Scheduling Malleable Jobs Under Topological Constraints

Evripidis Bampis[1], Konstantinos Dogeas[1], Alexander Kononov[2], Giorgio Lucarelli[3], Fanny Pascual[1]

[1] Sorbonne Université, CNRS, Laboratoire d'Informatique de Paris 6, LIP6, F-75005 Paris, France
{firstname.lastname}@lip6.fr

[2] Sobolev Institute of Mathematics and Novosibirsk State University, Novosibirsk, Russia
alvenko@math.nsc.ru

[3] University of Lorraine, Metz, France
giorgio.lucarelli@univ-lorraine.fr

1 Introduction

High Performance Computers (HPCs) are widely used to run applications of great societal importance, due to their extreme computational power. As the complexity of platforms increases, designers turn their focus on the scheduling algorithms implemented on the platforms. The need for algorithms, which take into consideration the various features of HPCs, is crucial. In this work, we propose generic scheduling algorithms for HPC platforms taking into account communication issues as well as the existence of input/output (I/O) nodes.

2 Model

We model the platform by distinguishing two kinds of nodes : a set $\mathcal{V}^{\mathrm{C}}$ of m^{C} nodes dedicated to computations, and a set $\mathcal{V}^{I/O}$ of $m^{I/O}$ nodes that are entry points to a high performance file system. Let $\mathcal{V} = \mathcal{V}^{\mathrm{C}} \cup \mathcal{V}^{I/O}$ and $m = m^{I/O} + m^{\mathrm{C}}$. Usually, $m^{I/O} << m^{\mathrm{C}}$. We assume that each node can either be a computing or an I/O node. Furthermore, we suppose that any computing or I/O node is dedicated to one application throughout its execution, meaning that two jobs cannot use the same node simultaneously. The network topology considered in this work is the line. In a line topology, all nodes (computing and I/O) form a single connected component, each one connected to two other nodes, except the two nodes in the extremities. We assume that the localisation of every node within the topology is known. In lines this can be very easily done, by numbering the nodes from left to right.

We see applications as jobs which are queued in a set $\mathcal{J}$. The total number of jobs is n. We distinguish two models with respect to the computing need of a job. In the *rigid* model a job $j \in \mathcal{J}$ requires a fixed number of computing nodes $q_j \leq m^{\mathrm{C}}$. The processing time is also fixed, denoted by p_j. In the *malleable* model a job $j \in \mathcal{J}$ asks for a number of computing nodes Q_j, and the scheduler can decide the number of computing nodes $q_j \leq Q_j$ to be used for its execution. The exact processing time of the job j depends on the number of assigned computing nodes. Each job j has a required execution load, denoted by a_j. Let $f : \mathbb{N} \to \mathbb{R}$ be a speed-up non-increasing and convex function. The processing time of j is defined as $p_j = a_j f(q_j)$. In other words, jobs are *monotonic* [2]. In this paper, we consider two cases. In the *generalized-malleable model*, the function f is an arbitrary convex non-increasing function. In the *proportional-malleable model*, the total work does not depend on the number of computing nodes assigned to it : $f(q_j) = \frac{1}{q_j}$ and hence the processing time is $p_j = \frac{a_j}{q_j}$. In addition, jobs have a demand for a specific I/O node, which is the entry point to the file system.

Due to the parallelization of the HPC jobs, all the parts of a job need to communicate with each other to complete the execution. We refer to this kind of data flows in the network as *computational communications*. Furthermore, jobs need to read data from the disk when they start execution and write data to the disk once they finish. We refer to this kind of data flows as

I/O communications. Given the direct topology of the line, each node is occupied when traffic needs to pass "through" itself in order to arrive to the destination. If this node is allocated to a different job, then we have the undesirable effect of delaying the completion time of one job in order to handle the traffic from a different job. In order to avoid both the aforementioned data flows, we use the definitions of *contiguity* and *locality* introduced in [1, 3] to restrict the number of possible allocations of a job. Given the overhead of distant communications, we may add a new kind of locality by introducing a limit on the number of machines that may be used for the execution of the jobs. This limitation is parameterized by a common resource requirement $Q_j = Q$ for all jobs in $\mathcal{J}$ in the malleable model. The value of Q is chosen based on the size and the structure of the platform. We call instances satisfying this kind of locality as *uniform* instances.

3 Results

Our work extends the model proposed by Bleuse et al. [1] for rigid jobs and introduces a new mechanism for addressing the energy/performance trade-off by using a malleable model. This is a first step towards more realistic models. The goal of our scheduling problem is to minimize the makespan with respect to contiguity and locality constraints in a line topology. We first reduce the approximability gap for the rigid model for which a 6-approximation algorithm is known [1]. We show that, for any $\epsilon > 0$, there is no approximation algorithm with ratio $\frac{3}{2} - \epsilon$ for the problem of scheduling rigid jobs with respect to contiguity and locality constraints, unless $\mathcal{P}$=$\mathcal{NP}$. Furthermore, we present complexity results for the proportional-malleable models, implying also the complexity of the generalized-malleable model. We show that the problem is $\mathcal{NP}$-hard even in very restricted cases. Moreover, we first deal with the proportional-malleable model in uniform instances and we propose a novel polynomial-time 2-approximation algorithm. Then, we present an approximation algorithm for the generalized-malleable problem. This algorithm is analyzed in a computational way and it achieves an approximation ratio which depends on the function f.

4 Conclusion

In this work we studied the makespan minimization problem on the malleable and the rigid models under contiguity and locality constraints. We give inapproximability results for the rigid model and complexity results for the malleable one. Focusing on the malleable model, we give approximation algorithms for the proportional uniform setting as well as the generalized one. As future work, it would be interesting to search for a constant factor approximation algorithm for the generalized-malleable problem and further close the approximability gap for both malleable and rigid settings. Furthermore, one can try to extend the topological constraints and therefore the algorithms mentioned in this work in more complex and differently structured topologies.

Références

[1] Raphaël Bleuse, Konstantinos Dogeas, Giorgio Lucarelli, Grégory Mounié and Denis Trystram *Interference-Aware Scheduling Using Geometric Constraints*. Euro-Par 2018 : Parallel Processing - 24th International Conference on Parallel and Distributed Computing, Turin, Italy.

[2] Gregory Mounie, Christophe Rapine and Denis Trystram. *A 3/2-Approximation Algorithm for Scheduling Independent Monotonic Malleable Tasks*. SIAM J. Comput.

[3] Raphaël Bleuse, Giorgio Lucarelli and Denis Trystram *A Methodology for Handling Data Movements by Anticipation : Position Paper*. Euro-Par 2018 : Parallel Processing Workshops - Euro-Par 2018 International Workshops, Turin, Italy.