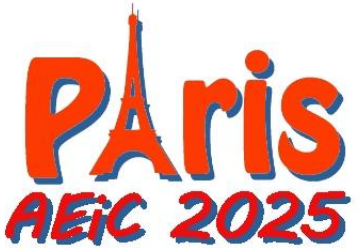
Task-to-Accelerator Mapping for Heterogeneous Systems Using Heuristics



Presenter:

Mohammad Samadi

Co-authors:

Tiago Carvalho, Luis Miguel Pinho, Sara Royuela

Polytechnic Institute of Porto, Portugal

Universitat Politècnica de Catalunya, Barcelona, Spain

Barcelona Supercomputing Center, Barcelona, Spain

June 2025

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Introduction

- GPUs are generally applied to accelerate diverse intensive computations.
- Performance of heterogeneous platforms can be improved using parallel programming models (e.g., OpenMP).
- An efficient task scheduling policy is necessary on these systems, especially for predictable OpenMP applications.
- Analyzing WCRT for heterogeneous systems is challenging because the number of processing elements (i.e., CPUs and GPUs) is high.
- Predictability and schedulability of time-critical systems can be achieved by reducing the variability of application response times and the WCRT.

Contributions/Objectives

- Propose an efficient task-to-accelerator mapping for multi-GPU systems considering OpenMP applications.
- Apply various heuristic algorithms with different characteristics during the allocation and dispatching phases to improve the load-balancing of queues and the work-conserving of mapping process.
- Use the suspension-aware process to free up the CPU while the kernel is running in the GPU.
- Improve the system predictability by reducing the response time variability of the parallel runtime.
- Improve the system schedulability by minimizing the WCRT.

System Model

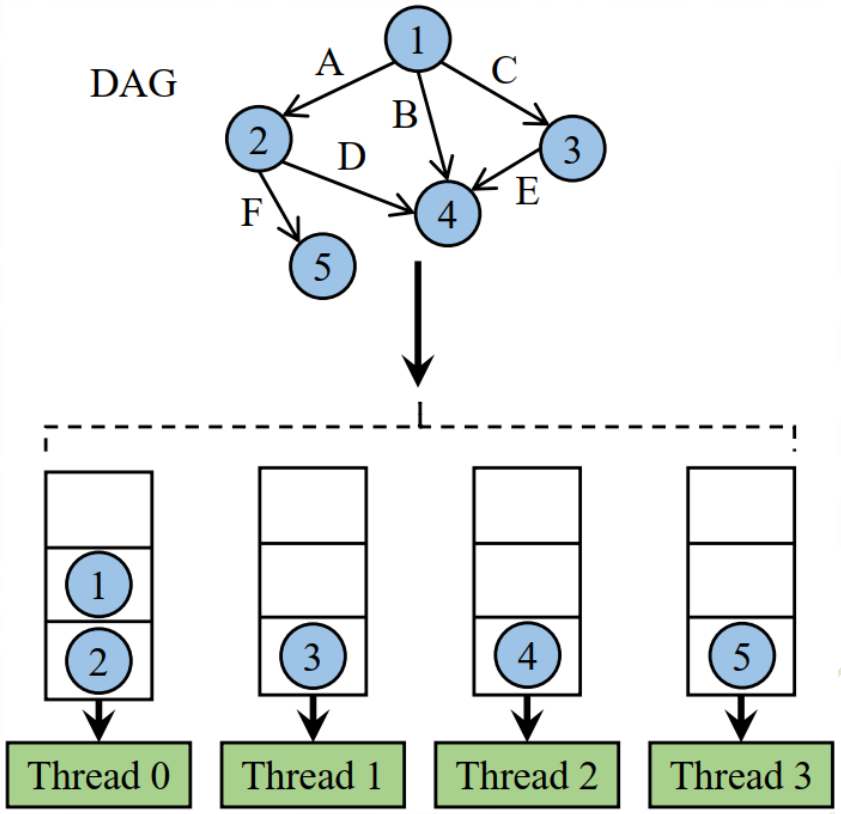
- Assume a multi-core platform with one or multiple GPU devices suitable for modern time-critical systems (e.g., CPSs).
- Suppose two types of tasks, including CPU-only tasks and GPU-using tasks.
- Execute CPU-only tasks using CPU cores and GPU-using tasks using both CPU and GPU (called host and device, respectively) resources.
- Dispatch GPU-using tasks to GPU(s) sequentially and non-preemptively.
- Manage tasks using a directed acyclic graph (DAG), where OpenMP tasks may involve data dependencies on each other.

System Model

OpenMP example

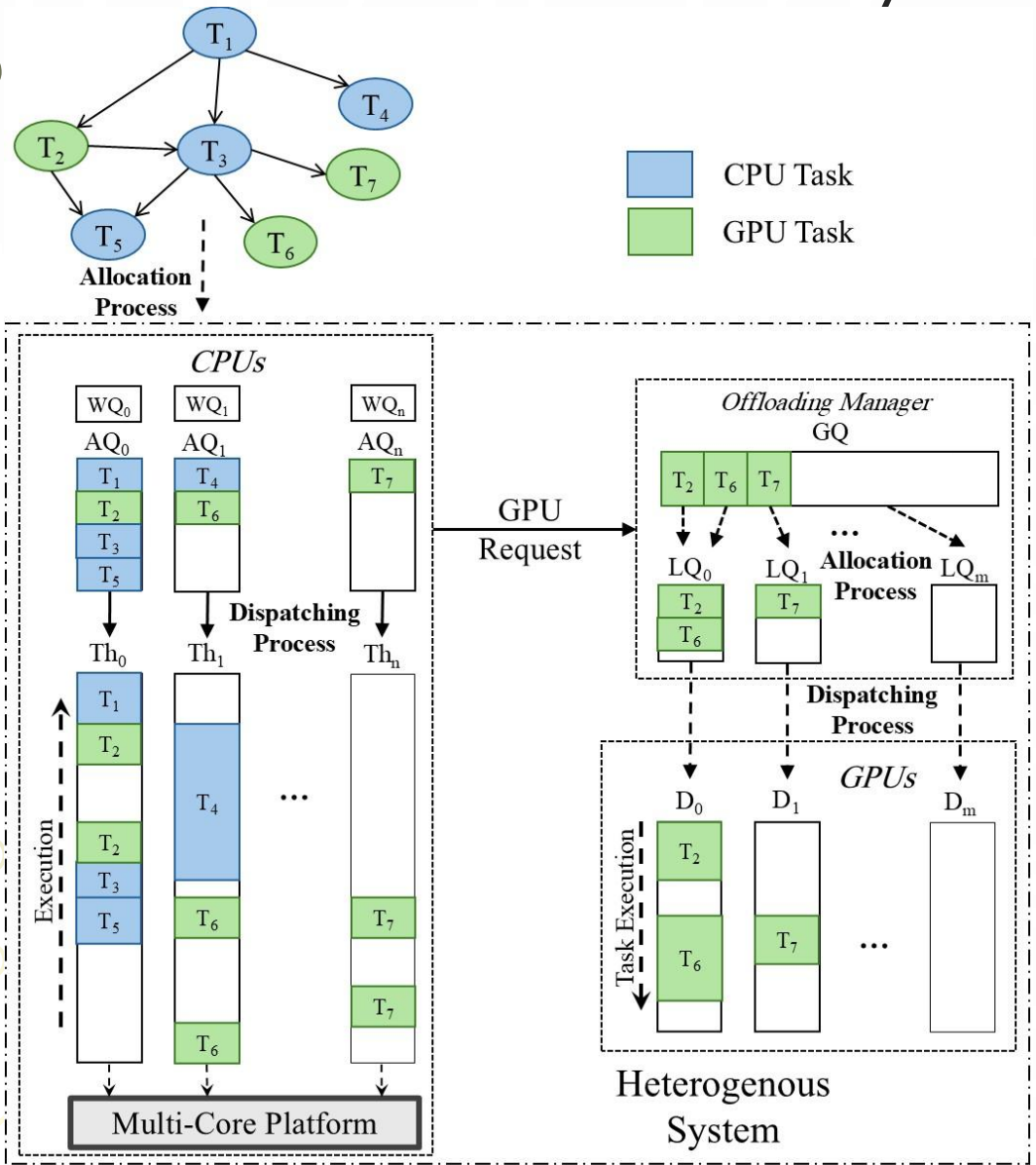
```
1  #pragma omp parallel
2  #pragme omp single
3  {
4    #pragma omp task depend(out: A,B,C)
5    Task1();
6
7    #pragma omp task depend(in: A) depend(out: D,F)
8    Task2();
9
10   #pragma omp task depend(in: C) depend(out: E)
11   Task3();
12
13   #pragma omp task depend(in: B,D,E)
14   Task4();
15
16   #pragma omp task depend(in: F)
17   Task5();
18 }
```

Multi-queue task system



Proposed Approach

System Overview



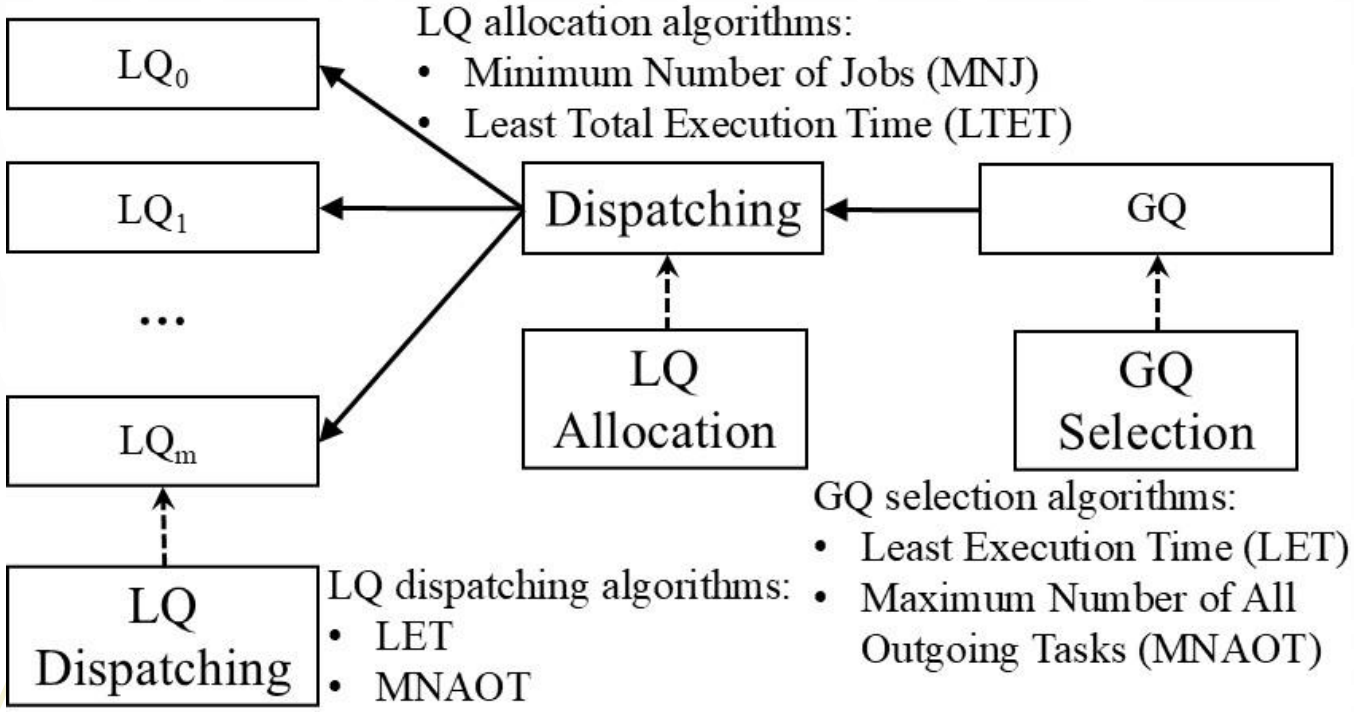
Kernel Execution

GQ selection:
Select a ready task from the global queue *GQ*.

LQ allocation:
Select a local queue *LQ* for the task.

LQ dispatching:
Select a task from *LQ* and dispatch it to the GPU.

Proposed Approach
Offloading Algorithms



Algorithms

LET and MNAOT can improve the work-conserving nature of the mapping process by executing thin tasks and tasks with more successors first, respectively, and increasing the number of ready tasks in the system.

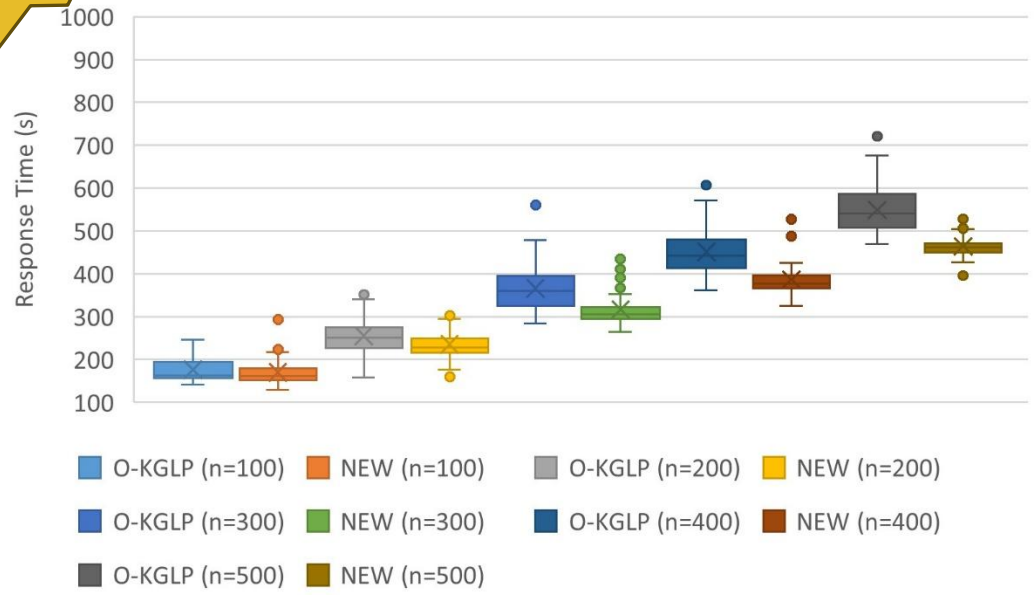
MNJ and LTET can improve the load-balancing of queues based on the number of tasks and execution time, respectively, to speed up the task execution process and reduce application response time and variability.

Number of combinations of the algorithms is $2^3 = 8$.

Simulation Results

High Parallelism

NEW: LET-LTET-MNAOT



Maximum task execution time: 10
Number of CPUs: 32
Number of GPUs is: 8
n: Maximum number of tasks

Evaluation

New method minimizes WCRT and reduce response time variability, providing more schedulability and predictability.

Its efficiency is remarkable when the maximum number of tasks is greater than or equal to 300, as there are enough options to select suitable queues for tasks and appropriate tasks from queues.

Algorithms that consider execution time to make their choices scale better with number of tasks, providing more reduced end-to-end response time.

Future Works

- The work depicted in this paper will be refined and extended by suggesting enhancements and new algorithms, such as multi-criteria decision analysis, for GPU scheduling.
- The performance of the work will be evaluated under multiple applications with different configurations on supercomputers, such as the MareNostrum 5 accelerated partition (ACC).

Thank you for attending!

For more information:
mmasa@isep.ipp.pt
mohammad.samadi@upc.edu