

Agenda

- Introduction
- Motivations
- Contributions
- Background
- Proposed Mapping Method
- Simulation Results
- Implementation Results
- Future Works

Introduction

Real-Time Systems

Demand

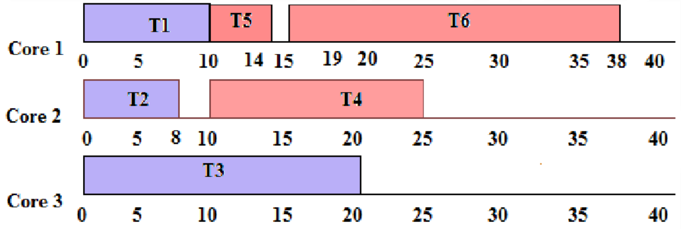
Modern Platform

Requirements

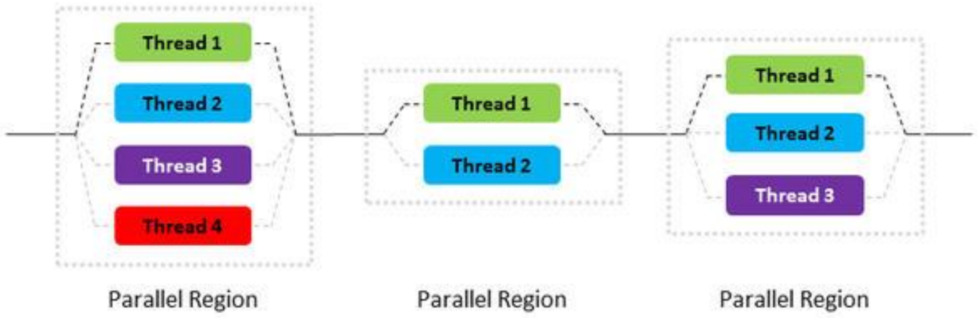
- Meet application deadline
- Minimize response time
- Enhance time-predictability

Use most of capacity

Task Scheduling



Task-to-thread mapping



Parallel Framework (e.g., OpenMP)

Motivations

Main issues with current mapping algorithms

Mostly do not consider temporal conditions, causing an increase in the application's response time

Mostly do not consider execution variability, requiring pessimistic analysis techniques

Contributions

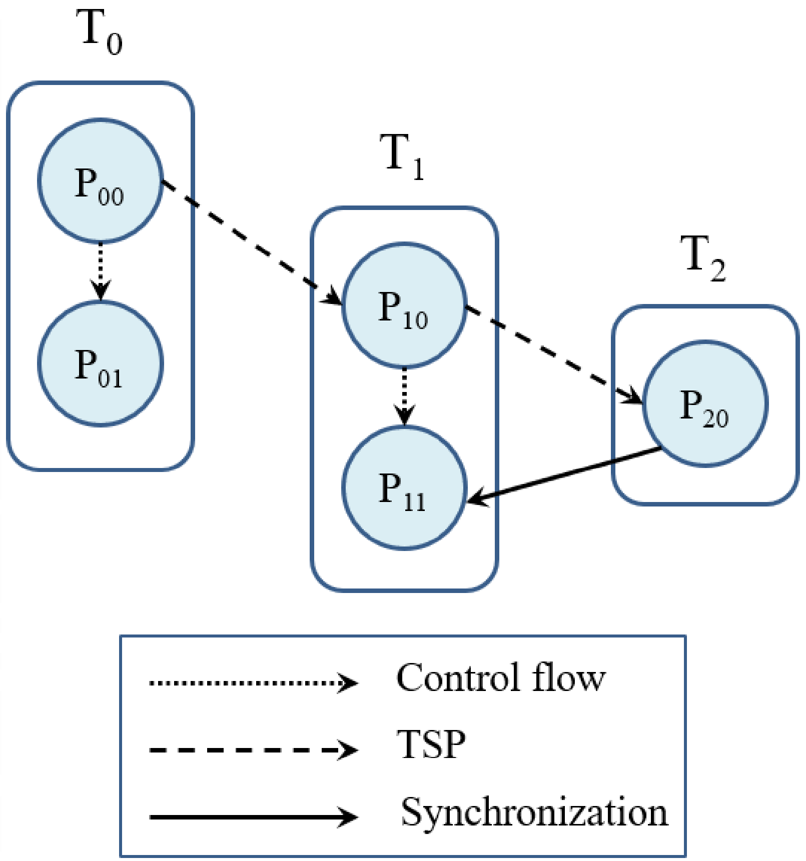
1. Reduce the contention by using distributed queues
2. A set of different heuristics for efficient task-to-thread mapping
3. Simulation-based analysis of the heuristics with/without runtime overhead
4. LLVM-based implementation and evaluation

Background

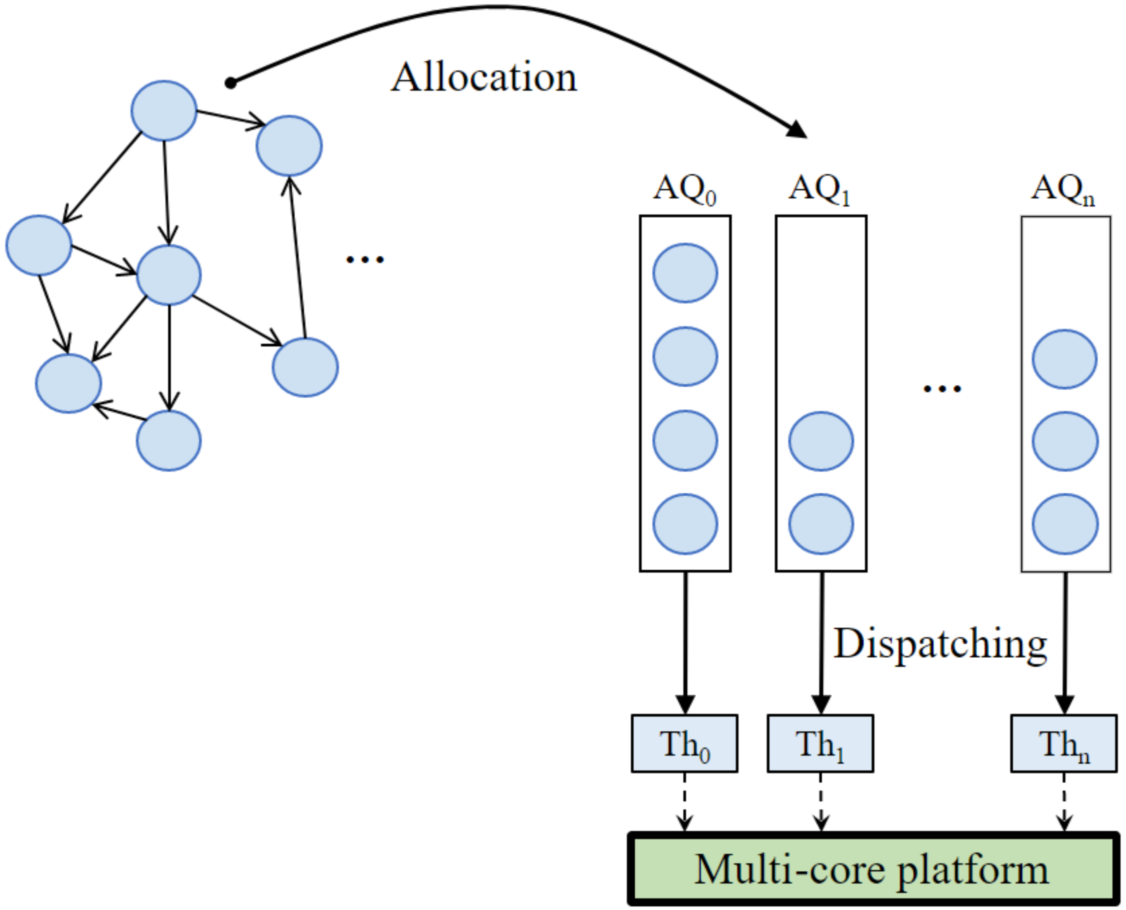
Source Code

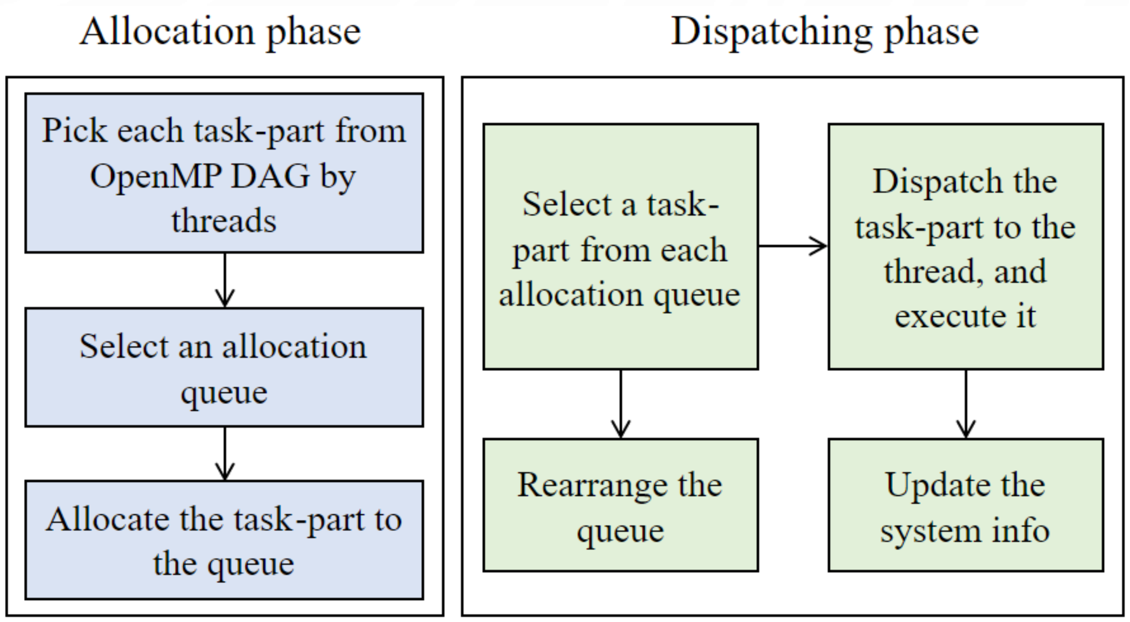
```
#pragma omp parallel num_threads (m)
{
    #pragma omp single // T0
    {
        code00;
        #pragma omp task // T1
        {
            code10;
            #pragma omp task // T2
            {
                code20;
            }
            #pragma omp taskwait;
            code11;
        }
        code01;
    }
}
```

Directed Acyclic Graph (DAG)



Proposed Mapping Method





Proposed Mapping Method

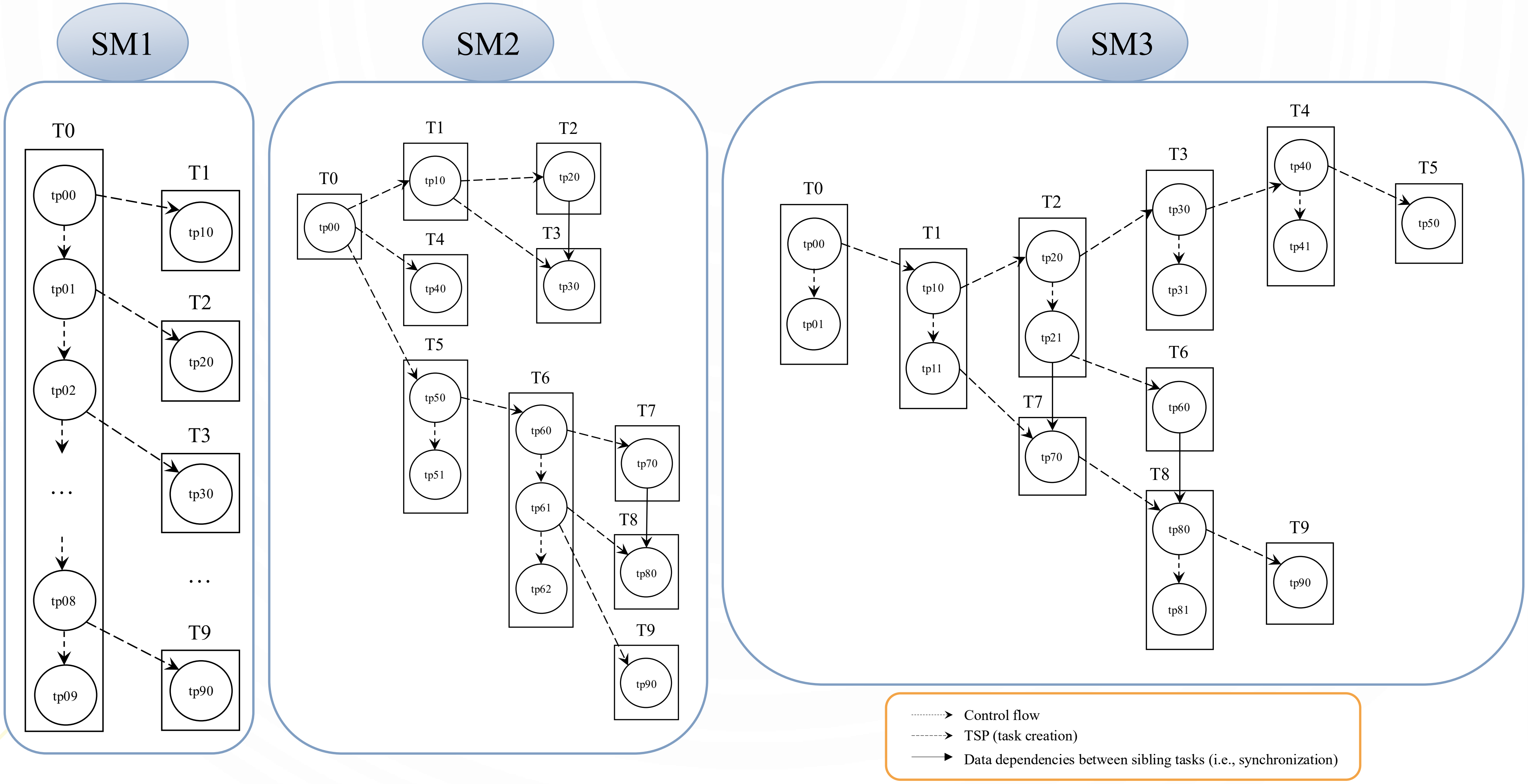
The allocation heuristics to select threads

MNTP	Minimum Number of Task-Parts
NT	Next Thread
MRIT	Most Recent Idle Time
MTET	Minimum Total Execution Time
MTRT	Maximum Total Response Time
TMCD	Total Multi-Criteria Decision based on MNTP, MRIT and MTET

The dispatching heuristics to select task-parts

MET	Minimum Execution Time
MRT	Maximum Response Time
MCD	Multi-Criteria Decision based on MET and MRT

Simulation Results
(System Model)



Simulation Results
(Including Overhead)

Performance of MTET-MET compared to the others with 8 threads

System Model	Tied Tasks			Untied Tasks		
	BFS	WFS	LNSNL	BFS	WFS	LNSNL
SM1	-4.03%	55.99%	-0.19%	-3.64%	39.21%	33.66%
SM2	20.13%	87.67%	37.29%	18.77%	43.81%	49.62%
SM3	19.88%	88.64%	49.06%	23.19%	47.49%	67.01%
Average:	11.99%	77.43%	28.72%	12.77%	43.50%	50.10%

Evaluation

1. MTET-MET outperforms BFS, except for SM1.
2. MTET-MET outperforms WFS, more effective for tied tasks.
3. MTET-MET works better than LNSNL, except for SM1 with tied tasks.

Simulation Results
(Without Including Overhead)

Performance of MTET-MET compared to the others with 8 threads

System Model	Tied Tasks			Untied Tasks		
	BFS	WFS	LNSNL	BFS	WFS	LNSNL
SM1	0%	49.90%	0%	2.39%	11.79%	2.42%
SM2	4.61%	86.40%	3.36%	11.39%	28.18%	3.35%
SM3	6.60%	86.92%	4.72%	13.12%	29.65%	4.98%
Average:	3.74%	74.41%	2.69%	8.97%	23.21%	3.58%

Evaluation

1. MTET-MET works better than BFS and LNSNL, except for SM1 with tied tasks.
2. WFS clearly works worser than the other methods, especially for tied cases.

Simulation Results
(Discussion)

With Overhead

- High efficiency of the new method
- Improving state-of-the-art online mapping mechanisms

Without Overhead

- Equivalent or better performance of the new mapping
- Validating its use in off-line mapping scenarios

Implementation Results
(Implementation of the Heuristics)

Implementing three heuristics within the LLVM-based OpenMP runtime

- MNTP
- MTET
- MET

Evaluating the performance of two pairs of the heuristics

- MNTP-MET
- MTET-MET

Implementation Results
(Applications)

Heat

Heat diffusion simulator implemented with Gauss-Seidel method, showing a Stencil computation

Number of tasks: 640
Number of data dependencies: 2128

SparseLU

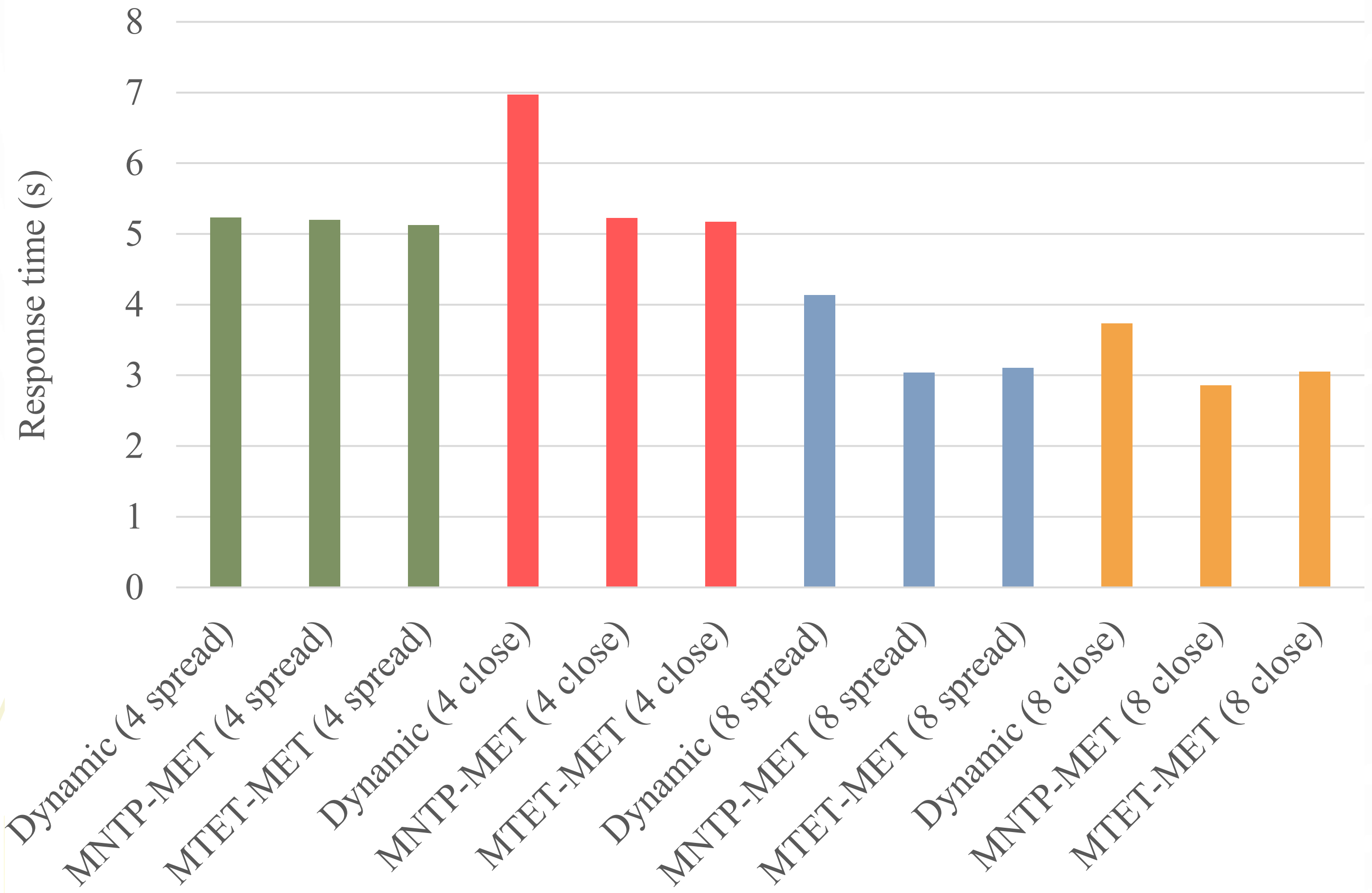
SparseLU matrix decomposition, showing an irregular form of parallel tree and retracting to one final task

Number of tasks: 1496
Number of data dependencies: 3960

Implementation Results
(Setup)

Platform	NVIDIA Jetson AGX Xavier
Configuration	• Number of threads: 4, 8 • Binding OS threads to cores: Spread, Close
OS	Linux based on the highest real-time priority (SCHED_FIFO) class
Execution	• Execute the applications in *r* runs, and *i* iterations on each run • Remove outliers

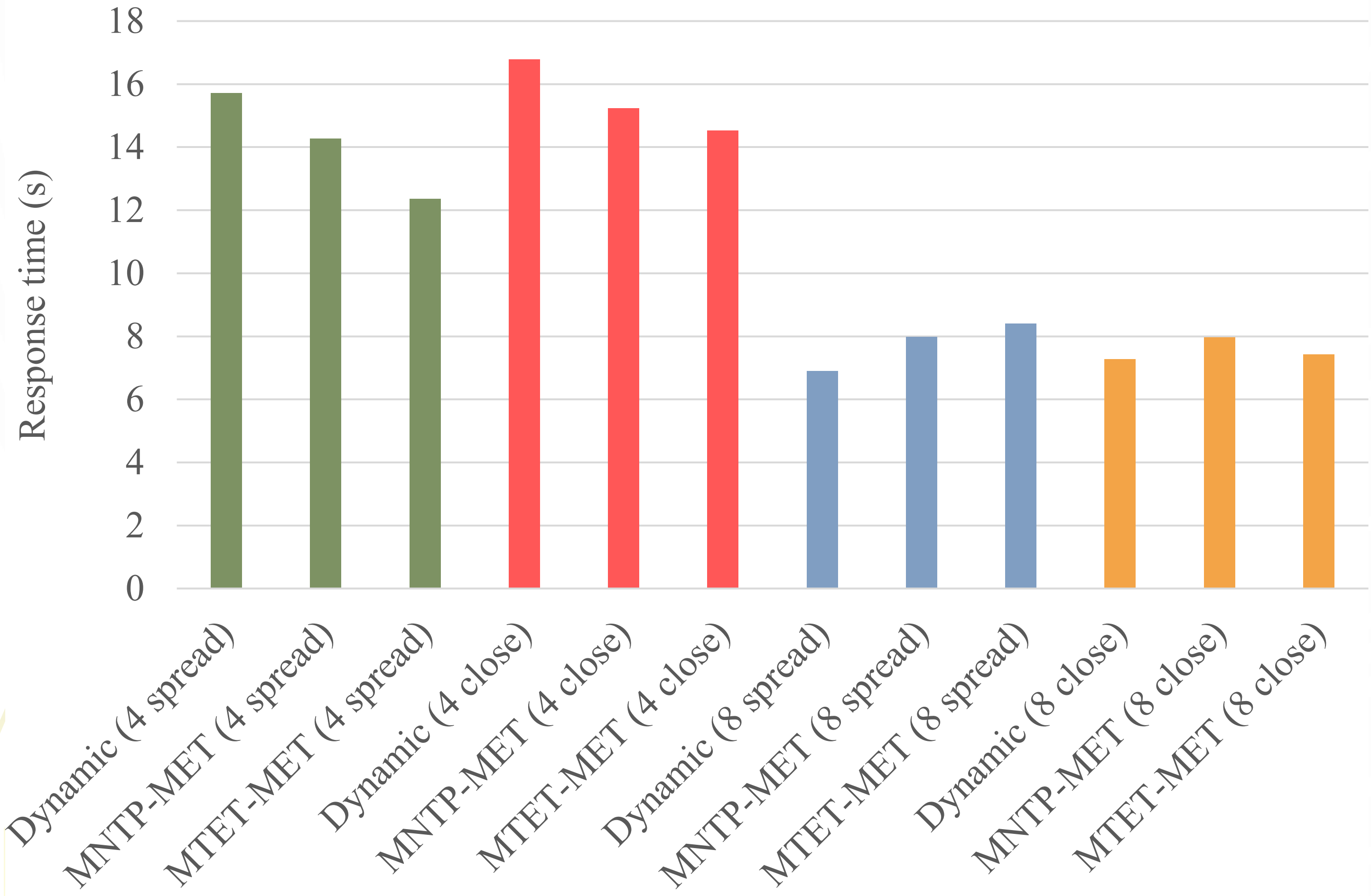
Implementation Results
(Heat Application – Experiments)



Evaluation

1. The algorithms are scalable.
2. The heuristics works more efficient than Dynamic in most of the case.
3. MTET-MET works slightly better than MNTP-MET with 4 threads.
4. MNTP-MET works slightly better than MTET-MET with 8 threads.

Implementation Results
(SparseLU Application – Experiments)



Evaluation

1. The algorithms are scalable.
2. The performance of the heuristics is higher than Dynamic with 4 threads.
3. Dynamic works slightly better with 8 threads.
4. MTET-MET works better than MNTP-MET, except in '8 spread'.
5. The efficiency of MTET overcomes its weakness on overhead.

Implementation Results
(Discussion)

Minimization of response time by the heuristics

Higher efficiency of MTET-MET than MNTP-MET

Future Works

1

The integration of tied and untied tasks in the same application

2

Evaluation of the new method using real-world use cases

3

The consideration of heterogeneous systems

Thanks for your attention!

For more information:
mmasa@isep.ipp.pt

isep Instituto Superior de Engenharia do Porto
BSC Barcelona Supercomputing Center Centro Nacional de Supercomputación
UPC UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

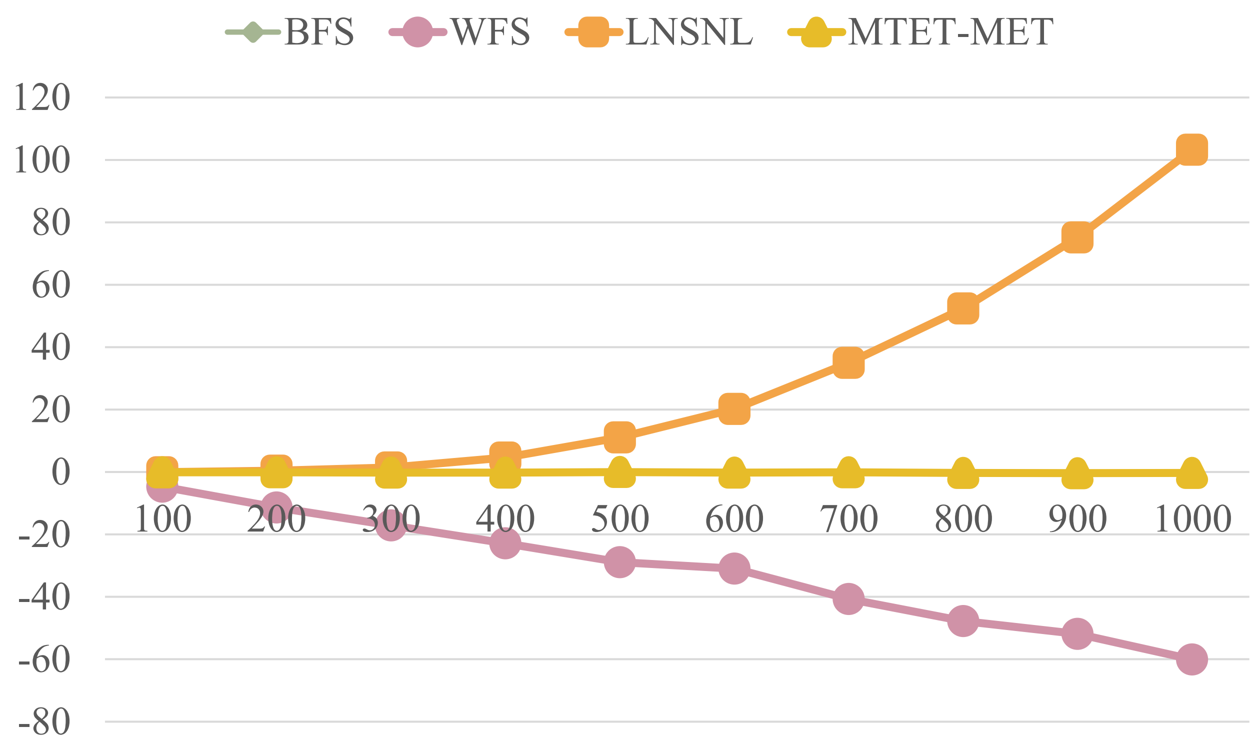
Appendix
(Simulation Results – Including Overhead)

Performance of MTET-MET compared to the others with 4 and 8 threads

System model	Number of threads	Tied tasks			Untied tasks		
		BFS	WFS	LNSNL	BFS	WFS	LNSNL
SM1	4	-2.52%	56.04%	-0.01%	-2.23%	21.71%	13.49%
	8	-4.03%	55.99%	-0.19%	-3.64%	39.21%	33.66%
SM2	4	10.03%	78.28%	22.12%	10.29%	37.25%	34.79%
	8	20.13%	87.67%	37.29%	18.77%	43.81%	49.62%
SM3	4	10.98%	77.99%	29.53%	13.56%	37.89%	51.42%
	8	19.88%	88.64%	49.06%	23.19%	47.49%	67.01%
	Average:	9.08%	74.10%	22.97%	9.99%	37.89%	41.67%

Appendix
(Simulation Results – Including Overhead)

Effect of the number of tasks on the difference between response time for tied and untied tasks in SM3 with 8 threads



$$Diff = RT_u - RT_t$$

Evaluation

1. The response time given by WFS for untied tasks is lower than tied tasks.
2. The response time given by LNSNL for tied tasks is lower than untied tasks.
3. The difference between BFS and MTET-MET is not very noticeable.

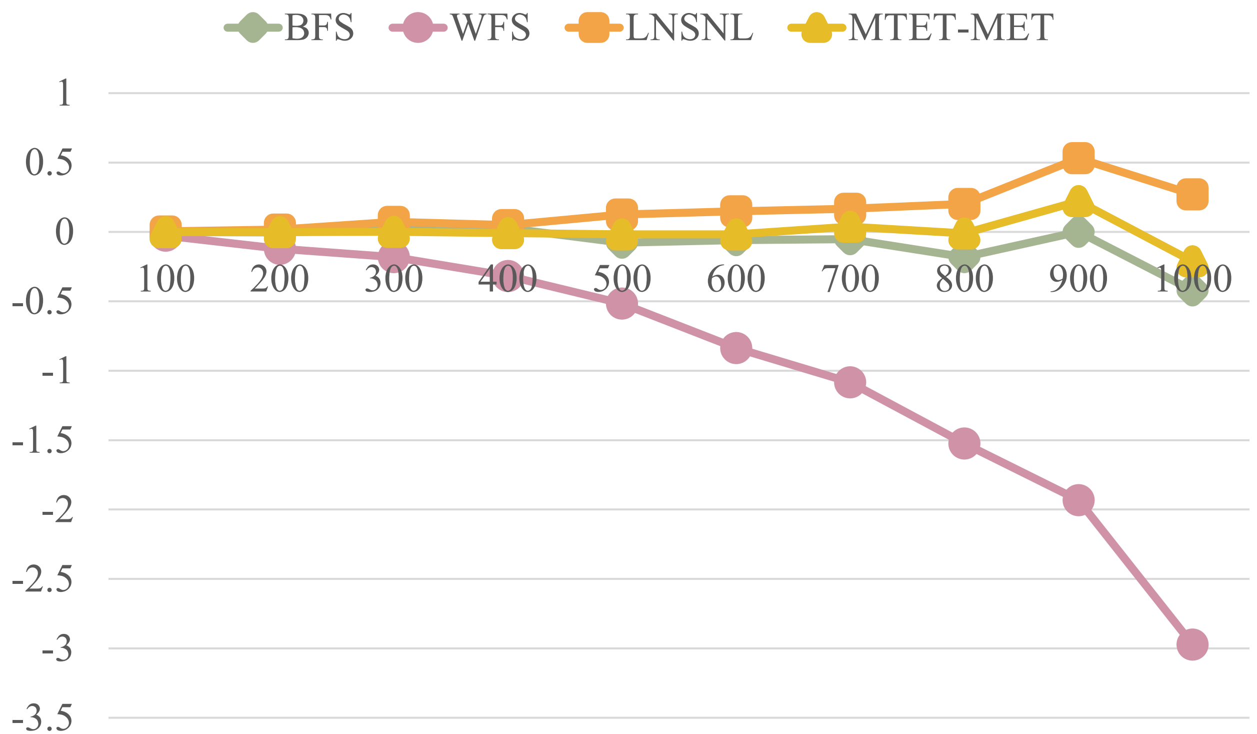
Appendix
(Simulation Results – Without Including Overhead)

Performance of MTET-MET compared to the others with 4 and 8 threads

System model	Number of threads	Tied tasks			Untied tasks		
		BFS	WFS	LNSNL	BFS	WFS	LNSNL
SM1	4	0%	49.82%	0%	1.43%	11.41%	1.41%
	8	0%	49.90%	0%	2.39%	11.79%	2.42%
SM2	4	10.78%	74.47%	4.91%	18.87%	31.87%	4.75%
	8	4.61%	86.40%	3.36%	11.39%	28.18%	3.35%
SM3	4	8.92%	74.02%	2.82%	19.09%	32.11%	4.89%
	8	6.60%	86.92%	4.72%	13.12%	29.65%	4.98%
Average:		5.15%	70.26%	2.64%	11.05%	24.17%	3.63%

Appendix
(Simulation Results – Without Including Overhead)

Effect of the number of tasks on the difference between running time for tied and untied tasks in SM1 with 4 threads



$Diff = RT_u - RT_t$

Evaluation

1. The running time for tied tasks, except LNSNL, is mostly higher than untied tasks.
2. The difference between tied and untied cases in WFS is higher than the other methods.
3. The difference is very considerable in most cases as the number of tasks increases.