Scheduling Parallel Programs on Hybrid Machines

Mommessin Clément
Univ. Grenoble Alpes

Research project performed at INRIA-Grenoble

Under the supervision of:
Prof. D. Trystram, Grenoble INP
Dr. G. Lucarelli, Grenoble INP

June, 24th, 2016

High Performance Computing

- Evolution of parallel platforms
  - Increasing number of nodes
  - Heterogeneity within the nodes (CPU, accelerator (GPU), I/O, analytics, ...)

⇒ Hard to efficiently manage this increasing number of resource types.

High Performance Computing

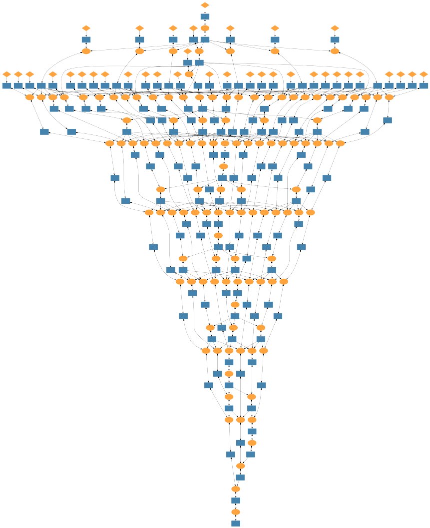
- Evolution of parallel platforms
  - Increasing number of nodes
  - Heterogeneity within the nodes (CPU, accelerator (GPU), I/O, analytics, ...)

$\Rightarrow$ Hard to efficiently manage this increasing number of resource types.

Ad-hoc algorithms vs. Generic algorithms

Problem Definition

- m identical CPUs
- k identical GPUs
- n dependent tasks T_j
- $\overline{p_j}$: processing time on CPU
- $\underline{p_j}$: processing time on GPU
- DAG $G = (V, E)$:
  precedence constraints



Problem Definition (cont.)

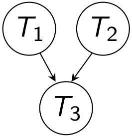
Goal

Minimize the *makespan*, completion time of the last task, for scheduling a set of dependent tasks to be executed on several identical CPUs and GPUs.
Specifically, scheduling a task is answering two questions:

- **Where?** – On which resource and which processor the task is executed
- **When?** – The date of execution of the task

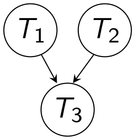
Example with List Scheduling

Task	Processing time on CPU / GPU
T_1	2 / 1
T_2	10 / 1
T_3	1 / 1



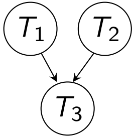
Example with List Scheduling

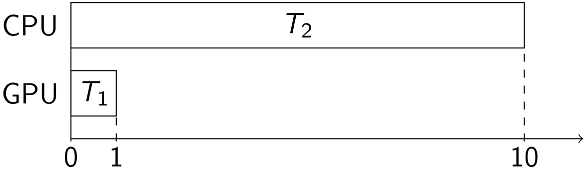
Task	Processing time on CPU / GPU
T_1	2 / 1
T_2	10 / 1
T_3	1 / 1



Example with List Scheduling

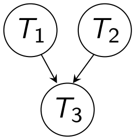
Task	Processing time on CPU / GPU
T_1	2 / 1
T_2	10 / 1
T_3	1 / 1

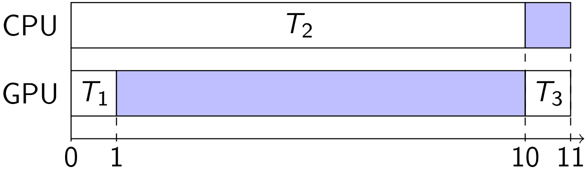




Example with List Scheduling

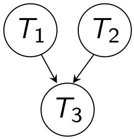
Task	Processing time on CPU / GPU
T_1	2 / 1
T_2	10 / 1
T_3	1 / 1

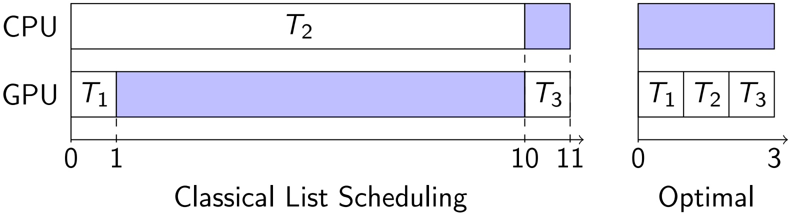




Example with List Scheduling

Task	Processing time on CPU / GPU
T_1	2 / 1
T_2	10 / 1
T_3	1 / 1





State of art

- Heuristics
  - Offline with dependent tasks and communications [Topcuoglu *et al.*, 1999]
- Approximation algorithms
  - Offline with independent tasks [Bleuse *et al.*, 2014]
  - Online with independent tasks [Chen *et al.*, 2014]
  - Offline with dependent tasks [Kedad-Sidhoum *et al.*, 2015]

Plan

- Heterogeneous Earliest Finish Time (HEFT)
- Heterogeneous Linear Program (HLP)
- New Algorithm (refined HLP)
- Experiments
- Conclusions and perspectives

HEFT [Topcuoglu *et al.*, 1999]

- Works in two steps:
  1. Task prioritization
  2. Task scheduling

- But no constant performance guarantee on the makespan
  - Counter-example with approximation ratio close to $\frac{m}{2}$ [Bleuse *et al.*, 2015]
  - **Improved** counter-example with approximation ratio close to $(1 - \frac{1}{e})m$ [This work]

HEFT [Topcuoglu *et al.*, 1999] (cont.)

Task prioritization: For the model of hybrid machines, the rank of each task T_j is recursively computed as follows:

$$rank(T_j) = \frac{m\overline{p_j} + k\underline{p_j}}{m+k} + \max_{i \in \Gamma^+(j)} \{rank(T_i)\}$$

Task scheduling: Schedules the task with the highest rank on the processor which minimizes the completion time of that task.

HLP [Kedad-Sidhoum *et al.*, 2015]

- Works in two steps:
  1. Assignment step: A linear program and a rounding method are used to assign each task to a resource type
  2. Scheduling step: A variant of List Scheduling schedules each task according to the assignment of the first step
- The approximation ratio is 6 [Kedad-Sidhoum *et al.*, 2015]
- The bound on the approximation ratio is **tight** [This work]

Assignment step

Variables used:

x_j : Binary assignment variable of T_j defined as:

$$x_j = \begin{cases} 1 & \text{if } T_j \text{ is processed on a CPU} \\ 0 & \text{otherwise} \end{cases}$$

C_j : Expected completion time of T_j

λ : Lower bound of the makespan

Assignment step (cont.)

$(ILP1) =$ minimize λ subject to:

$$C_i + \overline{p_j} x_j + \underline{p_j}(1 - x_j) \leq C_j \qquad \forall j \in V,\ \forall i \in \Gamma^-(j)$$

$$C_j \leq \lambda \qquad \forall j \in V$$

$$\sum_{j=1}^{n} \overline{p_j} x_j \leq m\lambda$$

$$\sum_{j=1}^{n} \underline{p_j}(1 - x_j) \leq k\lambda$$

$$x_j \in \{0, 1\} \qquad \forall j \in V$$

Assignment step (cont.)

$(LP1) =$ minimize λ subject to:

$$C_i + \overline{p_j} x_j + \underline{p_j}(1 - x_j) \leq C_j \qquad \forall j \in V,\ \forall i \in \Gamma^-(j)$$

$$C_j \leq \lambda \qquad \forall j \in V$$

$$\sum_{j=1}^{n} \overline{p_j} x_j \leq m\lambda$$

$$\sum_{j=1}^{n} \underline{p_j}(1 - x_j) \leq k\lambda$$

$$x_j \in [0, 1] \qquad \forall j \in V$$

Rounding Policy

Rounding of each variable x_j:

$$x_j = \begin{cases} 1 & \text{if } x_j^R \geq \frac{1}{2} \\ 0 & \text{otherwise} \end{cases}$$

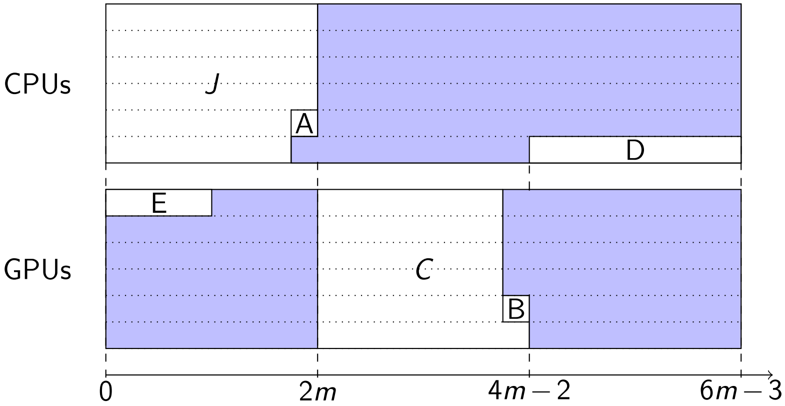
The goal of the rounding is to evenly balance the load between the CPUs and the GPUs.

Scheduling step

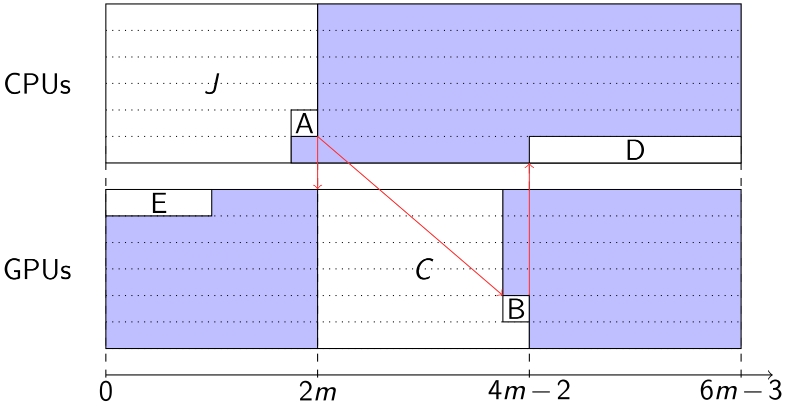
Algorithm 1

1: $S \leftarrow \emptyset$
2: **while** $S \neq T$ **do**
3: $R \leftarrow \{T_j \mid \Gamma^-(j) \subseteq S\}$: the set of ready tasks
4: $T_j \in R$: the task with the smallest possible starting time, with respect to the precedence constraints and the assignment variables
5: Schedule T_j on the processor which gives the smallest possible starting time
6: $S \leftarrow S \cup \{T_j\}$

Worst-case Example



Worst-case Example



New Scheduling Method

A recursive ranking method of each task is defined:

$$Rank(T_j) = \overline{p_j} x_j + \underline{p_j}(1 - x_j) + \max_{i \in \Gamma^+(j)} \{Rank(T_i)\}$$

The list of tasks is sorted in decreasing order of the ranks to give priority to the critical tasks.

New Linear Program

$(ILP2) =$ minimize λ subject to:

$$C_i + \overline{p_j} x_j + \underline{p_j}(1 - x_j) \leq C_j \qquad \forall j \in V,\ \forall i \in \Gamma^-(j)$$

$$C_j \leq \lambda \qquad \forall j \in V$$

$$\sum_{i \in A(j)} \frac{\overline{p_i} x_i}{m} + \overline{p_j} x_j + \underline{p_j}(1 - x_j) \leq C_j \qquad \forall j \in V$$

$$\sum_{i \in A(j)} \frac{\underline{p_i}(1 - x_i)}{k} + \overline{p_j} x_j + \underline{p_j}(1 - x_j) \leq C_j \qquad \forall j \in V$$

$$x_j \in \{0, 1\} \qquad \forall j \in V$$

New Algorithm

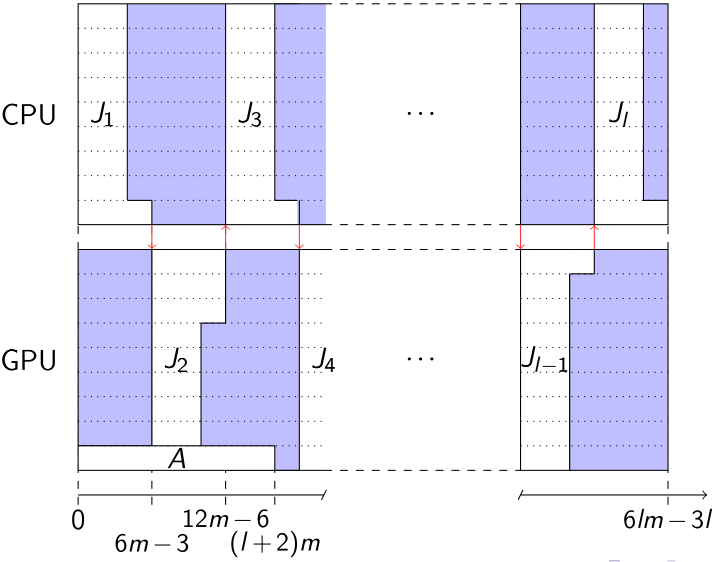
The refined algorithm is defined with:

- $(LP2)$
- Original rounding method
- List Scheduling with ranking of tasks

Proposition

The new algorithm has an approximation ratio of 6 and the bound is tight.

Worst-case Example



Experiments

Benchmark constructed from Chameleon:

- 6 applications of linear algebra for dense matrix
- 4 tilings of the matrices in sub-matrices
- 6 size of sub-matrices

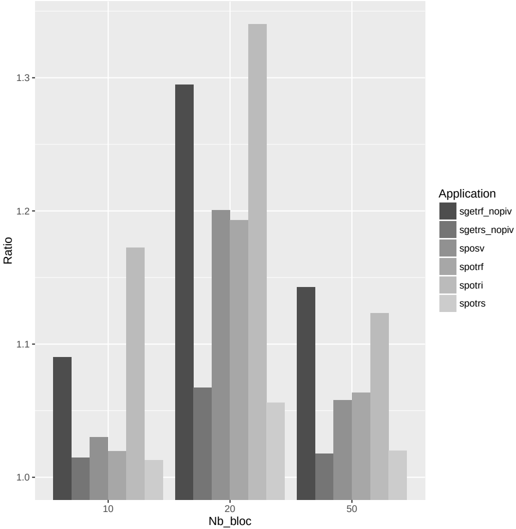
$\Rightarrow$ total of 24 configurations of each application.

Different couples (nb_cpu, nb_gpu) to simulate the hybrid machines.

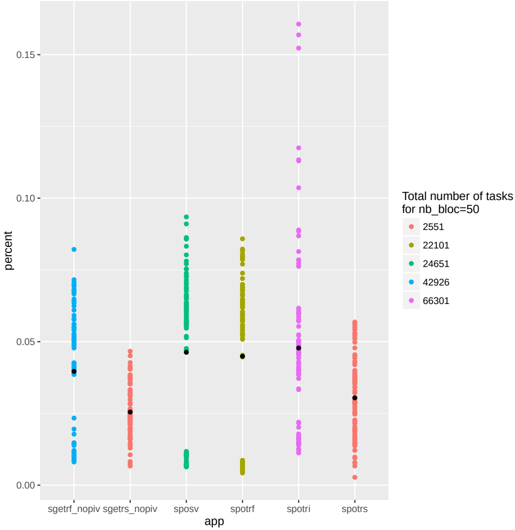
Different algorithms tested:

- HLP, refined HLP and HLP_ranked
- HEFT as a reference
- Greedy algorithm without LP

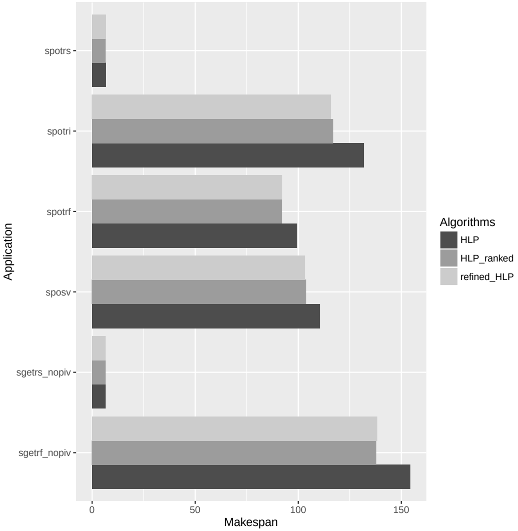
Analysis of HLP



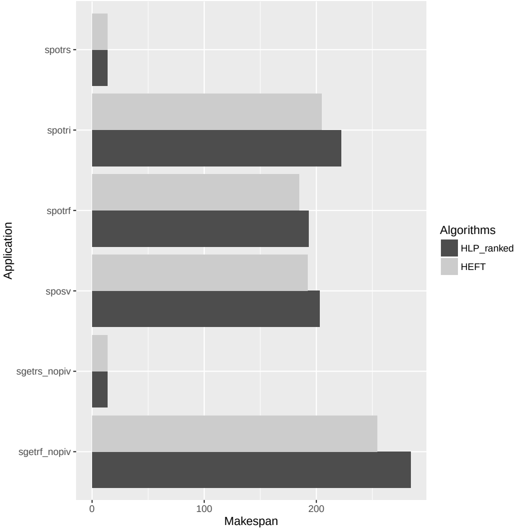
Analysis of HLP (cont.)



Comparison of the Algorithms



Comparison with HEFT



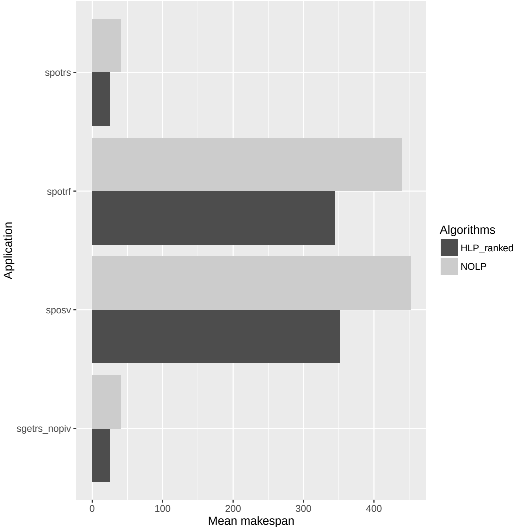
Algorithm without LP

- Decision rule for the assignment:

$$x_j = \begin{cases} 1 & \text{if } \frac{\overline{p_j}}{\sqrt{m}} \leq \frac{\underline{p_j}}{\sqrt{k}} \\ 0 & \text{otherwise} \end{cases}$$

- List Scheduling with ranking of tasks

LP versus Greedy



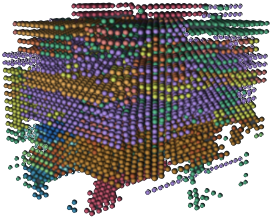
Conclusions and Contributions

- Worst-case example for HLP
  - Bound of approximation ratio tight at 6
- Design and analysis of the refined HLP algorithm
  - Approximation ratio of 6
  - Tight bound
- Generalization of HLP for more heterogeneous platforms
  - Tight $Q(Q+1)$ approximation analysis
- Improved lower bound for HEFT
  - Approximation ratio at least $(1-\frac{1}{e})m$
- Construction of a benchmark
  - 6 applications of dense matrix linear algebra
- Performance comparison of the algorithms

Future Work

- Improve the assignment step
  - More dynamic decision rules
  - Both for hybrid and heterogeneous platforms

- Consider the increasing complexity of the platforms
  - Different accelerators, I/O or visualization units
  - New constraints due to this heterogeneity

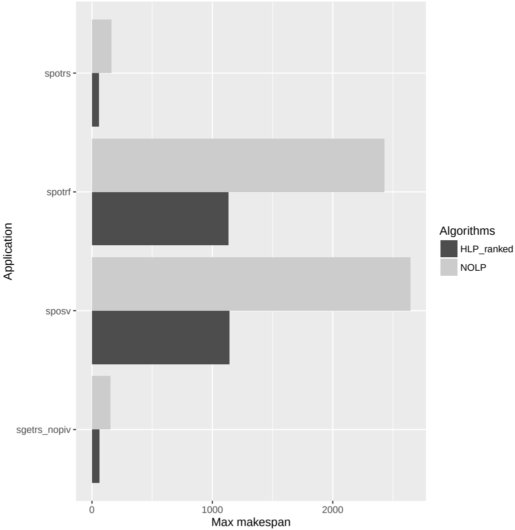


⇒ The design of an integrated scheduler for next-generation computing platforms

Thank you for your attention

Any question ?

Void



spotrs
spotrf
sposv
sgetrs_nopiv
Application
0
1000
2000
Max makespan
Algorithms
HLP_ranked
NOLP

(LP_Q) = minimize $\lambda = C_{end}$ subject to:

$$C_i + \sum_{q=1}^{Q} p_{j,q} x_{j,q} \leq C_j \qquad \forall j \in V,\ \forall i \in \Gamma^-(j)$$

$$\sum_{i \in A(j)} \frac{p_{j,q} x_{j,q}}{M_q} + \sum_{q=1}^{Q} p_{j,q} x_{j,q} \leq C_j \qquad \forall j \in V,\ \forall q = 1, \cdots, Q$$

$$\sum_{q=1}^{Q} x_{j,q} = 1 \qquad \forall j \in V$$

$$x_{j,q} \in \{0, 1\} \qquad \forall j \in V,\ \forall q = 1, \cdots, Q$$

1. $r_j = \arg\max_{q=1,\cdots,Q} \{x^R_{j,q}\} \quad \forall j \in V$
2. $x_{j,q} = \begin{cases} 1 & \text{if } q = r_j \\ 0 & \text{otherwise} \end{cases} \quad \forall j \in V, \ \forall q = 1, \cdots, Q$