IPDPS 2024 • San Francisco, California USA

JSSPP

Evaluation of Heuristic Task-to-Thread Mapping Using Static and Dynamic Approaches

Presenter:

Mohammad Samadi

Co-authors:

Tiago Carvalho, Luis Miguel Pinho, Sara Royuela

Polytechnic Institute of Porto & INESC TEC, Porto, Portugal

Barcelona Supercomputing Center, Barcelona, Spain

Universitat Politècnica de Catalunya, Barcelona, Spain

May 2024

isep Instituto Superior de Engenharia do Porto

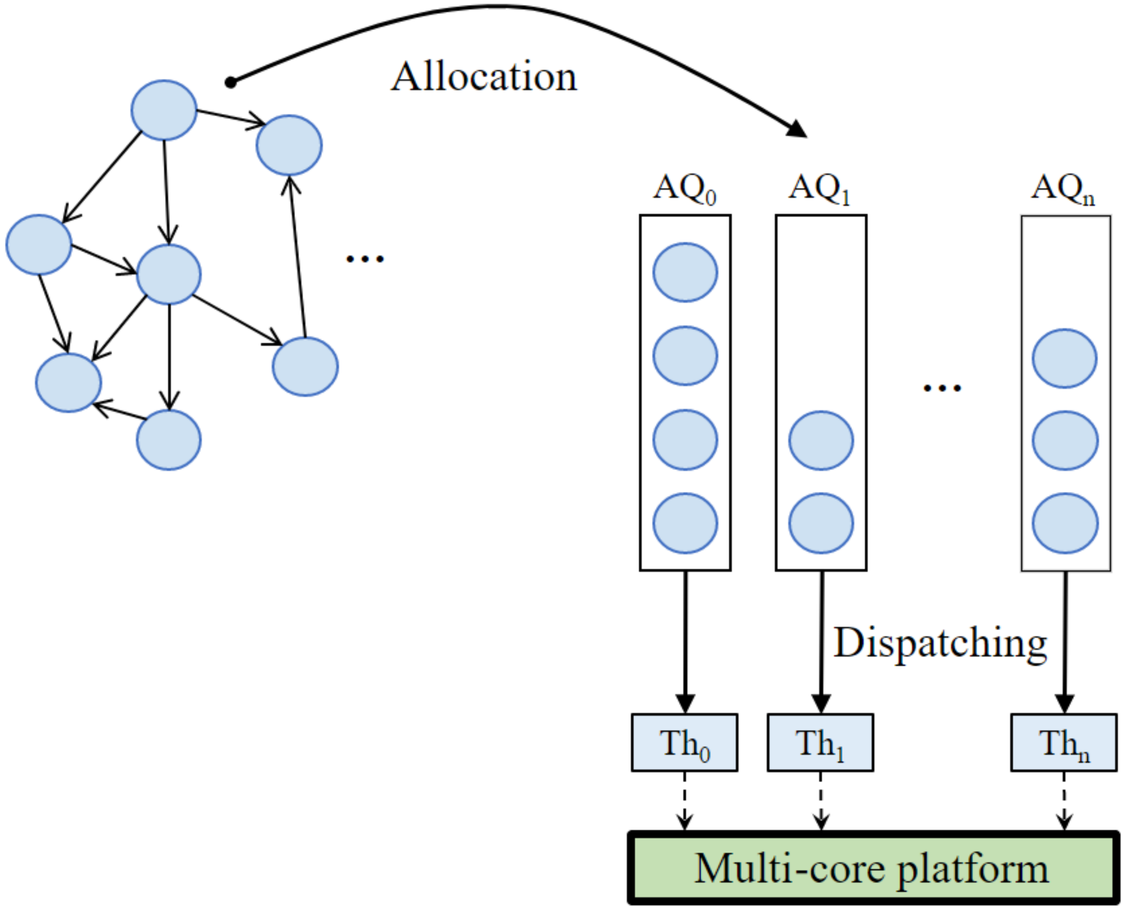
BSC Barcelona Supercomputing Center Centro Nacional de Supercomputación

UPC UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

Introduction

- Increase the need for high-performance computing (HPC) capabilities in complex cyber-physical systems
- Use OpenMP in real-time HPC applications running on shared-memory platforms
- Apply dynamic process in OpenMP runtime implementations
- Lack of time predictability in dynamic mapping algorithm
- Need time-predictable task-to-thread mapping in OpenMP

Heuristic Task-to-Thread Mapping



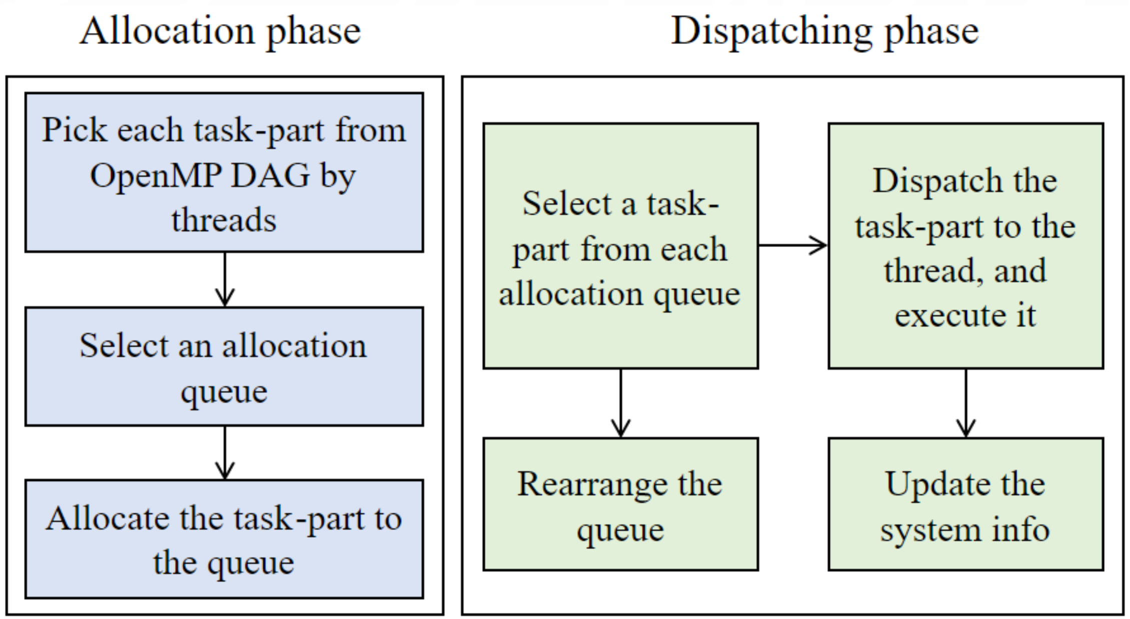
Allocation

AQ_0 AQ_1 … AQ_n

Dispatching

Th_0 Th_1 Th_n

Multi-core platform



Allocation phase

Pick each task-part from OpenMP DAG by threads

Select an allocation queue

Allocate the task-part to the queue

Dispatching phase

Select a task-part from each allocation queue

Dispatch the task-part to the thread, and execute it

Rearrange the queue

Update the system info

Heuristic Task-to-Thread Mapping

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

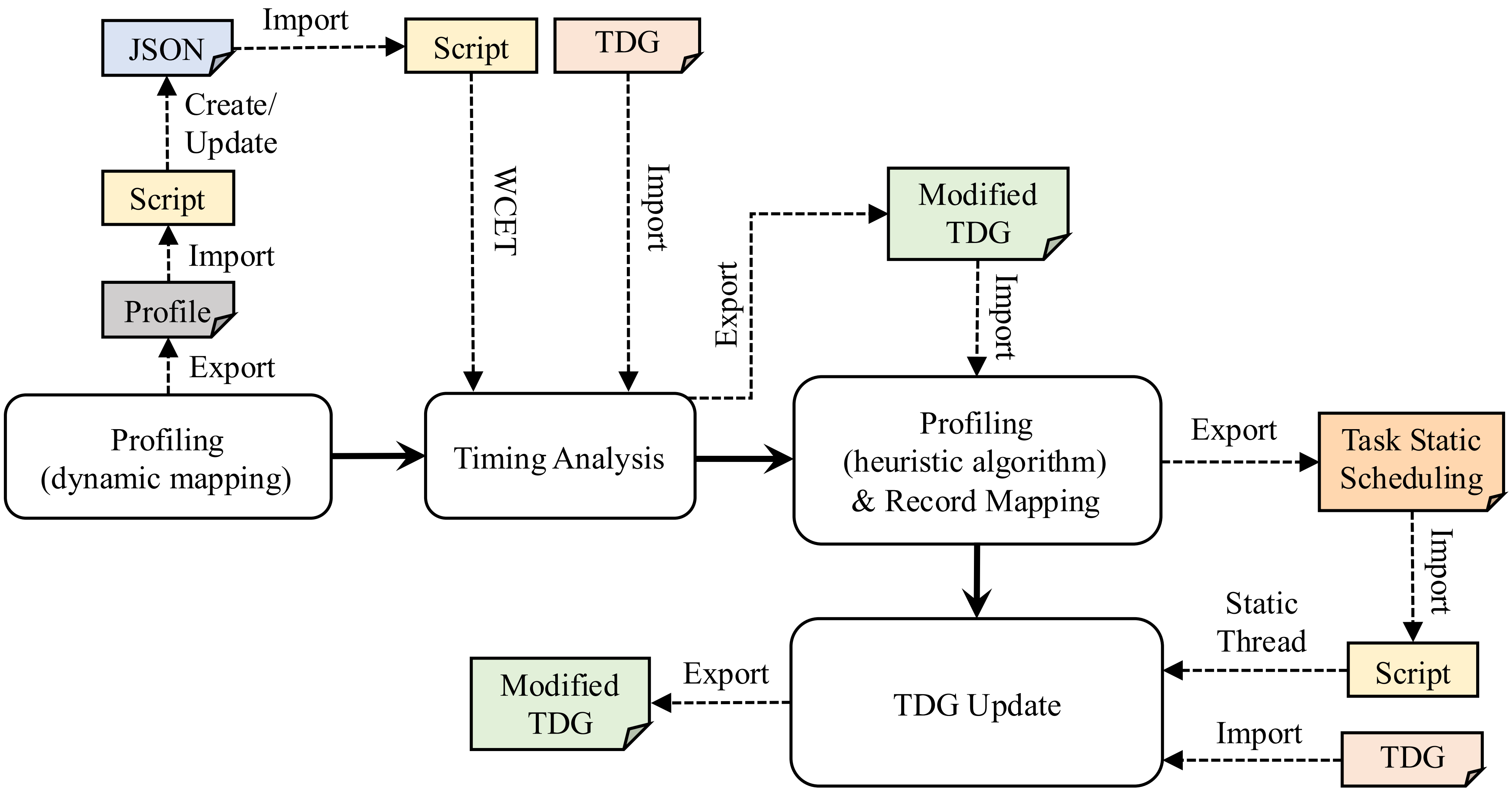
Problems With Heuristic Mapping

- Implementation of the MTET heuristic in LLVM is not efficient.
- MRT heuristic needs more information of the system.
- Current evaluations did not consider embarrassingly parallel applications.
- Experiments were carried out based on the WCET, without considering the variability of results.

Contributions

- Compare execution of the heuristics considering decisions made during execution, against static enforcement of the mapping.
- Determine static mapping by recording the execution in real platform instead by simulation.
- Reduce overhead by changing the implementation of one of the heuristics.
- Implement a new heuristic in the dispatching phase.

Methodology for Static Mapping



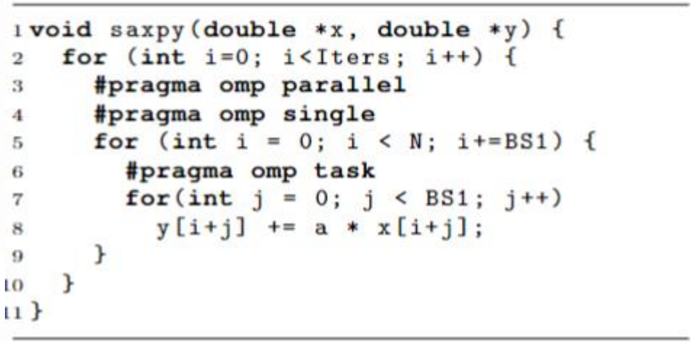
LLVM Extensions

- Improve the MTET heuristic by materializing the time expend in execution for each thread to reduce mapping overhead.
- Propose the highest execution time (HET) heuristic algorithm to reduce the workload of each queue and increase its chance of getting new tasks.

Applications

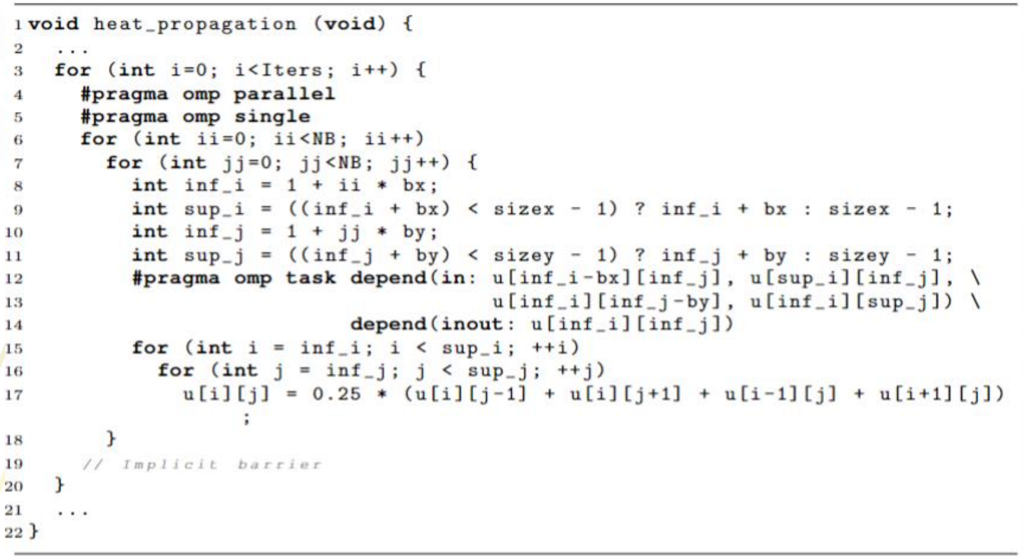
- Axpy: A Level 1 operation in the Basic Linear Algebra Subprograms (BLAS) package that combines scalar multiplication and vector addition.
- Heat: A simulator of heat diffusion based on the Gauss-Seidel method implementing a stencil computation.
- SparseLU: A matrix decomposition implementing a diamond-like form of irregular parallelism.

Applications



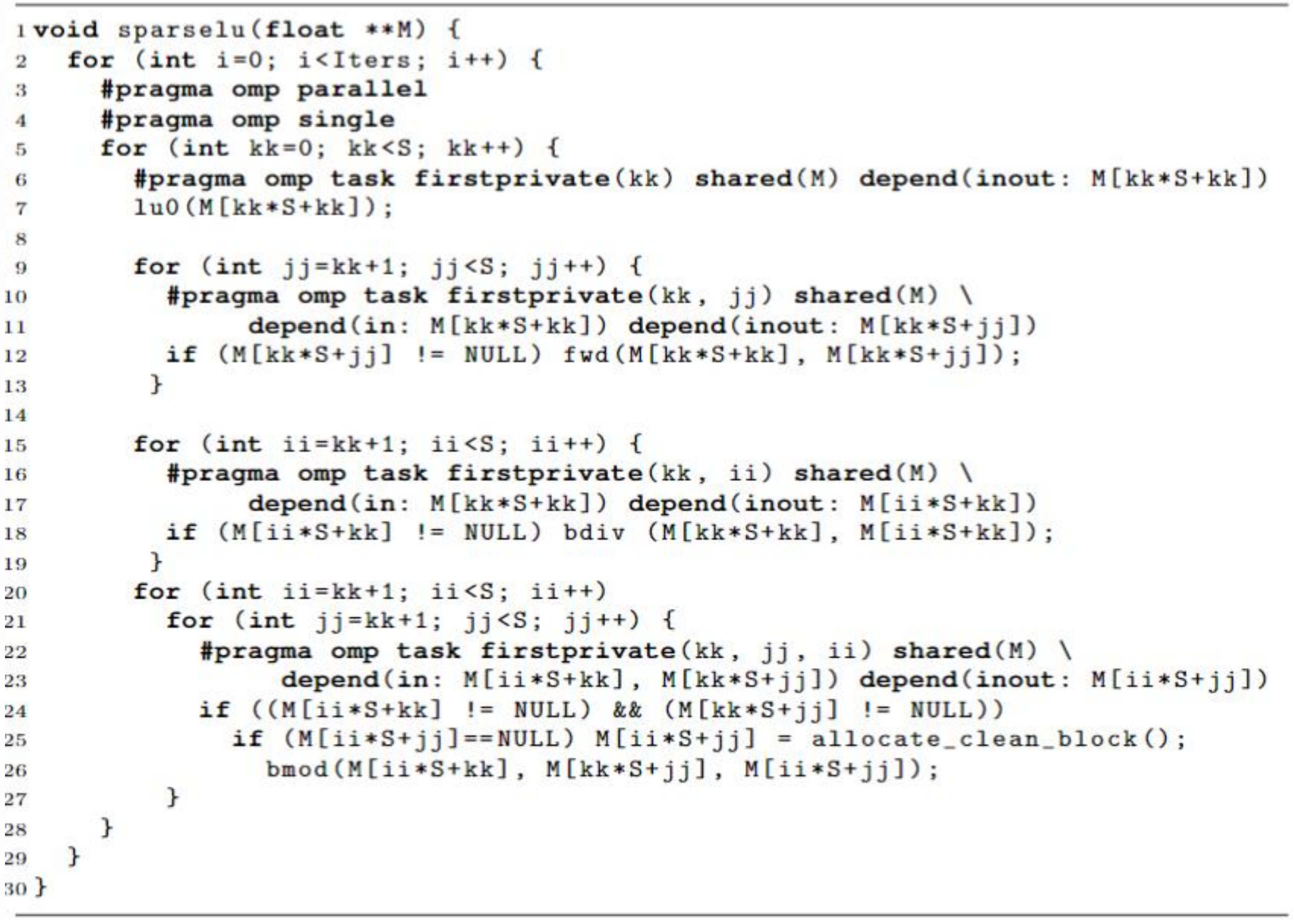
```
void saxpy(double *x, double *y) {
  for (int i=0; i<Iters; i++) {
    #pragma omp parallel
    #pragma omp single
    for (int i = 0; i < N; i+=BS1) {
      #pragma omp task
      for(int j = 0; j < BS1; j++)
        y[i+j] += a * x[i+j];
    }
  }
}
```

Axpy



```
void heat_propagation (void) {
  ...
  for (int i=0; i<Iters; i++) {
    #pragma omp parallel
    #pragma omp single
    for (int ii=0; ii<NB; ii++)
      for (int jj=0; jj<NB; jj++) {
        int inf_i = 1 + ii * bx;
        int sup_i = ((inf_i + bx) < sizex - 1) ? inf_i + bx : sizex - 1;
        int inf_j = 1 + jj * by;
        int sup_j = ((inf_j + by) < sizey - 1) ? inf_j + by : sizey - 1;
        #pragma omp task depend(in: u[inf_i-bx][inf_j], u[sup_i][inf_j], \
                                    u[inf_i][inf_j-by], u[inf_i][sup_j]) \
                        depend(inout: u[inf_i][inf_j])
        for (int i = inf_i; i < sup_i; ++i)
          for (int j = inf_j; j < sup_j; ++j)
            u[i][j] = 0.25 * (u[i][j-1] + u[i][j+1] + u[i-1][j] + u[i+1][j])
                ;
      }
    // Implicit barrier
  }
  ...
}
```

Heat



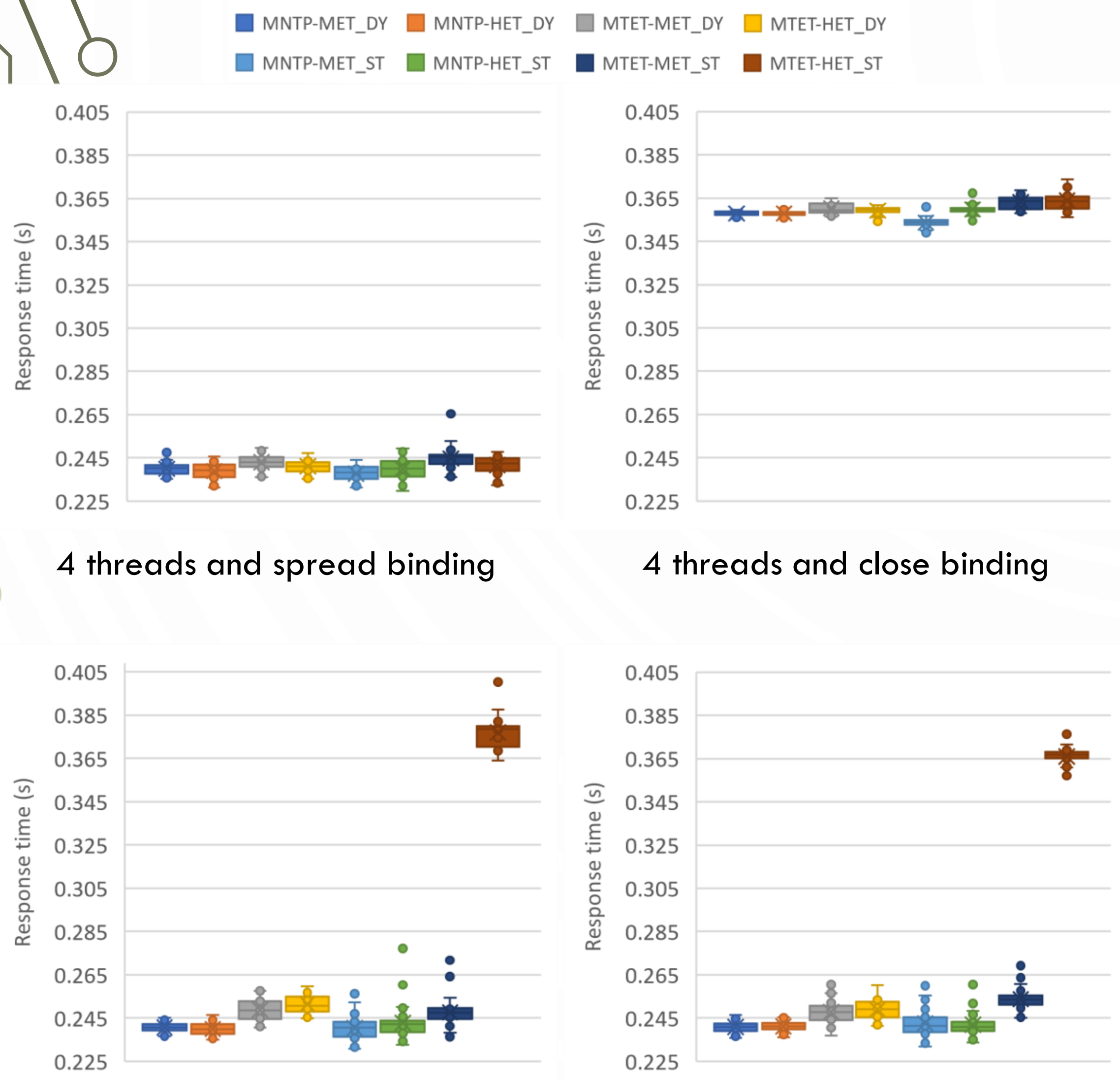
```
void sparselu(float **M) {
  for (int i=0; i<Iters; i++) {
    #pragma omp parallel
    #pragma omp single
    for (int kk=0; kk<S; kk++) {
      #pragma omp task firstprivate(kk) shared(M) depend(inout: M[kk*S+kk])
      lu0(M[kk*S+kk]);

      for (int jj=kk+1; jj<S; jj++) {
        #pragma omp task firstprivate(kk, jj) shared(M) \
             depend(in: M[kk*S+kk]) depend(inout: M[kk*S+jj])
        if (M[kk*S+jj] != NULL) fwd(M[kk*S+kk], M[kk*S+jj]);
      }

      for (int ii=kk+1; ii<S; ii++) {
        #pragma omp task firstprivate(kk, ii) shared(M) \
             depend(in: M[kk*S+kk]) depend(inout: M[ii*S+kk])
        if (M[ii*S+kk] != NULL) bdiv (M[kk*S+kk], M[ii*S+kk]);
      }
      for (int ii=kk+1; ii<S; ii++)
        for (int jj=kk+1; jj<S; jj++) {
          #pragma omp task firstprivate(kk, jj, ii) shared(M) \
               depend(in: M[ii*S+kk], M[kk*S+jj]) depend(inout: M[ii*S+jj])
          if ((M[ii*S+kk] != NULL) && (M[kk*S+jj] != NULL))
            if (M[ii*S+jj]==NULL) M[ii*S+jj] = allocate_clean_block();
             bmod(M[ii*S+kk], M[kk*S+jj], M[ii*S+jj]);
        }
    }
  }
}
```

SparseLU

Experimental Setup

- Hardware: NVIDIA Jetson AGX Xavier with 8-core CPU
- Operating system: Linux OS with the highest real-time priority
- Compiler and runtime system: An extended version of LLVM/Clang 17.0.0, supporting OpenMP 5.0 and Taskgraph
- Configurations: Different number of threads and binding options
- Execution process: Running the applications in 50 iterations and removing the first 10% of results as a warm-up phase

Experiments (Axpy)



4 threads and spread binding

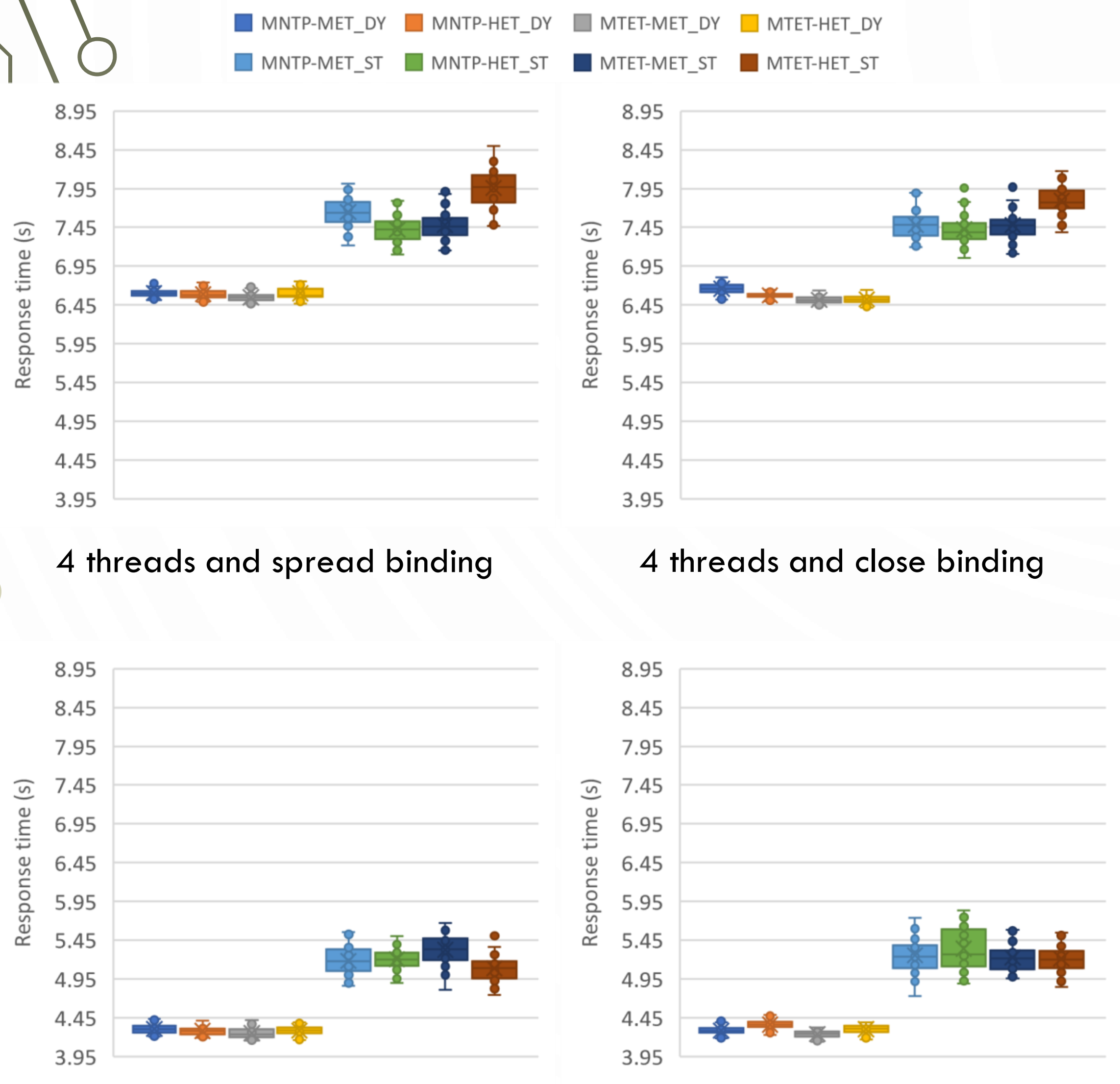
4 threads and close binding

8 threads and spread binding

8 threads and close binding

- Variability of MNTP-X in dynamic mapping is smaller than that of the other heuristic pairs.
- Response time with MNTP-X is slightly lower than others.
- Response time with MTET-HET and 8 threads in static mapping is higher than others.
- Response time with 4 threads and spread binding has not been decreased by increasing the number of threads to 8.
- There is not a considerable difference between the results obtained in dynamic and static mapping.

Experiments (Heat)



4 threads and spread binding

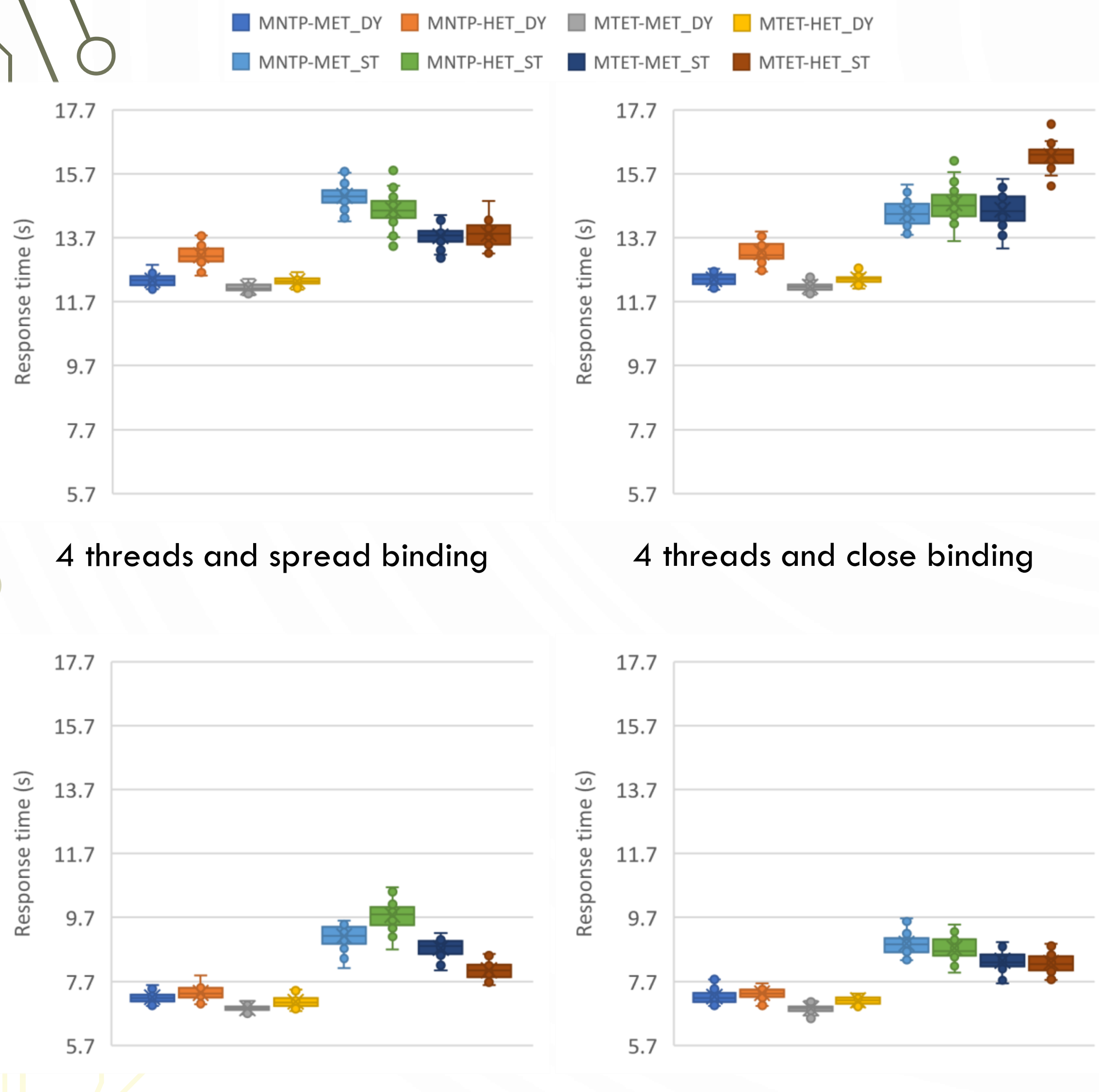
4 threads and close binding

8 threads and spread binding

8 threads and close binding

- Response time and variability of results achieved in dynamic mapping are lower than that in static mapping.
- Both the static and the dynamic versions show the same scalability when changing the number of threads.
- MTET-MET consistently shows slightly better performance and variability than the other pairs.

Experiments (SparseLU)



4 threads and spread binding

4 threads and close binding

8 threads and spread binding

8 threads and close binding

- Response time and variability given in dynamic mapping are better than those given in static mapping.
- Response time and variability of the results achieved with MTET-X are lower than those achieved with MNTP-X in most of the results.
- In this application with a complex TDG, there is a noticeable difference among the results given with the heuristics.

Discussion

- In the application without data dependencies, the performance of the heuristics in dynamic and static mapping is similar to each other.
- In the applications with data dependencies, their performance in dynamic mapping is better than that in static mapping.
- Variability in the dynamic mapping approach is lower than static mapping, important for real-time systems.

Future Works

The evaluation of the mappings under a wider set of conditions, supporting:

- Various artificial intelligence and data analytics applications
- Different hardware platforms with higher parallel capabilities

THANKS FOR YOUR ATTENTION!

For more information:
mmasa@isep.ipp.pt

isep Instituto Superior de Engenharia do Porto

UPC UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

BSC Barcelona Supercomputing Center Centro Nacional de Supercomputación