UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

Presentation of the dissertation submitted to the Department of Computer Architecture at Technical University of Catalonia

Mapping of Real-Time Computation to Parallel Platforms

PhD Candidate:

Mohammad Samadi Gharajeh

Supervisors:

Prof. Luís Miguel Pinho (ISEP)

Dr. Sara Royuela Alcázar (BSC)

Tutor:

Prof. Xavier Martorell Bofill (UPC)

ISEP INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

February 19th, 2026

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Outline

- Context, Vision, and Objectives of the Thesis
- OpenMP for Real-Time Systems
- Proposed Task-to-Thread Mapping for Homogeneous Platforms
- Runtime Implementation in LLVM and Empirical Validation
- Schedulability Analysis
- Proposed Task-to-Accelerator Mapping for Heterogeneous Platforms
- Overall Conclusions and Thesis Impact

The Context of Modern Critical Systems

Critical Systems (e.g., Drone and Automotive Industry)

Requirements

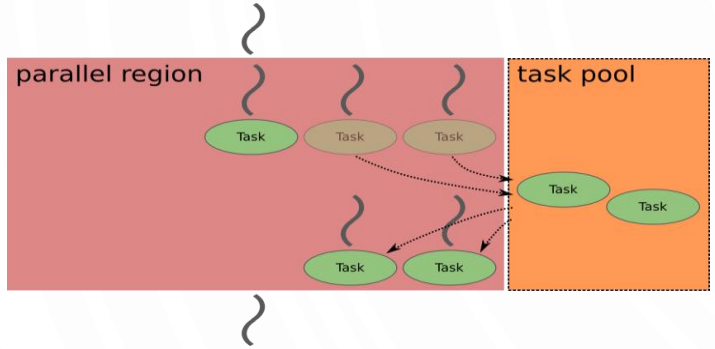
Hardware Capabilities

- Guaranteeing **predictability** and **schedulability**
- Reducing **response time** and **WCRT**
- Reducing **response time variability**
- Enhancing **high-performance** capability

Modern Platform (e.g., Jetson AGX Xavier)

Thesis Visions/Goals

The availability of **portable, performant and easily programmable parallel computing** paradigms

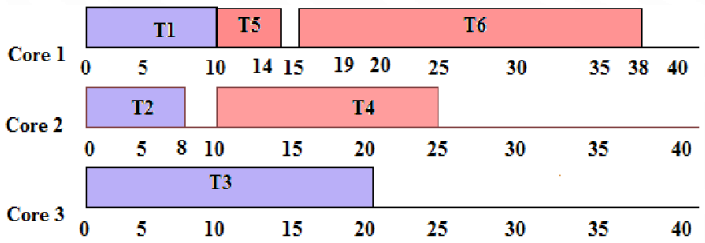


Why **OpenMP?**

OpenMP provides **HPC capabilities** in heterogeneous and highly parallel systems

OpenMP tasking allows **real-time-systems-compatibility**, enabling predictability and schedulability analyses

The need for **real-time HPC behavior** in the task-to-thread mapping



Needs

Improving **work-conservation** of the mapping

Improving **load-balancing** of the queues

Reducing **contention** on queues

Minimizing application **response time**

Thesis Objectives

1. Propose a **task-to-thread mapping** to **guarantee** system **predictability** and **schedulability**, and **reduce WCRT** and **variability**.

2. **Implement** and **evaluate** the **task-to-thread mapping** extending an existing **LLVM**-based OpenMP framework*.

3. **Analyze** the effect of **graph complexity**, **number of OpenMP threads**, and **task-to-core binding** on the mapping process.

4. **Evaluate** the use of **static** (offline) versus **dynamic** (online) **task-to-thread mapping** for parallel applications.

5. **Propose** and **evaluate** an approach for predictable **task-to-accelerator mapping** on **heterogeneous** platforms.

* A. Munera, et al. *Towards a qualifiable OpenMP framework for embedded systems*. DATE 2020.

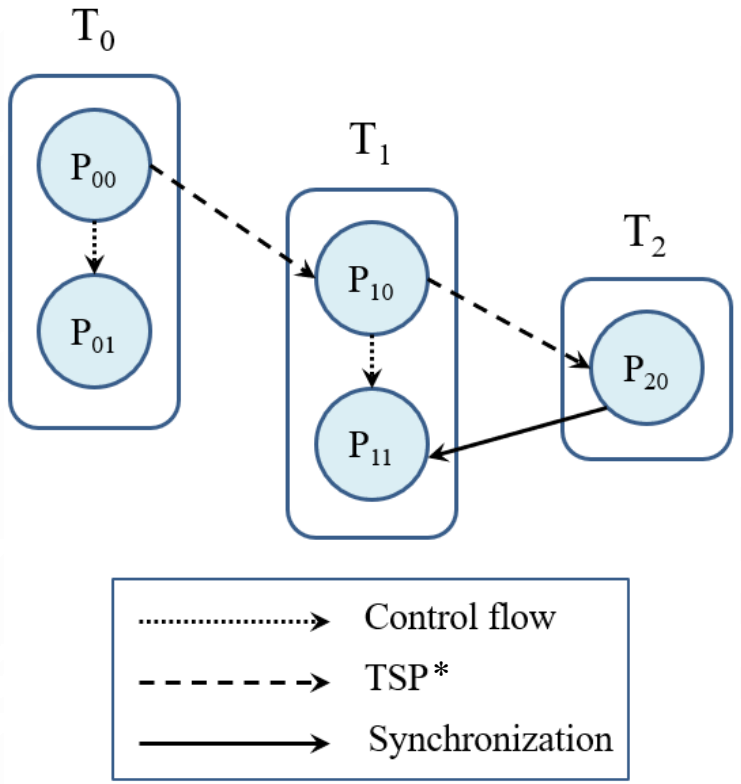
OpenMP for Real-Time Systems

OpenMP Tasking

Source Code

```
#pragma omp parallel num_threads(N)
{
    #pragma omp single              // T0
    {
        P00;
        #pragma omp task            // T1
        {
            P10;
            #pragma omp task        // T2
                P20;
            #pragma omp taskwait
            P11;
        }
        P01;
    }
}
```

Directed Acyclic Graph (DAG)

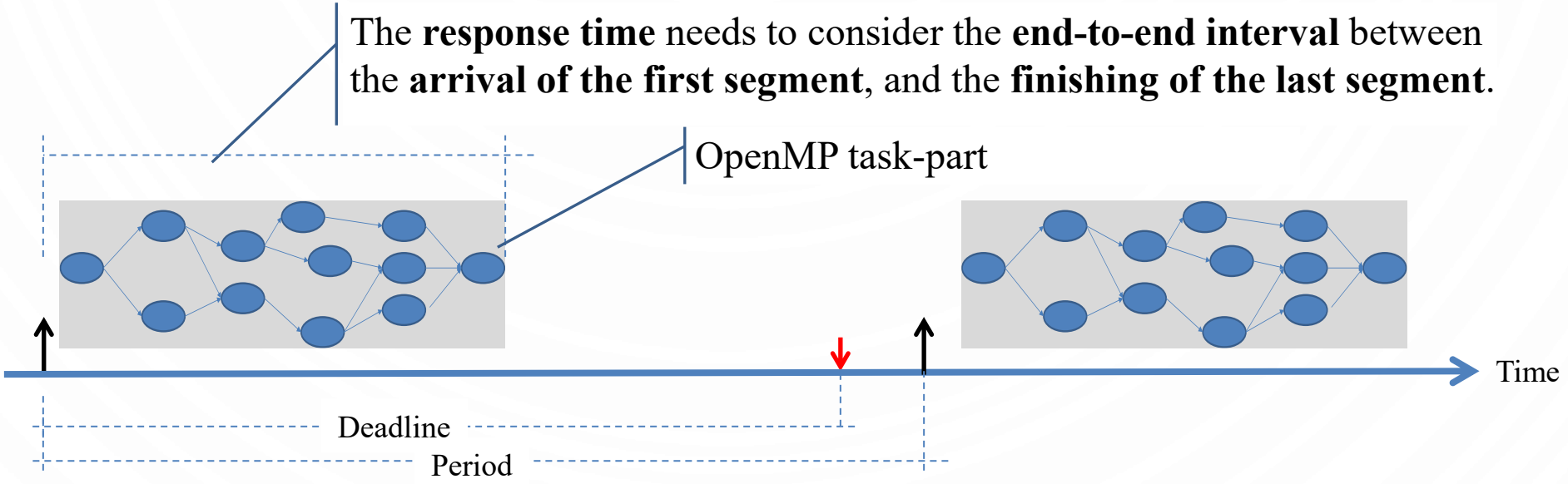


* Task Scheduling Point (TSP): A point during the execution of the current task region at which the task can be suspended to be resumed later

Real-Time Tasking

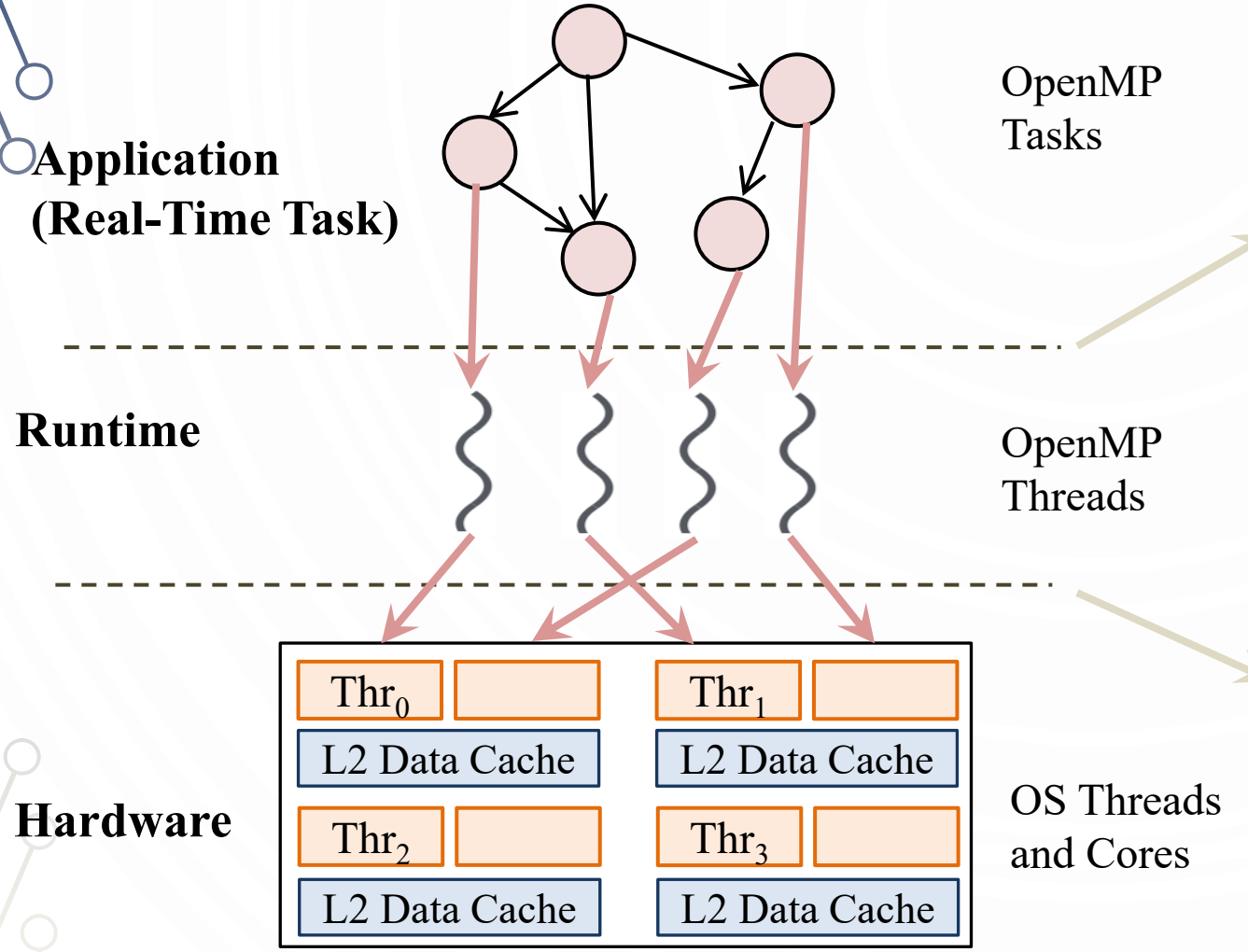
An application includes a **real-time task, with internal parallelism** using **OpenMP tasking**

A **real-time task** is a **recurrent (periodic) activity**, with characteristics such as a **period**, **deadline**, **execution time**, and **worst-case response time**.



When considering **multiple concurrent real-time tasks (multiple DAGs)**, the **response time** will also be impacted with **inter-task interference**. Except where noted, the thesis considers applications have a **single parallel real-time task (single DAG).**

OpenMP Tasking: Scheduling and Mapping



Scheduling: `tied` / `untied`

Tied tasks must be executed by the **same thread** starting them, while **untied** tasks can be resumed by **different threads**.

Binding*: `spread` / `close`

In **spread** binding, **OS threads** are **distributed** across **cores**, while in **close** binding, they are **bound** to the **closest core**.

* Other binding options are `false` (disable binding) and `primary` (all threads are bound to the same core).

OpenMP Efforts to Adapt to Real-Time

2011	*ompVerify: Polyhedral analysis for the OpenMP programmer*	IWOMP
2014	*Classification of common errors in OpenMP applications*	IWOMP
2015	*Compiler analysis for OpenMP tasks correctness*	CF
	Timing characterization of OpenMP4 tasking model	CASES
2017	*A functional safety OpenMP for critical real-time systems*	IWOMP
	Benchmarking OpenMP programs for real-time scheduling	RTCSA
2018	*Using polyhedral analysis to verify OpenMP applications are data race free*	Correctness
	Towards an OpenMP specification for critical real-time systems	IWOMP
2020	*Real-time scheduling and analysis of OpenMP DAG tasks supporting nested parallelism*	TC
2021	*Verifying pipeline implementations in OpenMP*	SPIN
2022	*ompTG: From OpenMP programs to task graphs*	JSA
2023	*Framework for the analysis and configuration of real-time OpenMP applications*	INDIN
2024	*Event-based OpenMP tasks for time-sensitive GPU-accelerated systems*	IWOMP
	OpenMP-RT: Native pragma support for real-time tasks and synchronization with LLVM under Linux	LCTES
2025	*A comparative analysis of OpenMP and CUDA in real-time applications*	ICCMC

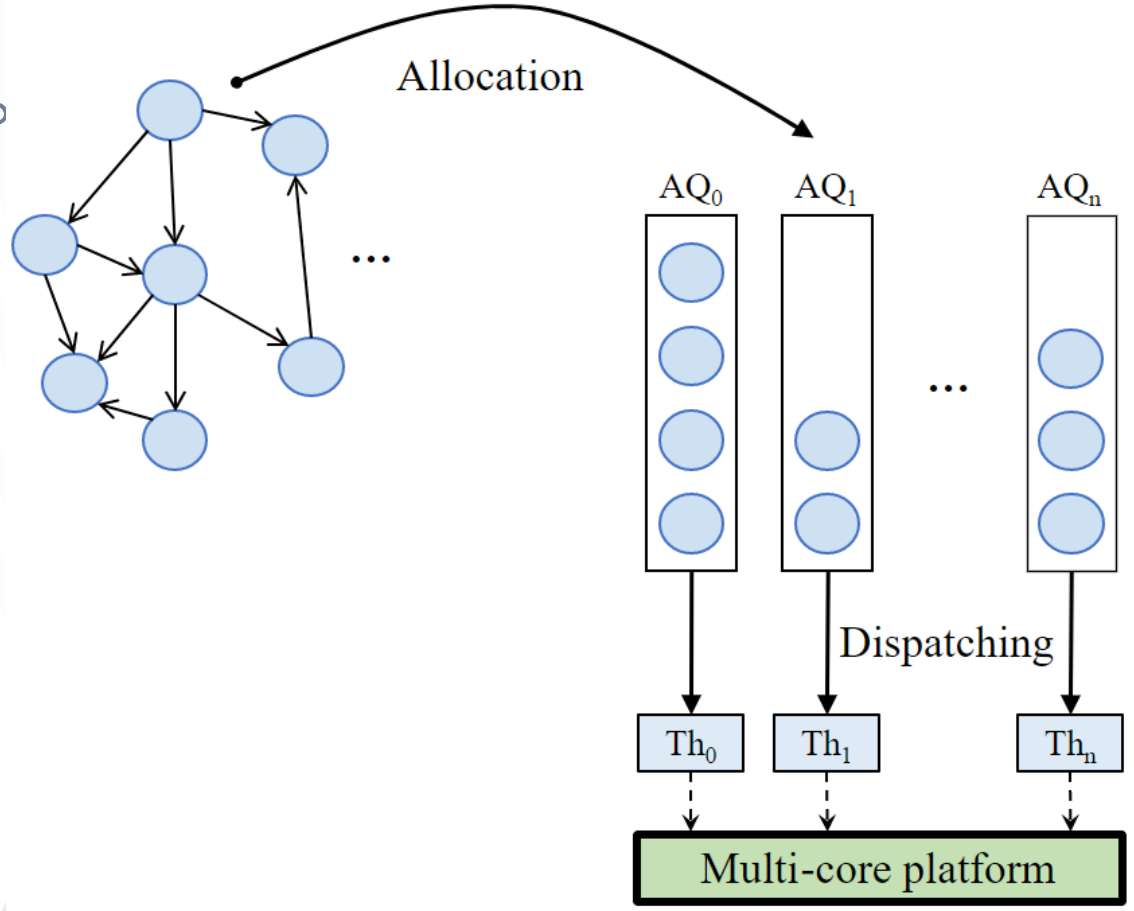
Functional verification

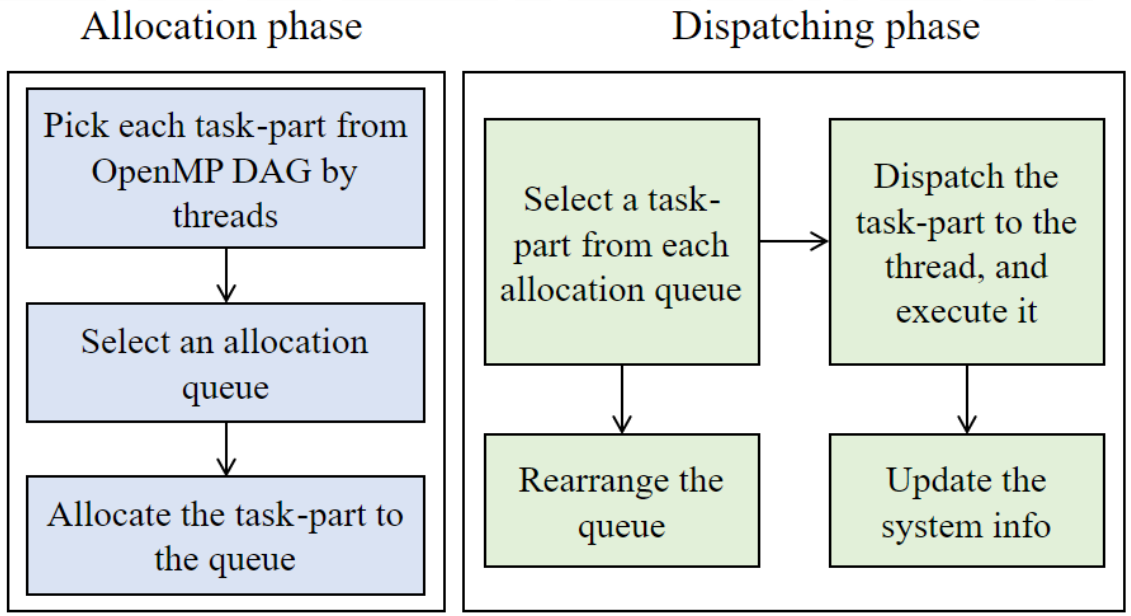
Real-time analysis

Proposed Task-to-Thread Mapping for Homogeneous Platforms

Methodology

Objective 1





M. Samadi, et al. *Heuristic-based task-to-thread mapping in multi-core processors.* ETFA WiP, 2022.

M. Samadi, et al. *Time-predictable task-to-thread mapping in multi-core processors.* Journal of Systems Architecture (JSA), 2024.

Allocation and Dispatching Algorithms

Allocation Heuristics

MNTP	Minimum Number of Task-Parts	**MNTP** and **NT** increase the **load balancing** of queues based on the **number of tasks**.
NT	Next Thread	
MRIT	Most Recent Idle Time	**MRIT** enhances the **workload** of queues by decreasing the **threads' idle time.**
MTET	Minimum Total Execution Time	**MTET** improves the **load balancing** of queues based on **execution time.**
MTRT	Maximum Total Response Time	**MTRT** reduces queues **load balancing** to evaluate system **performance against worst-case** condition.
TMCD	Total Multi-Criteria Decision	**TMCD** combines **MNTP**, **MRIT**, and **MTET.**

Dispatching Heuristics

MET	Minimum Execution Time	**MET** increases the number of **ready tasks** by reducing the time to meet dependencies.
MRT	Maximum Response Time	**MRT** reduces the **workload** of queues, enhancing the chances of getting new tasks.
MCD	Multi-Criteria Decision	**MCD** combines **MET** and **MRT**.

Applications for Empirical Evaluation

Axpy

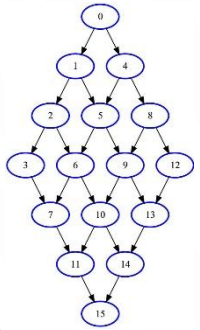
Embarrassingly parallel.
Basic linear algebra operations ($ax*y$).

Number of tasks: 128
Number of data dependencies: 0

Dotp

Embarrassingly parallel.
Basic linear algebra operations ($\Sigma x*y$).

Number of tasks: 100
Number of data dependencies: 0



Heat

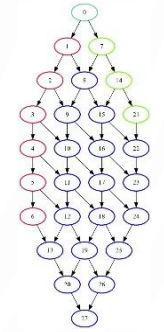
Wavefront parallelism based on stencil.
Heat diffusion based on the Gauss-Seidel.

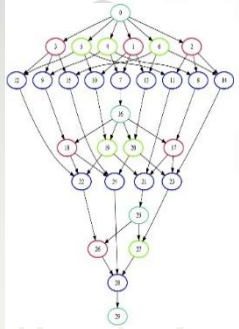
Number of tasks: 640
Number of data dependencies: 2128

HOG

Histogram of Gradients (HOG).
Feature extraction for object detection.

Number of tasks: 240
Number of data dependencies: 448





SparseLU

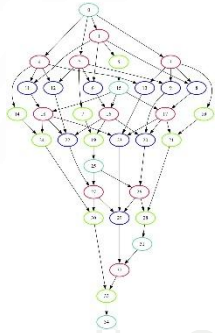
Irregular and dynamic parallelism.
SparseLU matrix decomposition.

Number of tasks: 1496
Number of data dependencies: 3960

Cholesky

Irregular and dynamic parallelism.
Cholesky factorization.

Number of tasks: 816
Number of data dependencies: 2040



Simulation: Representative Results

Slowdown vs MTET-X and TMCD-X using 8 threads

	Algorithm	Applications											
		Axpy		*Dotp*		*Heat*		*HOG*		*SparseLU*		*Cholesky*	
		Time (s)	Diff	Time (s)	Diff	Time (s)	Diff	Time (s)	Diff	Time (s)	Diff	Time (s)	Diff
*SotA**	BFS	0.374	13%	0.026	16%	3.391	10%	0.84	40%	8.12	11%	1.546	14%
	LPT	0.333	1%	0.023	6%	3.07	0%	0.72	20%	7.399	1%	1.416	4%
	SPT	0.358	9%	0.024	10%	3.069	0%	0.71	18%	7.497	2%	1.453	7%
	LNSNL	0.329	0%	0.023	4%	3.104	1%	0.82	37%	7.329	0%	1.414	4%
Proposed heuristics	MTET-MET	0.329	0%	0.022	0%	3.098	1%	0.62	3%	7.382	1%	1.382	2%
	MTET-MRT	0.329	0%	0.022	0%	3.159	3%	0.61	2%	7.591	4%	1.398	3%
	MTET-MCD	0.329	0%	0.022	0%	3.159	3%	0.61	2%	7.591	4%	1.398	3%
	TMCD-MET	0.358	9%	0.023	3%	3.098	1%	0.62	3%	7.645	4%	1.382	2%
	TMCD-MRT	0.358	9%	0.023	3%	3.159	3%	0.61	2%	7.948	8%	1.42	4%
	TMCD-MCD	0.358	9%	0.023	3%	3.159	3%	0.61	2%	7.948	8%	1.42	4%

— **MTET-X** is the more stable heuristic set among all, outperforming SotA algorithms.

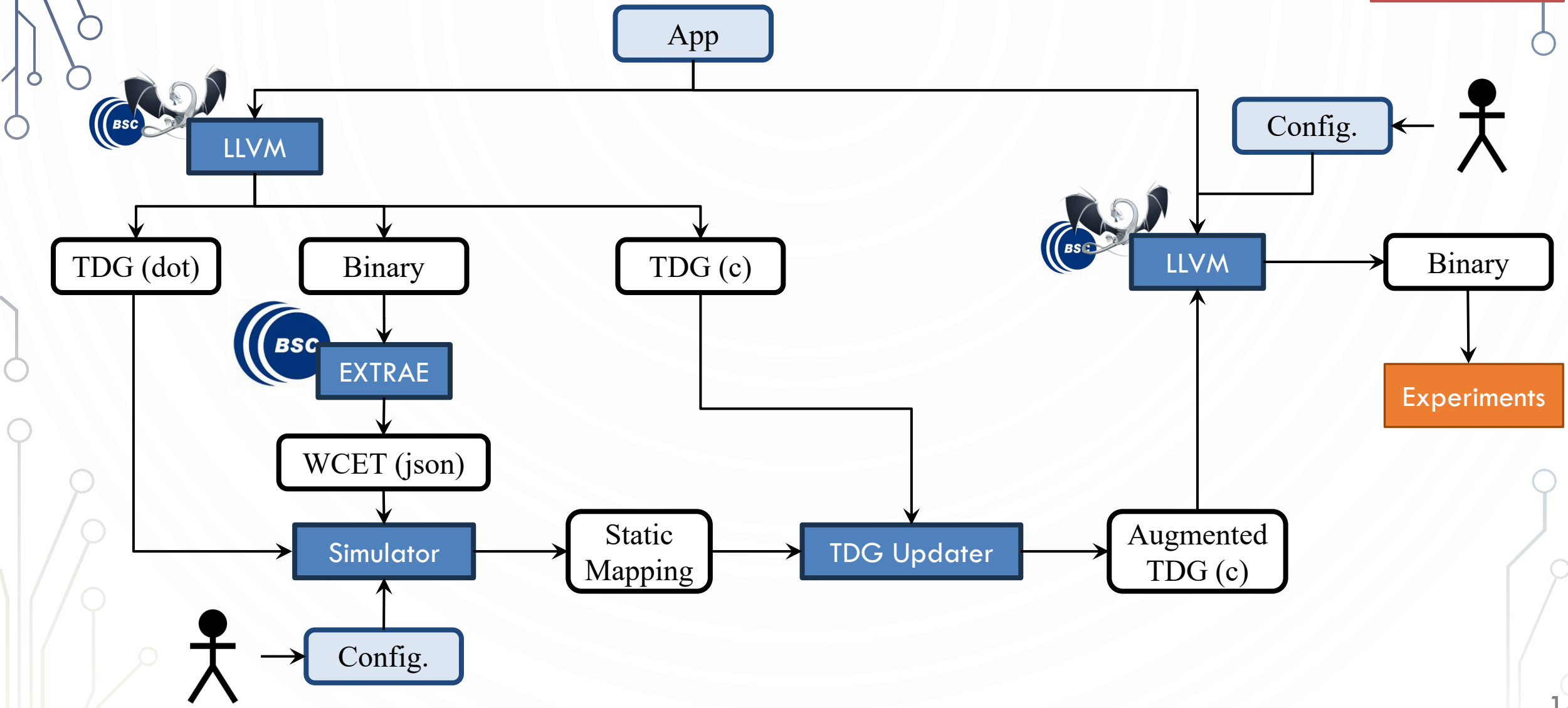
— **Higher number of threads induces more variability** (results are more stable with 4 threads), indicating optimal scheduler depends on application properties (MTET-X for embarrassingly parallel, and both MTET and TMCD for more unstructured parallelism).

*__BFS:__ Breadth-First Scheduler __LPT:__ Longest Processing Time __SPT:__ Shortest Processing Time __LNSNL:__ Largest Number of Successors in the Next Level

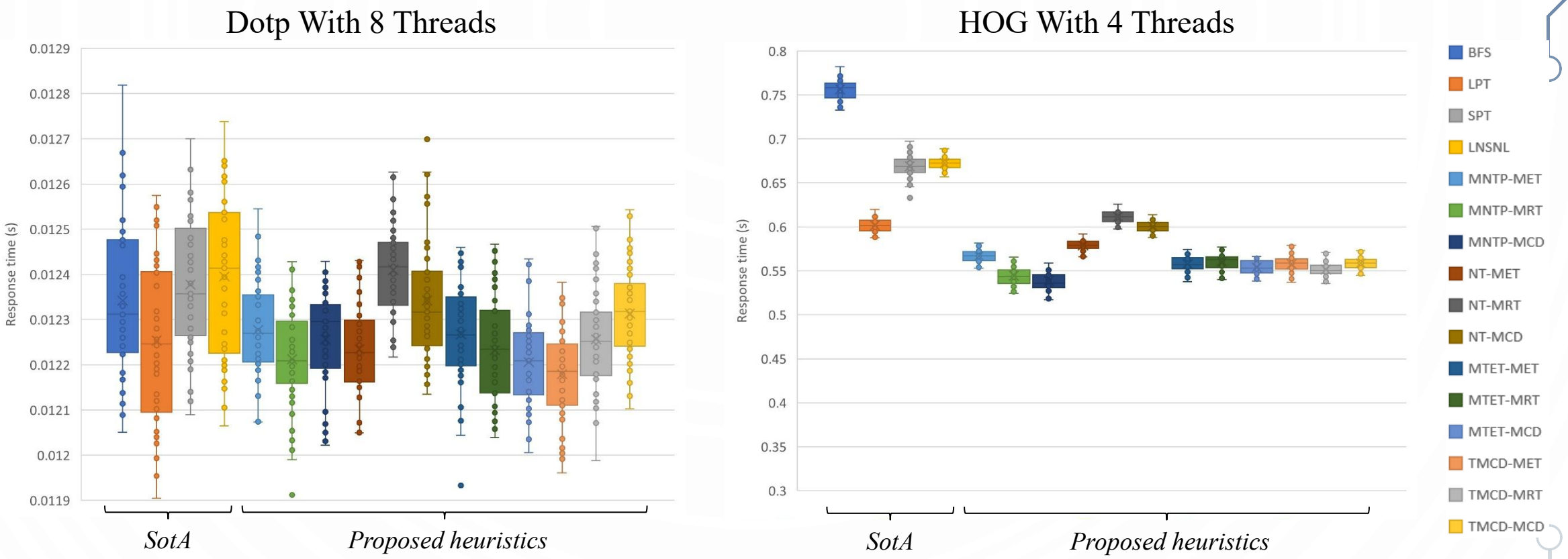
Runtime Implementation in LLVM and Empirical Validation

Simulation-Based Static Mapping

Objectives 2 and 3



Static Mapping: Representative Results



— **Heuristics' WCRT** and **response time variability** are lower than most SotA, particularly the **WCRT** obtained with **MNTP-X** and **MTET-X**.

— Selecting the **most efficient methods** from this set depends on the **application configuration** and the **hardware platform**.

Dynamic Mapping: Implementation

Objectives 2 and 3

Implemented Heuristics

MNTP and **MTET** from the **allocation** phase, and **MET** from the **dispatching** phase, because these heuristics have **lower complexity**, potentially leading to **lower overhead** .

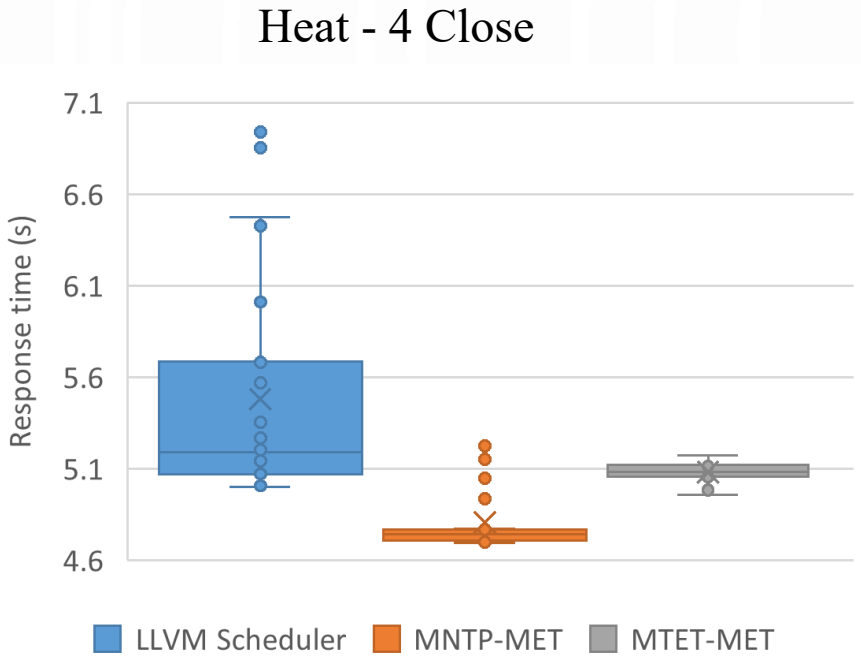
Changes in LLVM

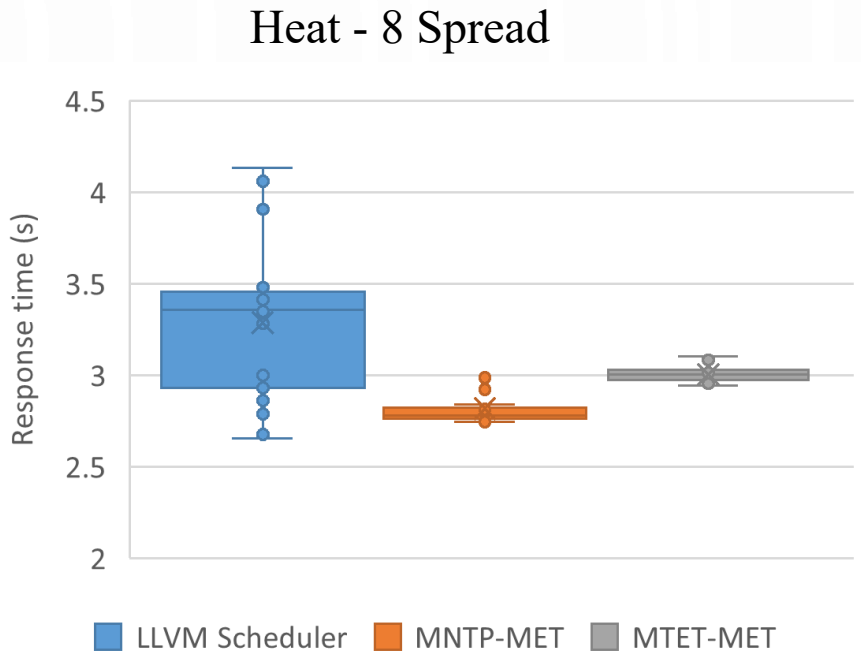
- **Disable task stealing**
- **Add** the '**tot_exe_time**' attribute to the ***kmp_base_thread_data*** structure in **kmp.h**
- **Implement MNTP** and **MTET** heuristics in the ***__kmp_push_task()*** function in **kmp_tasking.cpp**
- **Implement MET** heuristic in the ***__kmp_remove_my_task()*** function in **kmp_tasking.cpp**

M. Samadi, et al. *Time-predictable task-to-thread mapping in multi-core processors*, Journal of Systems Architecture, 2024.

Publicly available implementation: https://github.com/m-samadi/heuristic-mapping-LLVM

Dynamic Mapping: Representative Results



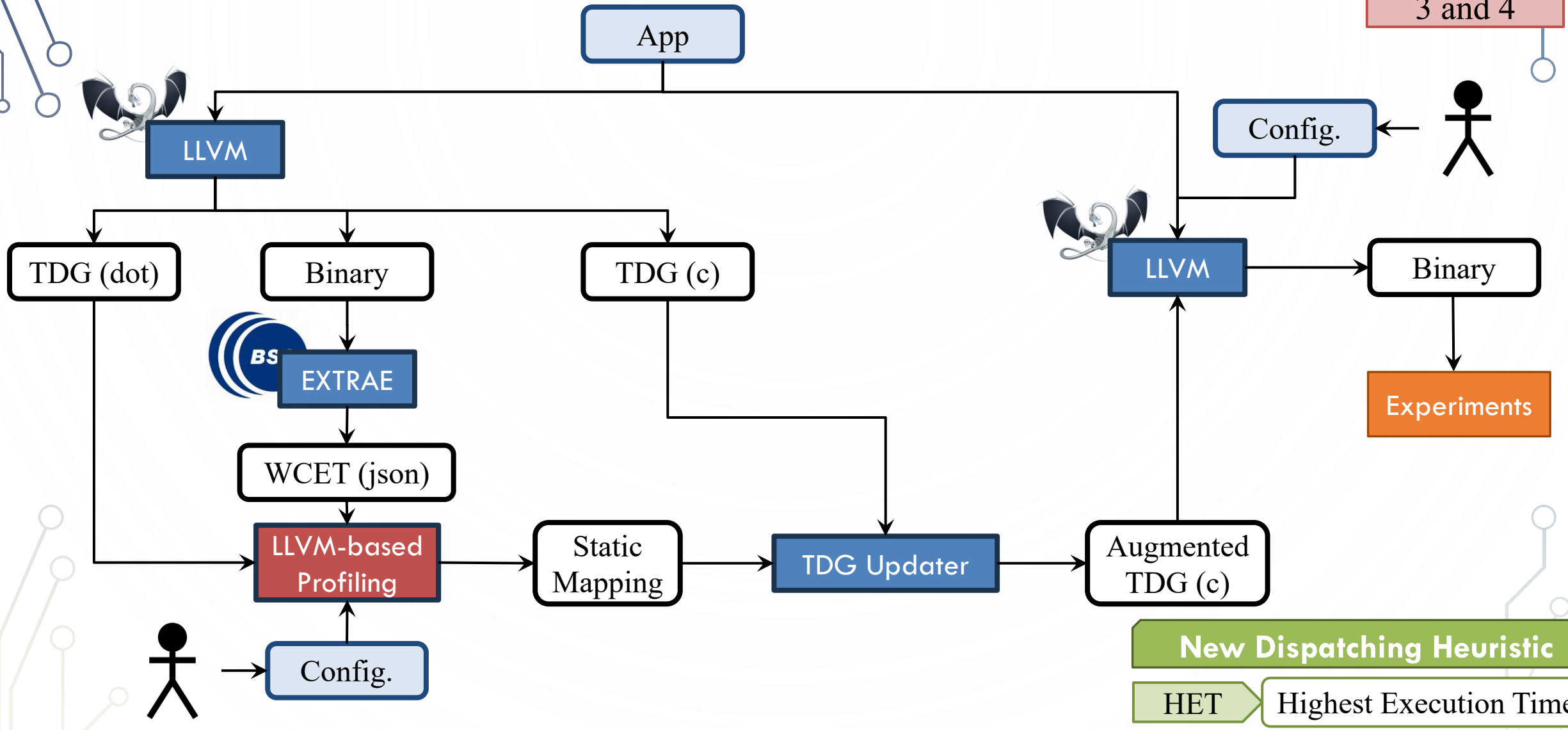


— **WCRT** and **response time variability of proposed algorithms** are **lower** than the LLVM's default, because **MNTP** and **MTET** reduce the **contention** on queues.

— **Improvements** are noticeable in **4 close** because **cache misses** are **not high** in close binding.

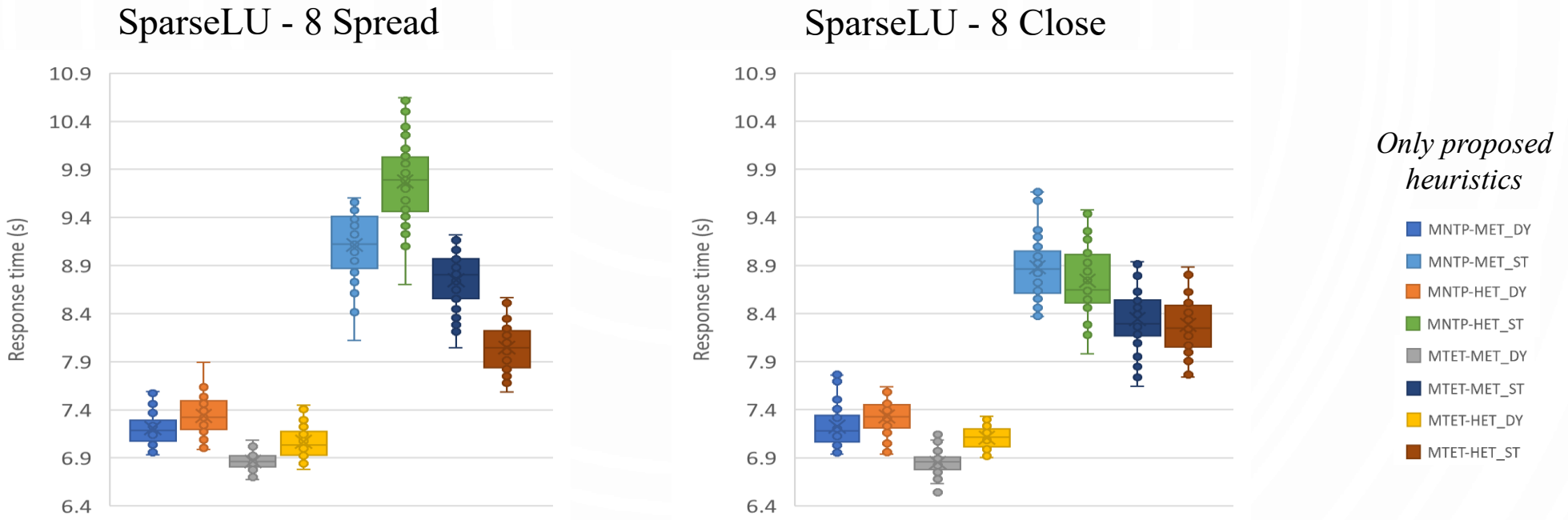
— **MTET** is **more suitable** when **number of threads is low**, while **MNTP** is **more suitable** when **number of threads is high**, as **MNTP** has **low overhead**.

Profiling-Based Static Mapping



M. Samadi, et al. *Evaluation of Heuristic Task-to-Thread Mapping Using Static and Dynamic Approaches*. JSSPP, 2024.

Static vs. Dynamic Mapping: Representative Results



— **WCRT** and **variability** in **dynamic** mapping are **better** than in **static**, as algorithms are **not work-conserving** in the **static** version.

— **Response time** and **variability** achieved with **MTET-X** are **lower** than **MNTP-X**, because **MTET improves load balancing** of queues based on **execution time**. In particular, **MTET-MET** shows **better performance** and **variability**, as **MET improves the work-conserving** nature of the mapping.

Schedulability Analysis

Methodology

Objective 1

The developed **simulator** is **integrated** with a state-of-the-art **analysis library** to perform schedulability analysis.

The **simulator** performs **static mapping**, while the **library** generates **random graphs** and performs their **schedulability analysis** based on the **static mapping**.

Synthetic benchmarks

Case #	Size of graph	Number of OpenMP tasks	Deadline policy
1	Small	1-50	Implicit
2	Large	51-100	Implicit
3	Small	1-50	Constrained
4	Large	51-100	Constrained

Deadline = Period
Simplified analysis
More rigid

Deadline ≤ Period
Complex analysis
More flexible

M. Samadi, et al. *Schedulability Analysis of OpenMP Applications Under Heuristic Task-to-Thread Mapping*. NG-RES, 2026.

Representative Results

		Case 1	Case 2		Case 3		Case 4	
		4 & 8 Threads	4 Threads	8 Threads	4 Threads	8 Threads	4 Threads	8 Threads
SotA	BFS	100%	93%	100%	35%	59%	4%	10%
	LPT	100%	91%	100%	34%	66%	2%	17%
	SPT	100%	88%	100%	38%	66%	2%	15%
	LNSNL	100%	91%	100%	37%	64%	5%	15%
Proposed heuristics	MNTP-MET	100%	99%	100%	27%	48%	3%	24%
	MNTP-MRT	100%	100%	100%	28%	47%	4%	25%
	MNTP-MCD	100%	100%	100%	28%	47%	4%	25%
	MTET-MET	100%	100%	100%	32%	48%	6%	25%
	MTET-MRT	100%	100%	100%	33%	48%	5%	25%
	MTET-MCD	100%	100%	100%	33%	48%	5%	25%
	TMCD-MET	100%	99%	100%	32%	48%	6%	25%
	TMCD-MRT	100%	100%	100%	34%	48%	2%	24%
	TMCD-MCD	100%	100%	100%	34%	48%	2%	24%

Note: % denotes percentage of the graphs completed within their deadline by each mapping method.

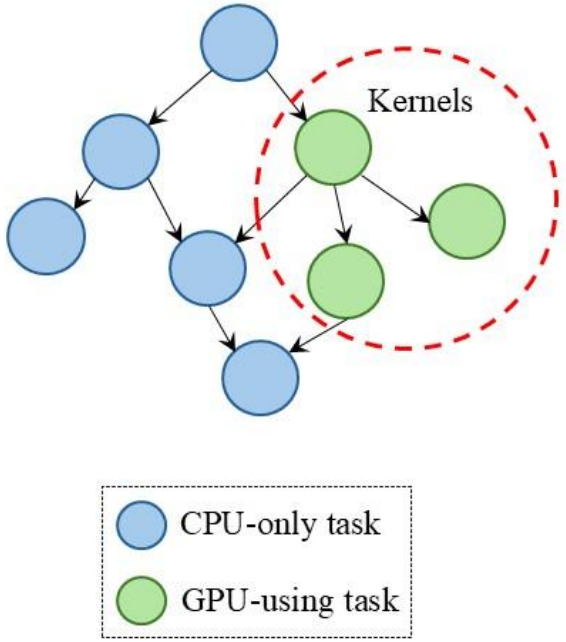
Evaluation Summary

- **Implicit deadlines** (cases 1 and 2):
  - Yield **better schedulability** than **constrained** deadlines.
  - **No significant performance difference** for **small graphs and large graphs with many threads**.
- The **heuristics** show **improved efficiency** in **larger DAGs** (cases 2 and 4), as they have a **better opportunity** to select the most **appropriate tasks** from **many ready tasks**, **enhancing work-conserving mapping** and **schedulability**.
- The **original version of heuristics** fail to take advantage of the parallel platform in **small and constrained DAGs** (Case 3). This evaluation led to an improved implementation shown in the results of the thesis.
- Proposed **heuristics** yield **higher schedulability** compared to SotA in most cases, as **most graphs** are **schedulable** with the **heuristics**.

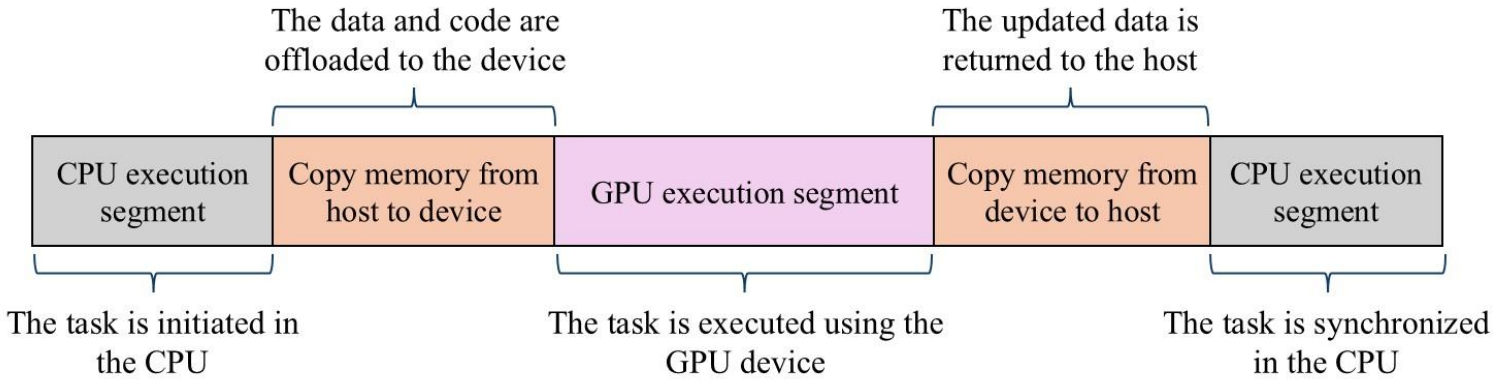
Proposed Task-to-Accelerator Mapping for Heterogeneous Platforms

Model

Target Application:
a heterogeneous program
represented as a computation graph

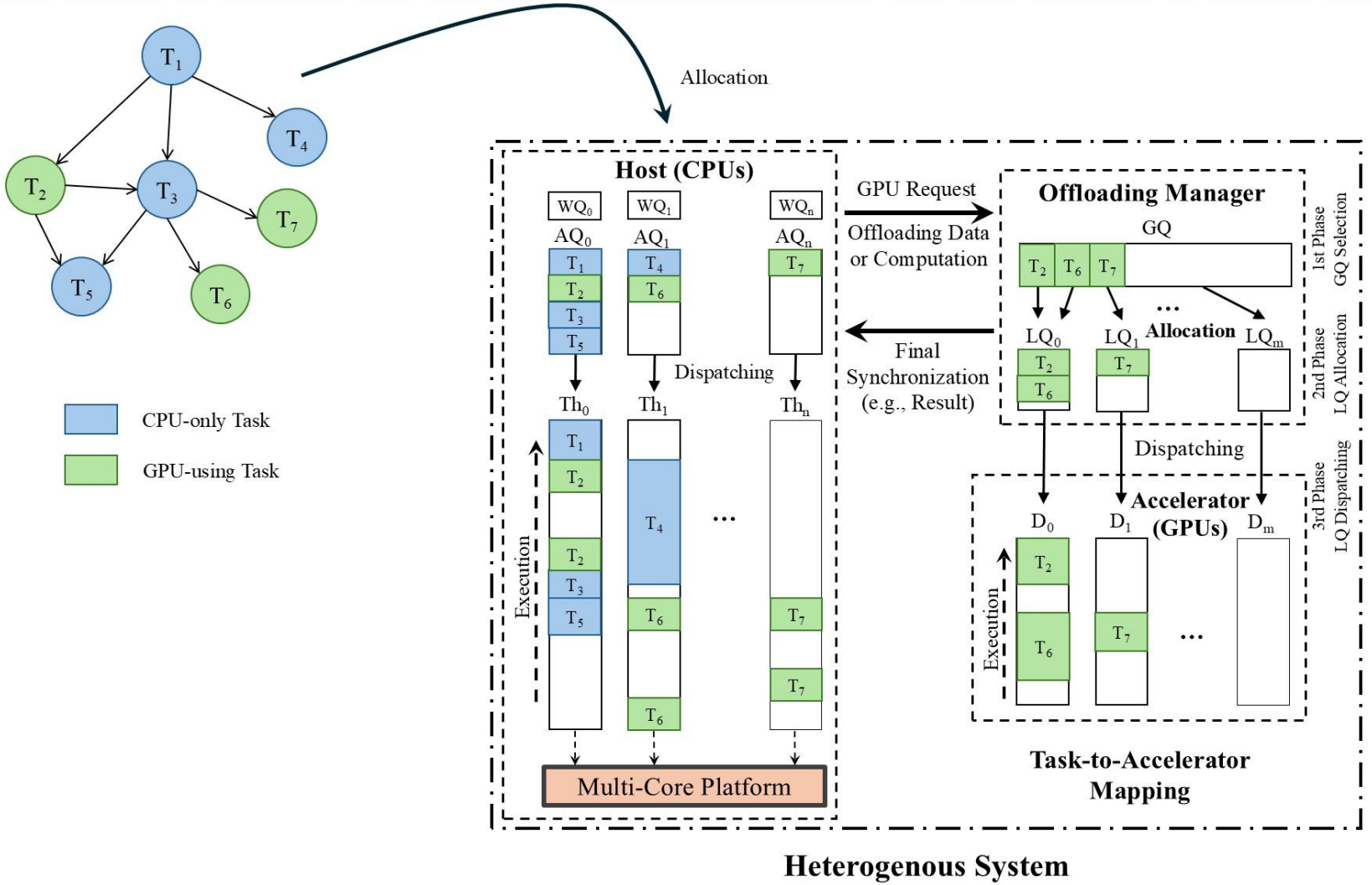


Main Parts of OpenMP Target Tasks
(those potentially using the GPU)



Methodology

Objective 5



M. Samadi, et al. *Task-to-Accelerator Mapping for Heterogeneous Systems Using Heuristics*. AEiC WiP, 2025.

M. Samadi, et al. *Task-to-Accelerator Mapping for Predictable OpenMP Applications*, Submitted to AEiC Journal Track, 2026.

Selection, Dispatching, and Allocation Algorithms

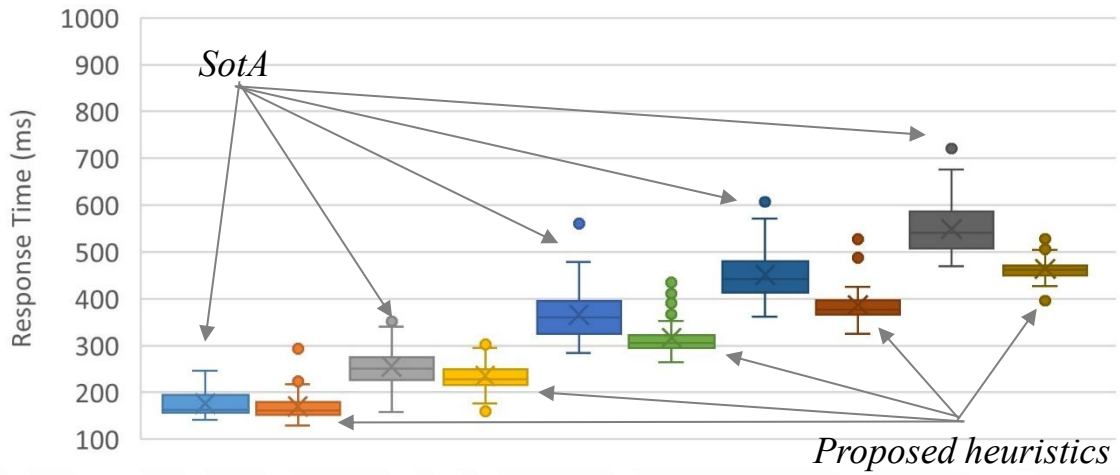
GQ Selection and LQ Dispatching

LET	Least Execution Time	**LET** creates **nested tasks** sooner.
MNAOT	Maximum Number of All Outgoing Tasks	**MNAOT** makes most **successors ready** for execution sooner.
SWSM	Weighted-Sum Model	**SWSM** combines **LET** and **MNAOT**.

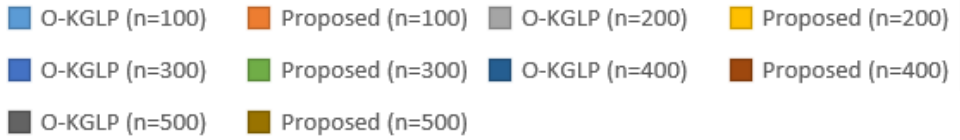
LQ Allocation

MNJ	Minimum Number of Jobs	**MNJ** improves the **load balancing** of queues based on the **number of tasks**.
LTET	Least Total Execution Time	**LTET** enhances the **load balancing** of queues based on **task execution time**.
AWSM	Weighted-Sum Model	**AWSM** combines **MNJ** and **LTET**.

Relevant Results



System setup
- Highly-parallel synthetic graphs
- Task execution time: [1, 10] *ms*
- Running with 32 CPUs and 8 GPUs



Proposed: LET for GQ selection
LTET for LQ allocation
MNAOT for LQ dispatching

n: Maximum number of tasks

— **New algorithms outperform** the **O-KGLP** SotA in most cases, **reducing** application's **WCRT** and response-time **variability**.

— Their **efficiency** is **high** for **>300 tasks**, allowing **optimal selection** of **queues** and **appropriate tasks** within those queues (better load balancing and work-conserving mapping) for **improved performance**.

Overall Conclusions and Thesis Impact

Conclusions

Proposed task-to-thread and **task-to-accelerator** algorithms demonstrate significant **improvements** in **schedulability**, **reducing WCRT** (and **average response time**) and **response time variability**.

Therefore, the **proposed methods increase** the **OpenMP** capabilities for **time-critical systems**, **guaranteeing compliance** with their essential **requirements**.

The **methods** proposed and the **evaluations** carried out in this thesis are based on the **OpenMP task model**. However, they can also be used in **other task-based parallel programming models** (e.g., Taskflow, SYCL, OmpSs-2, starPU).

Thesis Impact

The main publications of the PhD include **two journal (JSA)** (one pending) and **two workshop (JSSPP and NG-RES)** papers. Secondary publications include **six conference (ETFA, INDIN, and AEiC)** papers.

Publicly available implementation of the **task-to-thread mapping** on the widely spread LLVM compilation framework.

Contribution, including LLVM implementation, to **three EU projects** (ELASTIC, AMPERE and MATISSE) and **two Spanish projects** (RESPECT and HiPART).

Technological trends in **IoT** and **CPS** focus on **multi-core** and **heterogeneous systems**, making **proposed methods** valuable for **optimizing performance** in **critical areas** (e.g., automotive industry*).

* S. Royuela, et al. *Multi-criteria Analysis and Optimization in the AMPERE Ecosystem*. DeCPS, 2023.

Thank You for Your Attention!

Contact:
mohammad.samadi@upc.edu
mmasa@isep.ipp.pt

Publications

Main contributions:

- M. Samadi, S. Royuela, L. M. Pinho, T. Carvalho, and E. Quiñones. "*Time-predictable task-to-thread mapping in multi-core processors*". In: Journal of Systems Architecture 148 (2024), p. 103068.
- M. Samadi, T. Carvalho, L. M. Pinho, and S. Royuela. "*Evaluation of Heuristic Task-to-Thread Mapping Using Static and Dynamic Approaches*". In: Workshop on Job Scheduling Strategies for Parallel Processing. Springer. 2024, pp. 103–119. doi: 10.1007/978-3-031-74430-3_6.
- M. Samadi, T. Carvalho, L. Miguel Pinho, and S. Royuela. "*Schedulability Analysis of OpenMP Applications Under Heuristic Task-to-Thread Mapping*". In: 7th Edition of the Workshop on Next Generation Real-Time Embedded Systems (NG-RES), 2026.
- M. Samadi, T. Carvalho, L. Miguel Pinho, and S. Royuela. "*Task-to-Accelerator Mapping for Predictable OpenMP Applications*". Submitted to the journal track of AEiC, 2026.

Preliminary analysis:

- M. S. Gharajeh, T. Carvalho, and L. M. Pinho. "Configuration of Parallel Real-Time Applications on Multi-Core Processors". In: 2022 IEEE 20th International Conference on Industrial Informatics (INDIN). IEEE. 2022, pp. 67–73. doi: 10.1109/INDIN51773.2022.9976163.

Publications

Secondary contributions:

- M. S. Gharajeh, S. Royuela, L. M. Pinho, T. Carvalho, and E. Quiñones. "*Heuristic-based task-to-thread mapping in multi-core processors*". In: 2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA), work-in-progress session. IEEE. 2022, pp. 1–4. doi: 10.1109/ETFA52439.2022.9921453.
- M. Samadi, T. Carvalho, L. Miguel Pinho, and S. Royuela. "*Task-to-Thread Mapping in OpenMP Using Fuzzy Decision Making*". In: 28th Ada-Europe International Conference on Reliable Software Technologies (AEiC 2024), work-in-progress session. ACM SIGAda Ada Letters 44.1 (2024), pp. 107–111.
- M. Samadi, T. Carvalho, L. Miguel Pinho, and S. Royuela. "*Task-to-Accelerator Mapping for Heterogeneous Systems Using Heuristics*". In: 29th Ada-Europe International Conference on Reliable Software Technologies (AEiC 2025), work-in-progress session. ACM SIGAda Ada Letters (2025).

Other collaborations and contributions (AMPERE H2020 project):

- T. Carvalho, L. M. Pinho, M. Samadi, S. Royuela, A. Munera, and E. Quiñones. "*Framework for the Analysis and Configuration of Real-Time OpenMP Applications*". In: 2023 IEEE 21st International Conference on Industrial Informatics (INDIN). IEEE. 2023, pp. 1–8. doi: 10.1109/INDIN51400.2023.10218276.
- S. Royuela, A. Munera, E. Quiñones, T. Carvalho, L. Miguel Pinho, M. Samadi, T. Cucinotta, G. Ara, F. Paladino, S. Mazzola, et al. "*Multi-criteria Analysis and Optimization in the AMPERE Ecosystem*". In: Workshop on Challenges and New Approaches for Dependable and Cyber-Physical System Engineering (DeCPS 2023), ACM SIGAda Ada Letters 43.2 (2024), pp. 94–99.