POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

A stochastic approach for scheduling AI training jobs in GPU-based systems

Tesi di Laurea Magistrale in
Mathematical Engineering - Ingegneria Matematica

Lorenzo Marchi, 10530934

Advisor:
Prof. Danilo Ardagna

Co-advisors:
Federica Filippini

Academic year:
2021-2022

Abstract: Deep Learning methods are currently used to address a variety of complex tasks. This is partially motivated by the fact that these models are now trained on GPUs, expanding the range of problems that can be solved in a reasonable computating time. This has caused the demand for high-performance GPU-based cloud servers to increase dramatically, making it necessary for Cloud Service Providers (CSPs) to manage that demand effectively. In this thesis, we optimize the scheduling of Deep Learning training jobs from the perspective of a CSP running a data center, efficiently selecting resources for the execution of each job in order to minimize average power consumption. We modeled this problem through a Mixed Integer Linear Programming (MILP) formulation, and we developed a stochastic heuristic that, exploiting the probability distribution of early termination, determines how to vary the resource assignment during the execution of each job to minimize the expected value of the energy cost while still fulfilling deadlines. We set up an extensive experimental campaign to perform simulations and test the quality of the solution identified by our method. The results show that our heuristic guarantees significantly better results than other methods in the literature, with a percentage energy cost reduction of about 38-40% on average. We also prove the applicability of our method in real-world situations, as obtaining optimal schedules for systems of up to 100 nodes and 400 concurrent jobs requires less than 60 seconds. Finally we evaluated the effectiveness of GPU sharing, that is, running multiple jobs in a single GPU. The results demonstrate that, depending on the workload and GPU memory, this possibility can reduce the percentage cost by 17-29% on average.

Key-words: Deep Learning, GPU cluster, Scheduling, Average energy consumption minimization.

1. Introduction

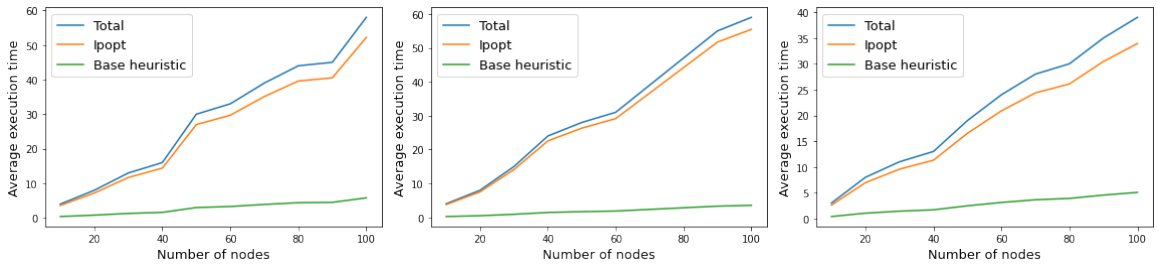
Nowadays, different kinds of problems are tackled with Deep Learning (DL) algorithms. Partially, this is because these models are now trained using GPUs instead of CPUs, achieving an execution speedup of about 5-40x [1], extending the set of problems that can be solved in a reasonable computating time.
The problem that arise in this context is that high-performance GPU-based servers are cost-prohibitive (about 200k USD for high-end systems like NVIDIA DGX A100 [2]). This has caused the demand for cloud servers

of this type to increase dramatically, making it necessary for Cloud Service Providers (CSPs) to manage that demand effectively.
The goal of this work is to optimize the scheduling of Deep Learning training jobs from the perspective of a CSP running a data center, efficiently selecting resources for the execution of each job in order to minimize power consumption costs.
To the best of our knowledge, methods to optimize this problem available in the literature are (i) simple job scheduling mechanisms such as Earliest-Deadline-First (EDF or First-in-First-Out (see [3]), (ii) more elaborate heuristics such as Random Greedy (RG) and Path Relinking, presented in [4, 5]. A limitation of these approaches is that they consider only the worst-case execution time in searching for an optimal schedule. Instead, our idea to address the problem is to exploit stochastic information about the training of DL jobs to minimize the expected value of energy costs.
The reference scenario is a CSP composed of several machines, each with different numbers and types of GPUs. Multiple Artificial Intelligence (AI) training jobs are submitted over time, and there is no information about future arrivals. Each job has different characteristics and is associated with an execution deadline and a priority for fulfilling it.
While optimizing energy consumption, the priority is to execute jobs by the due date; for this purpose, we model the due date violation with a tardiness cost, depending on the priority. To enforce its importance for the global optimization problem, this is set to be an order of magnitude higher than the energy cost.
A certain number of GPUs is assigned to each job; in particular, they can be executed on multiple nodes exclusively (multi-node jobs) or on a single node, shared with other jobs. Moreover, we considered GPU sharing, i.e., the possibility of executing multiple jobs on a single GPU (mini-jobs). Indeed, nowadays the GPUs available on the market are increasingly performing and have a large amount of memory, which enables multiple jobs to be trained simultaneously, saving in power consumption and resources although increasing the execution time due to overhead communication.
To properly select the resources to be allocated, it is necessary to estimate the time needed for the training depending on the chosen configuration. In addition to determining the worst-case execution time to reach the maximum number of epochs, we infer a probability distribution for early termination. From these two pieces of information, we define the execution time as a stochastic variable.
We formalize this problem through a Mixed Integer Linear Programming formulation. Due to the stochasticity of the parameters and and the large numbers of variables and constraints, this is too demanding to be solved directly.
Therefore, we develop a heuristic (starting from the work in [6]) that, using the probability distribution of the number of epochs required for execution, determines a dynamic schedule (i.e., it changes the resources) for each job such that we minimize the expected value of the energy cost while still fulfilling the deadline in the worst-case.
The idea behind this heuristic is to start with a low-power configuration and then gradually increase the assigned resources. This way, we reduce costs when a job ends early, but we can still recover otherwise.
We implement such heuristic and perform simulations with real-world data to validate its efficiency. We compare the results obtained with our stochastic method against state-of-the-art approaches. We show that with our approach, a significant increase in performance is achieved by obtaining an average percentage reduction of the energy cost of about $38 - 40\%$ with respect to RG and EDF.
We also run simulations to evaluate the effectiveness of GPU sharing, comparing the power consumption cost with a scenario where GPUs are devoted to single jobs. The results show how the possibility of co-locating multiple jobs on single GPUs yields a reduction in the average percentage cost in a range of $17 - 29\%$, depending on the workload and GPUs memory.
Finally, we demonstrate the applicability of our method in real-world situations, as we obtain optimal schedules for systems of up to 100 nodes and 400 concurrent jobs in less than 60 seconds.
This thesis is structured as follows:
In Section 2, we present the state-of-the-art for the problem we want to tackle, and in Section 3 we describe in details the work from the literature that is the basis of our stochastic approach.
In Section 4, we state the optimal job scheduling problem we address in this work, and in Section 5 we provide a mathematical formulation for it. In Section 6, we explain the stochastic heuristic developed to solve our model. In Section 7, we present the experimental setup for the simulations, and in Section 8 we show the obtained results. Finally, in Appendix A we give an overview of the code developed to implement the heuristic and explain how to use it to run the simulations.

2. Related Work

There are many challenges in managing GPU-accelerated clusters efficiently. It is necessary to define effective solutions for job scheduling and resource allocation to maximize performance and minimize power consumption

point of our problem. For systems of 100 nodes and 1000 jobs, this value is about 400 in the high-rate scenario, 200 for the exponential inter-arrival scenario, and 100 for the low-rate scenario.
Thus, we solved a single instance of the problem with our heuristic by generating a number of jobs equal to $4N$, $2N$, or N, where N is the number of nodes, submitting them all at the initial instant.
The results are shown in Figure 14.



(a) Exponential inter-arrival (b) High rate (c) Low rate

Figure 14: Average execution times needed for STS to solve a single instance of the problem

From the plots, we observe that the average execution time of a single instance has a linear dependency on the number of jobs to be trained concurrently, and is always under a minute, even for the larger systems.
The average execution time of the *exponential* and *high* cases is almost comparable, while, as expected, it is significantly reduced for the *low* case.
We separated the execution time of the non-linear solver of the stochastic model *Ipopt* from the rest of the heuristic (*preprocessing*, *assignment*, and *postprocessing*); in practice, about 90% of the total execution time is due to *Ipopt*.
This result was expected given the complexity of the stochastic model; to speed up the heuristic, it would be necessary to solve it more effectively by tackling it with a problem-based method.
In any case, the results obtained using *Ipopt* are already satisfying, being the resolution times negligible compared to the overall training times of the Deep Learning jobs considered for the simulations.

9. Conclusion

In this thesis, we presented a stochastic approach to model and tackle the optimal scheduling problem for AI training jobs in GPU-based systems. We provided a mathematical formulation of the problem by defining the execution time as a stochastic variable in Sections 4 and 5. Furthermore, we developed a stochastic heuristic that allocates jobs to the available resources to minimize the average energy cost while meeting the imposed deadlines as described in Section 6.
We set up an extensive experimental campaign and performed simulations to test the quality of our method by comparing it with other literature approaches.
The results, presented in Section 8, confirm that our heuristic guarantees significantly better results than the existing ones, with a percentage reduction in energy cost of about $38 - 40\%$.
This improvement is undoubted because stochastic information about the end of job training is used in our approach, which is not considered by the other methods.
However, we have shown, by proving the inefficiency of the reference heuristics when modified with the average execution time, that to address the problem stochastically it is necessary to develop a complete method that also considers the worst-case scenario, like ours does.
In order to assess the benefit of GPU sharing, we also ran simulations and compared the power consumption cost when this was or was not allowed. The results demonstrate that, depending on the workload and GPU memory, the possibility of co-locating multiple jobs on a single GPU can reduce the cost percentage between 17% and 29%.
Finally, by showing that solving instances of the problem with up to 100 nodes and 400 concurrent jobs requires less than 60 seconds, we demonstrated that our method could be applied efficiently in real-world scenarios.
One possible improvement of the proposed heuristic is to solve the stochastic model described in Section 3 more efficiently as described in [6]. It has to be solved at every rescheduling point for every job and every GPU type. Since it is crucial for the nature of the problem to obtain results in a short time, we set a low maximum number of iterations for the interior point method. This often causes the global optimum not to be found, so determining a specific solution for our nonlinear model could improve the method performance.

References

[1] Souley Madougou, Ana Varbanescu, Cees de Laat, and Rob van Nieuwpoort. The landscape of gpgpu performance modeling tools. *Parallel Computing*, 56:18–33, 2016. ISSN 0167-8191. doi: https://doi.org/10.1016/j.parco.2016.04.002. URL https://www.sciencedirect.com/science/article/pii/S0167819116300114.

[2] Vijay Anand. Nvidia dgx a100. *[Online]. Available: https://www.hardwarezone.com.sg/tech-news-nvidia-dgx-a100-supercomputer-super-performance-fight-covid-19 (visited on 04/10/2021).*

[3] Marcelo Amaral, Jordà Polo, David Carrera, Seetharami Seelam, and Malgorzata Steinder. Topology-aware gpu scheduling for learning workloads in cloud environments. In *HPCNSA Proc.* ACM, 2017.

[4] Federica Filippini, Marco Lattuada, Michele Ciavotta, Arezoo Jahani, Danilo Ardagna, and Edoardo Amaldi. A path relinking method for the joint online scheduling and capacity allocation of dl training workloads in gpu as a service systems. *IEEE Transactions on Services Computing*, pages 1–16, 2022.

[5] F. Filippini, D. Ardagna, M. Lattuada, E. Amaldi, M. Riedl, K. Materka, P. Skrzypek, M. Ciavotta, F. Magugliani, and M. Cicala. Andreas: Artificial intelligence training scheduler for accelerated resource clusters. In *2021 8th International Conference on Future Internet of Things and Cloud (FiCloud)*, pages 388–393, Los Alamitos, CA, USA, aug 2021. IEEE Computer Society. doi: 10.1109/FiCloud49777.2021.00063. URL https://doi.ieeecomputersociety.org/10.1109/FiCloud49777.2021.00063.

[6] Jonatha Anselmi and Bruno Gaujal. Energy Optimal Activation of Processors for the Execution of a Single Task with Unknown Size. In *30th International Symposium on the Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, Nice, France, October 2022. URL https://hal.archives-ouvertes.fr/hal-03682485.

[7] Jacob R. Lorch and Alan Jay Smith. Improving dynamic voltage scaling algorithms with pace. In *Proceedings of the 2001 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '01, page 50–61, New York, NY, USA, 2001. Association for Computing Machinery. ISBN 1581133340. doi: 10.1145/378420.378429. URL https://doi.org/10.1145/378420.378429.

[8] Baolin Li, Tirthak Patel, Siddharth Samsi, Vijay Gadepally, and Devesh Tiwari. Miso: Exploiting multi-instance gpu capability on multi-tenant gpu clusters. In *Proceedings of the 13th Symposium on Cloud Computing*, SoCC '22, page 173–189, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450394147. doi: 10.1145/3542929.3563510. URL https://doi.org/10.1145/3542929.3563510.

[9] Y. Peng, Y. Bao, Y. Chen, C. Wu, C. Meng, and W. Lin. Dl2: A deep learning-driven scheduler for deep learning clusters. *IEEE TPDS*, 32(08):1947–1960, 2021.

[10] Yanghua Peng, Yixin Bao, Yangrui Chen, Chuan Wu, and Chuanxiong Guo. Optimus: An efficient dynamic resource scheduler for deep learning clusters. In *EUROSYS*, 2018.

[11] Vaibhav Saxena, K. R. Jayaram, Saurav Basu, Yogish Sabharwal, and Ashish Verma. Effective elastic scaling of deep learning workloads. In *MASCOTS proceedings*, pages 1–8, 2020.

[12] Wencong Xiao, Romil Bhardwaj, Ramachandran Ramjee, Muthian Sivathanu, Nipun Kwatra, Zhenhua Han, Pratyush Patel, Xuan Peng, Hanyu Zhao, Quanlu Zhang, et al. Gandiva: Introspective cluster scheduling for deep learning. In *USENIX OSDI*, 2018.

[13] Kshiteej Mahajan, Arjun Balasubramanian, Arjun Singhvi, Shivaram Venkataraman, Aditya Akella, Amar Phanishayee, and Shuchi Chawla. Themis: Fair and efficient GPU cluster scheduling. In *USENIX (NSDI 20)*, pages 289–304, 2020.

[14] Shubham Chaudhary, Ramachandran Ramjee, Muthian Sivathanu, Nipun Kwatra, and Srinidhi Viswanatha. Balancing efficiency and fairness in heterogeneous gpu clusters for deep learning. In *EUROSYS*, 2020.

[15] Y. Bao, Y. Peng, and C. Wu. Deep learning-based job placement in distributed machine learning clusters. In *IEEE INFOCOM 2019*, pages 505–513, 04 2019.

[16] Hongyun Liu, Peng Chen, and Zhiming Zhao. Towards a robust meta-reinforcement learning-based scheduling framework for time critical tasks in cloud environments. In *2021 IEEE 14th International Conference on Cloud Computing (CLOUD)*, pages 637–647, 2021. doi: 10.1109/CLOUD53861.2021.00082.

[17] S. Zhang, C. Wang, and A. Y. Zomaya. Robustness analysis and enhancement of deep reinforcement learning-based schedulers. *IEEE Transactions on Parallel Distributed Systems*, (01):1–12, nov 5555. ISSN 1558-2183. doi: 10.1109/TPDS.2022.3218649.

[18] Chen Chen, Yingwen Chen, Zhaoyun Chen, Jianchen Han, and Guangtao Xue. Pickyman: A preemptive scheduler for deep learning jobs on gpu clusters. In *2022 IEEE International Performance, Computing, and Communications Conference (IPCCC)*, pages 120–129, 2022. doi: 10.1109/IPCCC55026.2022.9894345.

[19] Haoyu Wang, Guoxin Liu, and Haiying Shen. Cooperative job scheduling and data allocation in data-intensive parallel computing clusters. *IEEE Transactions on Cloud Computing*, pages 1–14, 2022. doi: 10.1109/TCC.2022.3206206.

[20] Yuan Yao, Marco Paolieri, and Leana Golubchik. Scheduling to optimize sojourn time of successful jobs, 2022. URL `https://arxiv.org/abs/2205.12891`.

[21] Gingfung Yeung, Damian Borowiec, Renyu Yang, Adrian Friday, Richard Harper, and Peter Garraghan. Horus: Interference-aware and prediction-based scheduling in deep learning systems. *IEEE TPDS*, 33(1): 88–100, 2022.

[22] Hugo Larochelle, Dumitru Erhan, and Y. Bengio. Zero-data learning of new tasks. volume 2, pages 646–651, 01 2008.

[23] Amazon web service pricing list, 2021. URL `https://aws.amazon.com/ec2/pricing/on-demand/?nc1=h_ls`.

[24] Azure cloud services pricing list, 2021. URL `https://azure.microsoft.com/en-us/pricing/details/cloud-services/`.

[25] Eugenio Gianniti., Li Zhang., and Danilo Ardagna. Performance prediction of gpu-based deep learning applications. In *CLOSER*, volume 1, pages 279–286, 2019.

[26] Airlab website. URL `https://airlab.deib.polimi.it`.

[27] URL `https://github.com/FFede0/GPUspb/tree/StochasticScheduling`.

[28] Ipopt website. URL `https://coin-or.github.io/Ipopt/index.html#Overview`.

[29] Pyomo website. URL `http://www.pyomo.org/`.

[30] William E Hart, Jean-Paul Watson, and David L Woodruff. Pyomo: modeling and solving mathematical programs in python. *Mathematical Programming Computation*, 3(3):219–260, 2011.

[31] Flask website. URL `https://flask.palletsprojects.com/en/2.2.x/`.

[32] Restc-cpp repository. URL `https://github.com/jgaa/restc-cpp`.