Optimal Workload Scheduling Algorithm for Data-Parallel Applications on Heterogeneous Platforms based on Dynamic Programming

Laércio Lima Pilla

Optimal Workload Scheduling Algorithm for Data-Parallel Applications on Heterogeneous Platforms based on Dynamic Programming

Laércio Lima Pilla

RESEARCH REPORT
N° 9487
September 2022
Project-Team STORM

ISSN 0249-6399 ISRN INRIA/RR--9487--FR+ENG

Optimal Workload Scheduling Algorithm for Data-Parallel Applications on Heterogeneous Platforms based on Dynamic Programming

Laércio Lima Pilla*

Project-Team STORM

Research Report n° 9487 — September 2022 — 6 pages

Abstract: This report discusses a new algorithm for makespan minimization on situations where the workload can be partitioned among heterogeneous resources. Without making any assumptions regarding the behavior or shape of the functions giving the execution time on a resource, we are able to provide optimal solutions with a dynamic programming algorithm in $O(T^2n)$ for a workload of T tasks and n heterogeneous resources. This report includes a short state of the art, the problem's model, the algorithm, and its optimality proof.

Key-words: Scheduling, partitioning, optimization, algorithms, parallel processing, heterogeneous platforms, data-parallel application

* Univ. Bordeaux, CNRS, Bordeaux INP, Inria, LaBRI, UMR 5800, F-33400 Talence, France

RESEARCH CENTRE
BORDEAUX – SUD-OUEST

200 avenue de la Vieille Tour
33405 Talence Cedex

Algorithme d'ordonnancement optimal basé sur la programmation dynamique pour les applications aux données parallèles sur des plates-formes hétérogènes

Résumé : Ce rapport discute d'un nouvel algorithme pour la minimisation du makespan dans des situations où la charge de travail peut être partitionnée entre des ressources hétérogènes. Sans faire aucune hypothèse sur le comportement ou la forme des fonctions donnant le temps d'exécution sur une ressource, nous sommes capables de fournir des solutions optimales avec un algorithme de programmation dynamique en $O(T^2n)$ pour une charge de travail de T tâches et n ressources hétérogènes. Ce rapport comprend un bref état de l'art, le modèle du problème, l'algorithme, et sa preuve d'optimalité.

Mots-clés : Ordonnancement, partitionnement, optimisation, algorithmes, parallélisme, plates-formes hétérogènes, application de données parallèles

1 Context

Workload distribution algorithms can be employed to minimize the computation time, energy consumption, or other metrics for some kinds of applications. Their main idea is to distribute a given number of tasks T among n resources in order to optimize the metric of interest.

In the specific case of heterogeneous platforms, Khaleghzadeh, Manumachu, and Lastovetsky [4] presented a branch-and-bound algorithm to minimize the makespan of applications without any assumptions on the shape of the execution time functions with worst-case complexity in $O(T^3n^3)$. They have applied this solution for matrix multiplication and fast Fourier transform applications. Later, Khaleghzadeh et al. [3] adapted this algorithm to optimize both execution time and energy consumption with complexity in $O(n^3T^3\log(nT))$. They have also applied this solution to a gene sequencing application. We previously proposed OLAR [6] as an optimal greedy solution for minimizing the makespan of Federated Learning applications in $\Theta(n+T\log n)$ when the execution time functions are monotonically increasing. Finally, we have proposed a dynamic programming solution for minimizing the energy consumption of Federated Learning without any assumptions on the shape of the functions [5]. The solution is computed in $O(T^2n)$ by using an approach based on a new knapsack problem. This solution originally minimizes the sum of costs (instead of the maximum cost as in the makespan minimization problems), but we will show its adaptation to our problem later in this report.

2 Problem Definition

Our problem can be formulated similarly to the problem of scheduling independent tasks (1-D data) on heterogeneous resources [2, Chapter 6.1] [1], similarly to our previous works [5,6].

Consider n heterogeneous resources organized in a set $\mathcal{R} = \{1, 2, \ldots, n\}$ and a total workload of T identical, independent, and atomic tasks to be distributed. Each resource $i \in \mathcal{R}$ has an upper limit $U_i \in \mathbb{N}$ and a lower limit $L_i \in \mathbb{N}$ on the number of tasks it can compute, and its own cost function $C_i{:}[L_i, U_i] \to \mathbb{R}_{\geq 0}$ that represents the time it takes to compute with a given number of tasks. No assumptions are made regarding the behavior or shape of this function. We use $\mathcal{U} = \{U_1, \ldots, U_n\}$, $\mathcal{L} = \{L_1, \ldots, L_n\}$, and $\mathcal{C} = \{C_1, \ldots, C_n\}$ to represent the sets of upper limits, lower limits, and cost functions, respectively. Finally, consider the schedule $X = \{x_1, \ldots, x_n\}$ that assigns $x_i \in [L_i, U_i]$ tasks to each resource $i \in \mathcal{R}$.

Definition 1 (Makespan Optimization Problem)**.** *Given a problem instance* $(\mathcal{R}, T, \mathcal{U}, \mathcal{L}, \mathcal{C})$*, the goal is to find an optimal schedule* X^* *that minimizes the makespan* C_{max}*, i.e.:*

$$\textit{minimize}_X \; C_{max} \coloneqq \max_{i \in \mathcal{R}} C_i(x_i) \tag{1a}$$

$$\textit{subject to} \sum_{i \in \mathcal{R}} x_i = T, \tag{1b}$$

$$L_i \leq x_i \leq U_i, \; \forall i \in \mathcal{R} \tag{1c}$$

As customary, we focus on non-trivial, valid problem instances. We also assume that $T > n$ without loss of generality.

3 Dynamic Programming Solution

Our solution to this problem comes in the form of an adaptation of our previous total cost minimization algorithm [5]. This previous solution uses a dynamic programming algorithm that

solves the Multiple-Choice Minimum-Cost Maximal Knapsack Packing Problem ((MC)^{2}MKP), which is a generalization of the total cost minimization problem.

Consider $\mathcal{Z}_r(\tau)$ as an optimal solution value for a partial problem with the first r resources that schedules τ tasks. Assume that $\mathcal{Z}_r(\tau) \coloneqq \infty$ if no solution exists and that $\mathcal{Z}_0(0) \coloneqq 0$. $\mathcal{Z}_r(\tau)$ is defined in Equation (2) and its value can be recursively computed following Equation (3).

$$\mathcal{Z}_r(\tau) \coloneqq \min \left\{ \max_{i=1,\cdots,r} C_i(x_i) \,\middle|\, \sum_{i=1}^{r} x_i = \tau \right\} \tag{2}$$

$$\mathcal{Z}_r(\tau) = \min_{j \in [L_r, U_r], j \leq \tau} \max\left(\mathcal{Z}_{r-1}(\tau - j), C_r(j)\right) \tag{3}$$

We use the ideas behind Equation (3) to propose Algorithm 1. It differs from the original (MC)^{2}MKP algorithm in [5] by its use of maximums instead of sums, and by simplifying the end of the algorithm. The algorithm uses two matrices, K and I, to store the minimal costs that are progressively computed and the schedules that are part of these solutions, respectively.

The algorithm first stores all possible solutions for $\mathcal{Z}_1$ (lines 7–9) and then iteratively computes the optimal solutions for an increasing number of resources and tasks to schedule (lines 10–19). As given by Equation (3), the best solution for a given resource i and t tasks is computed based on the previous best solution with $i-1$ resources. By comparing the maximum between the previous best solution and the cost for a given number of tasks in resource i, the makespan of this partial solution is computed. By keeping the minimal solution (lines 13–16), the minimal makespan is found.

By the end of Algorithm 1, the makespan is obtained in line 20. The optimal schedule, in its turn, is obtained by going in the reverse order of resources through matrix I (lines 21–24). With some minor changes to the loops and the end of this algorithm, it can be adapted (I) to find the minimal makespan with the maximum number of tasks scheduled possible, and (II) to work with non-contiguous intervals of possible schedules for each resource. We recommend our readers to check the original algorithm in [5] for an example.

Algorithm 1 shows a space bound in $O(Tn)$ and it requires $O(T\sum_{i=1}^{n}(U_i - L_i + 1))$ operations. In the worst-case scenario, at most $T+1$ assignments are possible for each resource (from 0 to T). This gives us a worst-case complexity in $O(T^2 n)$.

3.1 Proof of optimality

The optimality of Algorithm 1 can be easily demonstrated by induction, and it is very similar to a previous proof for the total cost minimization problem [5]. As row r in matrix K represents the optimal solutions found for $\mathcal{Z}_r$, our proof will be based on the properties of Equations (2) and (3). Our algorithm also benefits from the fact that the maximum and minimum operations are commutative and associative, so the order that the resources and number of tasks are considered do not have an impact on the optimal makespan found.

Lemma 1. *The solutions in $\mathcal{Z}_1$ are optimal.*

Proof. The only possible solutions for $\mathcal{Z}_1$ are $\mathcal{Z}_1(j) = C_i(j)$ for all $j \in [L_i, U_i]$, therefore they are optimal. □

Lemma 2. *If the solutions in $\mathcal{Z}_i$ are optimal, then the solutions in $\mathcal{Z}_{i+1}$ are also optimal.*

Proof. By Equation (3), the value of $\mathcal{Z}_{i+1}(\tau)$ for $\tau \in [0, T]$ is the minimum among all maximums between $\mathcal{Z}_i(\tau - j)$ and $C_{i+1}(j)$ for $j \in [L_{i+1}, U_{i+1}], j \leq \tau$. To consider that there would be another schedule with a smaller value is a contradiction: it would require having a sub-optimal

Algorithm 1 DP solution for the Makespan Optimization Problem.

Input: Set of resources $\mathcal{R}$, number of tasks to schedule T, set of upper limits $\mathcal{U}$, set of lower limits $\mathcal{L}$, set of cost functions $\mathcal{C}$.
Output: Makespan C_{max}, optimal schedule X.

1: **for** $i = 1, \ldots, n$ **do** ▷ *Initialization of minimal costs*
2: **for** $t = 0, \ldots, T$ **do** ▷ *and partial solutions matrices.*
3: $K[i][t] \leftarrow \infty$; $I[i][t] \leftarrow \varnothing$
4: **end for**
5: $x_i \leftarrow \varnothing$
6: **end for**
7: **for** $j = L_r, \ldots, U_r$ **do** ▷ *Only solutions for* $\mathcal{Z}_1$.
8: $K[1][j] \leftarrow C_1(j)$; $I[1][j] \leftarrow j$
9: **end for**
10: **for** $i = 2, \ldots, n$ **do** ▷ *Computes* $\mathcal{Z}_i$ *for all capacities.*
11: **for** $j = L_r, \ldots, U_r$ **do** ▷ *Using all possible schedules for* i.
12: **for** $t = j, \ldots, T$ **do**
13: **if** $\max(K[i-1][t-j], C_i(j)) < K[i][t]$ **then**
▷ *Best solution for* $\mathcal{Z}_i(t)$ *so far.*
14: $K[i][t] \leftarrow \max(K[i-1][t-j], C_i(j))$
15: $I[i][t] \leftarrow j$
16: **end if**
17: **end for**
18: **end for**
19: **end for**
20: $C_{max} \leftarrow K[n][T]$ ▷ *Finds* C_{max}.
21: $t \leftarrow T$
22: **for** $i = n, \ldots, 1$ **do** ▷ *Finds* X.
23: $j \leftarrow I[i][t]$; $x_i \leftarrow j$; $t \leftarrow t - j$
24: **end for**
25: **return** C_{max}, X

$\mathcal{Z}_i$, or a value that is smaller than the minimum among all possible optimal solutions for $\mathcal{Z}_{i+1}(\tau)$. Therefore, $\mathcal{Z}_{i+1}$ is optimal. □

Theorem 1. *$\mathcal{Z}_n(T)$ provides the optimal solution for our Makespan Optimization Problem.*

Proof. Lemmas 1 and 2 prove the optimality of the base case and the induction step, so $\mathcal{Z}_n(T)$ is optimal. □

References

[1] O. Beaumont, V. Boudet, A. Petitet, F. Rastello, and Y. Robert. A proposal for a heterogeneous cluster scalapack (dense linear solvers). *IEEE Transactions on Computers*, 50(10):1052–1070, 2001.

[2] Henri Casanova, Arnaud Legrand, and Yves Robert. *Parallel algorithms.* CRC Press, 2008.

[3] Hamidreza Khaleghzadeh, Muhammad Fahad, Arsalan Shahid, Ravi Reddy Manumachu, and Alexey Lastovetsky. Bi-objective optimization of data-parallel applications on heterogeneous

hpc platforms for performance and energy through workload distribution. *IEEE Transactions on Parallel and Distributed Systems*, 32(3):543–560, March 2021.

[4] Hamidreza Khaleghzadeh, Ravindranath Reddy Manumachu, and Alexey Lastovetsky. A novel data-partitioning algorithm for performance optimization of data-parallel applications on heterogeneous hpc platforms. *IEEE Transactions on Parallel and Distributed Systems*, 29(10):2176–2190, Oct 2018.

[5] Laércio Lima Pilla. Scheduling Algorithms for Federated Learning with Minimal Energy Consumption. working paper or preprint, September 2022.

[6] Laércio Lima Pilla. Optimal task assignment for heterogeneous federated learning devices. In *2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 661–670, 2021.

RESEARCH CENTRE
BORDEAUX – SUD-OUEST

200 avenue de la Vieille Tour
33405 Talence Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399