POLITECNICO
MILANO 1863

# Job Scheduling and Optimal Capacity Allocation Problems for Deep Learning Training Jobs

M.Sc. Thesis of
**Federica Filippini**

Supervisor:
Prof. Danilo Ardagna

Co-supervisors:
Prof. Edoardo Amaldi
Dr. Marco Lattuada

Programme:
*Mathematical Engineering*

Department of Mathematics
Politecnico di Milano

# Contents

# List of Figures

# Abstract

The Deep Learning (DL) paradigm has gained remarkable popularity in the last few years. DL models are often used to tackle complex problems in the fields of, e.g., image recognition and healthcare; however, the training of such models requires a very large computational power. The recent adoption of GPUs as general-purpose parallel processors has partially fulfilled this need, but the high costs related to this technology, even in the Cloud, dictate the necessity of devising efficient capacity planning and job scheduling algorithms to reduce operational costs via resource sharing.

The proposed work addresses the capacity planning and job scheduling problems jointly, considering both the Cloud end-users' and the Cloud Providers' perspective. The complexity of the problem, that represents a great challenge in terms of modeling and solvability, is even exacerbated, in the envisioned scenario, by the fact that capacity allocation and scheduling are analysed in an online setting. DL training jobs are therefore submitted in a continuous fashion, so that no scheme can be detected in their arrival times or characteristics, particularly in terms of priority.

Two different Mixed Integer Linear Programming (MILP) formulations, as well as alternative heuristic methods, inspired to greedy and local search techniques, have been developed to shape efficient and scalable solutions for the proposed problem.

An extensive experimental campaign proves the feasibility of the developed approaches for practical scenarios, showing considerable improvements in the computational time required to determine good-quality solutions. Moreover, significant cost savings are attained with respect to first principle methods based on, e.g., first-in-first-out or earliest deadline first, achieving between 50 and 80% cost reductions on average.

# Sommario

I modelli di Deep Learning (DL) hanno assunto una notevole popolarità negli ultimi anni. Essi vengono frequentemente applicati alla soluzione di problemi complessi, in settori come riconoscimento di immagini e assistenza sanitaria; l'addestramento di tali modelli richiede, tuttavia, una potenza computazionale molto ampia. La recente adozione delle GPU per il calcolo parallelo ha parzialmente soddisfatto questa esigenza, ma i costi elevati legati a questa tecnologia, anche nel Cloud, impongono la necessità di ideare algoritmi efficienti di pianificazione della capacità e scheduling delle applicazioni, allo scopo di ridurre i costi operativi attraverso un'opportuna condivisione delle risorse.
Il lavoro proposto affronta i problemi di allocazione della capacità e scheduling delle applicazioni in modo congiunto, considerandoli sia dalla prospettiva degli utenti finali Cloud sia dal punto di vista dei Cloud Provider. La complessità del problema, che rappresenta una grande sfida sia in termini di modellizzazione sia dal punto di vista della risolvibilità, é persino esacerbata, nello scenario proposto, dal fatto che i problemi di allocazione della capacità e scheduling di applicazioni sono analizzati in un contesto online. Nessuno schema può, pertanto, essere rilevato nell'ordine di arrivo delle applicazioni, le cui caratteristiche, soprattutto in termini di priorità, sono del tutto impredicibili. Sono state sviluppate due formulazioni matematiche e diversi metodi euristici, ispirati ad algoritmi greedy e local search, allo scopo di fornire soluzioni efficienti e scalabili per l'ottimizzazione di questi problemi.
Un'ampia campagna sperimentale ha dimostrato l'applicabilità degli approcci proposti a scenari pratici, mostrando notevoli progressi dal punto di vista del tempo computazionale richiesto dagli algoritmi proposti rispetto alla proposta di letteratura precedente. Significativi risparmi sui costi sono stati ottenuti, inoltre, rispetto a modelli di gestione delle risorse, basati, ad esempio, su first-in-first-out o earliest deadline first, con guadagni compresi, in media, tra il 50 e l'80%.

CHAPTER 1

# Introduction

The Deep Learning (DL) paradigm, as part of the wide family of Machine Learning methods, has gained remarkable popularity in the last years. This tool allows to process huge amount of data and to extract patterns also from poorly structured information. It relies on Neural Networks (NNs): nested structures of artificial neurons, organized in a deeply layered architecture. This models allow to efficiently reconstruct the hierarchical organisation of information, by representing complex concepts in terms of simpler elements.

DL algorithms are used to tackle nowadays classification problems, in the fields, e.g., of image [27] and voice [35] recognition, self-driving cars [26] or healthcare applications [66], becoming more and more complex. This requires to process huge datasets to train NNs on large amount of data before they are able to comply with a reasonable accuracy in prediction.

To enable DL algorithms to tackle increasingly complex problems, a brute-force solution, scaling up to the infrastructure level, is often applied [19]. The number of employed neurons grows in order to deal with an increasing amount of data, making the training process of the resulting network more and more computationally demanding. The increase in the required computational power needed to solve these problems has been faced in multiple ways. Various techniques aim to reduce the dimensionality of the problem and the number of parameters that should be tuned to fully characterize the architecture of the network, either by employing different approaches as Convolutional or Recurent Neural Networks or by imposing sparsity constraints on the space itself. This dimensionality reduction can be also achieved by introducing a structure in the dataset, by providing *a priori* information that come from any prior knowledge in the analysed field. All these approaches, even if they reach a certain level of efficiency in making the problem more easy to solve, cannot fully remove the intrinsic complexity of the addressed task.

The amount of computational power required for training DL applications, other than the need of high storage capabilities to deal with huge or complex datasets, remain a core issue in the field of Machine Learning. The habit of employing Graphic Processing Units (GPUs) as General Purpose parallel processors has been explored in the last years as a technique able to enhance the computational power. The resulting massive parallelism enables to solve, with a reasonable computational time and effort, increasingly complex tasks, as those arising in the fields of DL and Artificial Intelligence. The growth of the market related to GPU services, as a consequence, is expected to be considerably high in the next years, starting from over 700 million USD in 2019 and increasing with a compound annual rate of over 38% up to 2024 [40]. GPU acceleration, and especially the possibility of efficiently performing matrix multiplications in parallel, thanks to highly specialized linear algebra libraries, is particularly suited to DL training tasks, guaranteeing, with respect to CPU-based systems, a performance boost from 5 to 40x [13], [49]. Moreover, an additional gain in performance is ensured by the fact that Deep Neural Network models are often designed in order to be efficiently deployed on GPU-based systems, taking full advantage from their architecture. Despite those great benefits, the process of training DL applications remains a computationally intensive task. Moreover, GPU-based servers are characterized by considerably high costs (up to 500k USD if, for instance, NVIDIA DGX-2 is considered [21]). As a consequence, they remain often unaffordable for the general public, consisting also of small organizations with limited budget.
This growing demand and the issues related to accessibility of GPU-based architecture determined, in the last years, a consequent progression of Cloud services and solutions aiming to enhance the use of those resources in different contexts. The possibility of having access to GPU-based systems according to pay-to-go pricing models has contributed to the wide spread of those technologies applied to the solution of different issues, involving the DL training problems discussed before. Getting rid of the initial investments and the high costs required to maintain Data Centers over time, those technologies became more easily accessible to a wide range of organizations, that turned out to be able to take advantages from such architectures with reasonable financial burdens.
Despite the advantages coming from the Cloud Computing paradigm and the consequent possibility of having access to an ideally unlimited computational and storage power, the time unit costs of Virtual Machines (VMs) based on GPUs is still remarkably high, being 5-8x more expensive than those of VMs exploiting only CPUs [22]. As a consequence, the problem of determining an efficient scheduling for DL training jobs and other applications that should be deployed on those systems has still a great importance. Such an issue is the core of this thesis work. It is addressed both from the point of view of the Cloud end-users, who submit their applications to the Cloud and try to minimize the expected cost of their execution, and from the perspective of

Cloud Providers (CPs), who are in charge of meeting the clients' requirements, while maximizing the profit. The problem encompasses two main aspects: capacity allocation, concerning the placement of VMs on physical servers, and job scheduling, aiming to identify the most efficient way to share the available resources (VMs and GPUs) among concurrently running applications. The joint problem is addressed in the literature by exploiting different approaches and it represents a great challenge in terms of modeling and solvability. The complexity is even exacerbated, in the envisioned scenario, by the fact that capacity allocation and scheduling are analysed in an online setting, where multiple DL training jobs are submitted in a continuous fashion, so that no scheme can be detected in their arrival times or characteristics, particularly in terms of priority.

The main goal of the proposed mathematical formulation of the problem and the proposed algorithms is to design an optimal scheduling for the jobs, that allows to minimize the overall execution costs, while meeting the constraints related to system capacity and applications' deadlines. Resource sharing is designed in such a way that multiple tasks can be deployed on the same machine, with a variable number of dedicated GPUs. Moreover, jobs can be preempted, in order to be able to prioritize, if needed, subsequently arriving jobs with different characteristics in terms of execution times and expected deadlines.

Starting from an initial model presented in [41], the proposed work aims to enhance the scalability of the solution by exploiting a different Mixed Integer Linear Programming (MILP) formulation, applied to the aforementioned problem in a hierarchical fashion. Moreover, different heuristic approaches, inspired to greedy and local search techniques, have been designed and implemented to determine an efficient solution both in terms of computational time and accuracy.

The thesis is organized as follows: Chapter 2 presents the state of the art, by providing a general overview of the technological fields explored in this work and by providing a background on the optimization approaches adopted to design the proposed algorithms. Chapter 3 describes the joint capacity allocation and job scheduling problem as it is addressed in this work and recalls the formulation and the results obtained in [41], used as starting point for the subsequent analysis. Chapter 4 addresses the joint capacity allocation and job scheduling problem from the Cloud end-users' perspective, by presenting a novel MILP formulation and the different algorithms that have been designed in order to determine an effective and scalable approach. Chapter 5 modifies the perspective by presenting the same problem from the point of view of CPs, who are in charge of dealing with the requests submitted to very large Data Centers. Chapter 6 reports and discusses the results of an extensive experimental campaign aimed at evaluating the performance of the proposed approaches in a variety of scenarios of practical interest. Finally, Chapter 7 draws the conclusions of this thesis work and introduces some possible future reseach directions.

CHAPTER 7

# Conclusions and future work

The core objective of this thesis work has been the analysis of the joint capacity allocation and job scheduling for Deep Learning (DL) training jobs. This has been addressed both from the perspective of Cloud end-users (who try to minimize the expected execution costs of the application submitted to the Cloud) and from the perspective of Cloud Providers (who are in charge of meeting their customers' requirements, while maximizing the expected income).
The modeling and solution of this problem represent great challenges exacerbated, in the envisioned scenario, by the fact that the problem is setup in an online setting, where multiple DL training jobs are submitted in a continuous fashion, so that no scheme can be detected in their arrival times or characteristics, particularly in terms of priority.
The aim of this work was to provide, starting from an initial model presented in [41], alternative formulations and solution methods that allow to enhance the scalability of the initial approach. The complexity of the previous model and of the Monolithic approach that had been designed to solve it, indeed, allow to analyse systems composed by 40 nodes at most. Even in a context where the problem is solved from the point of view of Cloud end-users, this number is too small and is not representative of real instances, that can include hundreds of nodes. This issue is further exacerbated in the context of Data Center environments, that usually involve thousands of machines and have to deal with requests coming from multiple organization domains.
Two different Mixed Integer Linear Programming (MILP) formulations have been proposed, in Chapters 4 and 5, in order to describe the joint problem in the two scenarios. The main goal of these MILP models was to design an optimal scheduling for jobs, that would allow to minimize the overall execution costs, while meeting the constraints related to system capacity and applications' deadlines. Resource sharing has been designed in such a way that multiple tasks could be deployed on the same machine, with a variable

number of dedicated GPUs. Moreover, jobs could be preempted, in order to be able to prioritize, if needed, subsequently arriving jobs with different characteristics in terms of execution times and expected deadlines.
The first approach developed to solve the proposed model, discussed in Section 4.2, aimed to tackle the solution of the problem in a hierarchical fashion, i.e., to reduce the dimensionality by splitting the Monholitic MILP formulation in a set of smaller subproblems. Furthermore, three different heuristic methods, inspired to greedy and local search techniques, have been developed, as reported in Sections 4.3, 4.4 and 4.5, in order to further enhance the performance obtained with the Hierarchical approach, while maintaining the quality of the results in terms of scheduling costs.
An extensive experimental campaign has been performed, aimed at evaluating the performance of the proposed solutions in a variety of scenarios of practical interest. The results, extensively discussed in Chapter 6, highlight the effectiveness of the proposed approaches, both in terms of scalability and quality of the identified solutions. The comparison with first principle methods based on First-In-First-Out and Earliest Deadline First highlights significant cost savings, with an average gain between 50 and 80%, for all the proposed approaches. The analysis of the results obtained by the different methods in all the considered scenarios, as discussed in Section 6.7, allow to conclude that the models and approaches developed for this thesis work fully attain the goal of providing scalable techniques for the solution of the joint capacity allocation and job scheduling problem. The very good results obtained in terms of computational time, as well as the quality of the solutions, allow to exploit the proposed methods to solve large problem instances, involving hundreds of nodes, as those arising in practical scenarios.
Future developments of this work include, first of all, the implementation of all the proposed approaches, namely the Hierarchical Model, the Greedy, Randomized Greedy and Local Search algorithms, to the solution of the Data Center environments' problem in its heterogeneous version. This framework, indeed, by extending the space of available solutions, allows to fully explore the power of randomization and local search techniques.
Moreover, the Local Search algorithm will be extended to cope with a different objective function, aiming to enhance the stability of the proposed solutions by minimizing the difference between the finish times of jobs deployed on the same node. Since the Local Search step demonstrated to be very fast compared to the time required by the Randomized Greedy step, also its effectiveness to reduce the number of random iterations will be investigated. The randomized greedy construction introduces, indeed, a linear increase in complexity with respect to the pure greedy approach, while the overhead added by the local search phase has been demonstrated to be small in percentage. Therefore, the possibility of maintaining or improving the quality of the obtained solutions by performing a reduced number of random iterations in the construction phase, followed by a step of local search, can be explored as a future work.

Finally, also the possibility of exploiting the Greedy, Randomized Greedy and Local Search methods in a hierarchical fashion could be considered as an interesting topic for future developments.

CHAPTER 8

# Appendix

This appendix provides an overview of the two tools implemented for the solution of the Hierarchical Model described in Section 4.2 and the three heuristic approaches, namely, Greedy, Randomized Greedy and Local Search, described in Sections 4.3, 4.4 and 4.5. The general structure of the tools flow, that is the same for all the frameworks, is described in Section 8.1, while a more detailed description of the simulation and solution processes implemented for the different methods is provided in Sections 8.2 and 8.3.

## 8.1 General structure

The general structure of the process implemented to solve the problem presented in Chapter 4 is independent from the chosen approach and therefore is common to all the available methods. As illustrated in Figure 8.1, it consists of mainly three components, namely a data generator, a simulator and a solver. The first is, in particular, common to all the implemented tools, while the others have been differently specialized for the two cases of the Hierarchical approach and the heuristic methods.
The data generation process has been implemented in Python. Given a list of inputs concerning the characteristics of the required system, it allows to generate all data that are needed to run the simulation process described in the following. In particular, the parameters that can be provided, through a suitable configuration file, concern the number of nodes that will be available in the system, the total number of applications that are submitted along the simulation, denoted in the following as $J_c$, and the average inter-arrival time to be considered.
Starting from a pool of candidate applications, whose characteristics, in terms of execution times on all the available configurations, have been determined as described in Section 6.1, a Python code randomly selects $J_c$ jobs to form

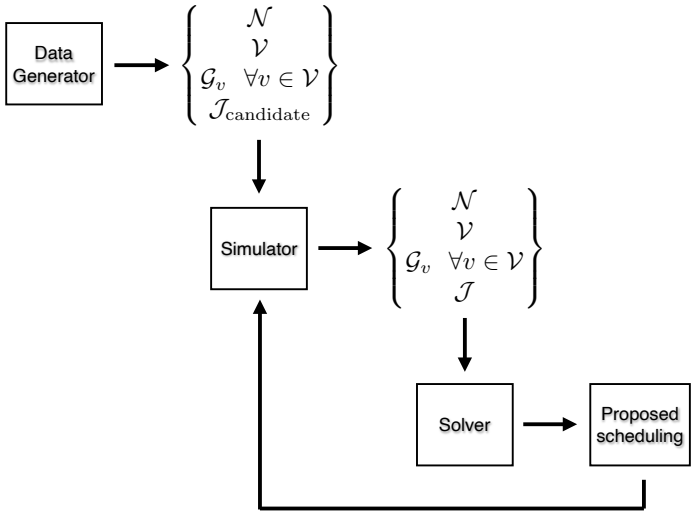


**Figure 8.1** – Global structure of the solution process

the set $\mathcal{J}_{\text{candidate}}$. All information related to submission times, deadlines and tardiness weights are computed in this phase as described in Section 6.1. The applications in $\mathcal{J}_{\text{candidate}}$ are going to be selected, during the simulation process, to form the queue $\mathcal{J}$ of submitted jobs. Furthermore, all the information related to the resources available in the system, namely the set $\mathcal{N}$ of nodes and the sets $\mathcal{V}$ and $\mathcal{G}_v$, for all $v \in \mathcal{V}$, representing, respectively, the catalog of available VMs and the GPUs that can be instantiated on each VM, are generated in this section and used in the following to solve the provided instance.

The simulation process is in charge of reproducing the behaviour of a real online system, that receives a series of requests and runs the existing solver to determine the best scheduling for all the submitted applications. The structure of the simulator is similar for the Hierarchical method and for the heuristic procedures, but it has been implemented separately in order to be better interfaced with the underneath solver.

The simulation process mainly consists of a time loop, whose iterations are leaded by the submission of new applications or the completion of their execution. In particular, the new jobs are extracted from $\mathcal{J}_{\text{candidate}}$ according to their submission time and are inserted in the queue $\mathcal{J}$. This queue, as well as the information concerning the available resources, is provided to the solver, that analyses the current instance in order to determine the best possible scheduling. This is then returned to the simulator, that computes the costs related to the jobs' execution and, if an application is completed, the possible penalties due to deadlines' violations. The process ends when all jobs have been completely executed.

A more detailed description of the simulation and the solution process is provided in the following sections.

## 8.2 Hierarchical Model

For what concerns the Hierarchical Model, the solution process that determines the optimal scheduling is performed, at each iteration, by Gurobi Optimizer [**?**]. It consists of a mathematical optimization solver that can be applied to Linear, Quadratic and Mixed Integer Programming. The interface between the simulator and the solver has been implemented by relying on Pyomo [**?**], a Python-based modeling language that allows to define a symbolic model and to generate the instance that has to be solved. The whole simulation process has been therefore implemented, in the case of the Hierarchical Model, as a Python library. It includes, in addition to the aforementioned procedures, also the initial splitting of the set $\mathcal{N}$ of available nodes and of the list of candidate jobs that has beed described in Section 4.2. This takes place at the beginning of the process, so that, if $K$ is the number of local controllers, the list $\mathcal{J}_{\text{candidate}}$ is split in $K$ different portions before entering in the time loop representing the simulation. This has been done in order to simulate a real system, such that all controllers run independently one from the others, by performing $K$ different simulations, whose results in terms of computed costs are gathered only at the end of the whole process.

## 8.3 Heuristic approaches

Both the simulator and the solver developed for the heuristic algorithms presented in Sections 4.3, 4.4 and 4.5 have been implemented in a C++ program. In particular, the three methods have been designed as subsequent specializations of a unique class, denoted as Heuristic, that can be used as basis to implement different heuristic procedures, involving also the first principle methods already mentioned in Chapter 6. As a consequence, they share the implementation of the simulator, as well as the maximal part of the implementation of the solver, whose three phases are described in Section 4.3. In particular, the Randomized Greedy method extends the solver by introducing an internal loop, that is in charge of performing the required number of iterations for the randomized construction. The Local Search class, in turn, inherits from the class representing the Randomized Greedy and adds a method implementing the local search at the end of the scheduling step. Greedy, Randomized Greedy and Local Search have been inserted in a factory, so that the choice of the method that has to be used to run the simulation can be performed at runtime by editing the same configuration file that is used to select the number of nodes and jobs to be considered in the system.

# Bibliography


[1] Amazon Web Services, http://aws.amazon.com/

[2] Google Docs web service, https://www.google.com/intl/en_US/docs/about/

[3] Google Inc., http://www.google.com/about/company/

[4] Google Sheets web service, https://www.google.com/intl/en_US/sheets/about/

[5] Microsoft Corporation, http://www.microsoft.com/

[6] Top500 list - top500 supercomputer sites (November 2019), https://www.top500.org/lists/2019/11/

[7] Amazon web service pricing list (2020), https://aws.amazon.com/ec2/pricing/on-demand/?nc1=h_ls

[8] Azure cloud services pricing list (2020), https://azure.microsoft.com/en-us/pricing/details/cloud-services/

[9] Pytorch machine learning framework (2020), https://pytorch.org

[10] Tensorflow machine learning platform (2020), https://www.tensorflow.org

[11] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I.J., Harp, A., Irving, G., Isard, M., Jia, Y., Józefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D.G., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P.A., Vanhoucke, V., Vasudevan, V., Viégas, F.B., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X.: Tensorflow: Large-scale machine learning on heterogeneous distributed systems. CoRR **abs/1603.04467** (2016), http://arxiv.org/abs/1603.04467

[12] Amaral, M., Polo, J., Carrera, D., Seelam, S., Steinder, M.: Topology-aware gpu scheduling for learning workloads in cloud environments. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. SC '17, Association for Computing Machinery, New York, NY, USA (2017). https://doi.org/10.1145/3126908.3126933

[13] Bahrampour, S., Ramakrishnan, N., Schott, L., Shah, M.: Comparative study of deep learning software frameworks. arXiv (2015)

[14] Bao, Y., Peng, Y., Wu, C.: Deep learning-based job placement in distributed machine learning clusters. In: IEEE INFOCOM 2019 - IEEE Conference on Computer Communications. pp. 505–513 (April 2019). https://doi.org/10.1109/INFOCOM.2019.8737460

[15] Bölöni, L., Turgut, D.: Value of information based scheduling of cloud computing resources. Future Generation Computer Systems **71**, 212 – 220 (2017). https://doi.org/10.1016/j.future.2016.10.024, http://www.sciencedirect.com/science/article/pii/S0167739X16304472

[16] Briscoe, G., Marinos, A.: Digital ecosystems in the clouds: Towards community cloud computing. In: 2009 3rd IEEE International Conference on Digital Ecosystems and Technologies. pp. 103–108 (2009)

[17] Cai, Z., Li, X., Ruiz, R., Li, Q.: A delay-based dynamic scheduling algorithm for bag-of-task workflows with stochastic task execution times in clouds. Future Generation Computer Systems **71**, 57 – 72 (2017). https://doi.org/10.1016/j.future.2017.01.020, http://www.sciencedirect.com/science/article/pii/S0167739X17300870

[18] Cao, M., Jia, W., Li, S., Li, Y., Zheng, L., Liu, X.: Gpu-accelerated feature tracking for 3d reconstruction. OLT (Elsevier) **110**, 165–175 (2019)

[19] Chen, X., Lin, X.: Big data deep learning: Challenges and perspectives. IEEE Access **2**, 514–525 (2014)

[20] Coffman, E.G., Garey, M.R., Johnson, D.S.: Approximation Algorithms for Bin-Packing — An Updated Survey, pp. 49–106. Springer Vienna, Vienna (1984). https://doi.org/10.1007/978-3-7091-4338-4-3

[21] high performance computing, T.: Nvidia tesla gpu servers (gpx). [Online]. Available: https://www.thinkmate.com/systems/servers/gpx (visited on 26/01/2020)

[22] Corporation, N.: Nvidia virtual gpu technology. [Online]. Available: https://www.nvidia.com/en-us/design-visualization/technologies/virtual-gpu/ (visited on 26/01/2020)

[23] Correia, I., Gouveia, L., da Gama, F.S.: Solving the variable size bin packing problem with discretized formulations. Computers And Operations Research **35**(6), 2103 – 2113 (2008). https://doi.org/10.1016/j.cor.2006.10.014, http://www.sciencedirect.com/science/article/pii/S0305054806002747

[24] Cung, V.D., Martins, S., Ribeiro, C., Roucairol, C.: Strategies for the parallel implementation of metaheuristics (01 2001). https://doi.org/10.1007/978-1-4615-1507-4_13

[25] Delorme, M., Iori, M., Martello, S.: Bin packing and cutting stock problems: Mathematical models and exact algorithms. European Journal of Operational Research **255**(1), 1 – 20 (2016). https://doi.org/10.1016/j.ejor.2016.04.030, http://www.sciencedirect.com/science/article/pii/S0377221716302491

[26] Do, T., Duong, M., Dang, Q., Le, M.: Real-time self-driving car navigation using deep neural network. In: 2018 4th International Conference on Green Technology and Sustainable Development (GTSD). pp. 7–12 (2018)

[27] Dong, Y., Liang, G.: Research and discussion on image recognition and classification algorithm based on deep learning. In: 2019 International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI). pp. 274–278 (2019)

[28] Filho], M.C.S., Monteiro, C.C., InÃ¡cio, P.R., Freire, M.M.: Approaches for optimizing virtual machine placement and migration in cloud environments: A survey. Journal of Parallel and Distributed Computing **111**, 222 – 250 (2018). https://doi.org/10.1016/j.jpdc.2017.08.010, http://www.sciencedirect.com/science/article/pii/S074373151730240X

[29] Flexera: https://www.flexera.com/about-us/press-center/rightscale-2019-state-of-the-cloud-report-from-flexera-identifies-cloud-adoption-trer html

[30] Gao, W., Friedrich, T., Neumann, F., Hercher, C.: Randomized greedy algorithms for covering problems. pp. 309–315 (07 2018). https://doi.org/10.1145/3205455.3205542

[31] Gianniti, E., Zhang, L.: Performance prediction of gpu-based deep learning applications. pp. 167–170 (09 2018). https://doi.org/10.1109/CAHPC.2018.8645908

[32] Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016), http://www.deeplearningbook.org

[33] Gopalakrishnan, P., B., U.M.: Research on enterprise public and private cloud service **8**, 1453–1459 (04 2019). https://doi.org/10.35940/ijitee.f1296.0486s419

[34] H.A., E., CL., S.: Heuristic Algorithms. Springer, Berlin, Heidelberg (2000). https://doi.org/10.1007/978-3-662-04197-0_11

[35] Hannun, A., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., Prenger, R., Satheesh, S., Sengupta, S., Coates, A., Ng, A.Y.: Deep speech: Scaling up end-to-end speech recognition (2014)

[36] Haouari, M., Serairi, M.: Heuristics for the variable sized bin-packing problem. Computers And Operations Research **36**(10), 2877 – 2884 (2009). https://doi.org/10.1016/j.cor.2008.12.016, http://www.sciencedirect.com/science/article/pii/S0305054808002748

[37] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition (2015)

[38] IDC: https://www.idc.com/getdoc.jsp?containerId=prUS45340719#targetText=With%20a%20five%2Dyear%20compound,nearly%20%24500%20billion%20in%202023.&targetText=Infrastructure%20as%20a%20Service%20(IaaS)%20will%20be%20the%20second%20largest,as%20a%20Service%20(PaaS).

[39] Ilager, S., Muralidhar, R., Rammohanrao, K., Buyya, R.: A data-driven frequency scaling approach for deadline-aware energy efficient scheduling on graphics processing units (gpus). Tech. rep., Cloud Computing and Distributed Systems (CLOUDS) Laboratory; School of Computing and Information Systems, The University of Melbourne, Australia, http://www.buyya.com/papers/DDFreqScaleGPU.pdf

[40] Insights, G.M.: Gpu as a service market size by product. [Online]. Available: https://www.gminsights.com/industry-analysis/gpu-as-a-service-market (visited on 26/01/2020)

[41] Jahani, A., Lattuada, M., Ciavotta, M., Ardagna, D., Amaldi, E., Zhang, L.: Optimizing on-demand gpus in the cloud for deep learning applications training. In: 2019 4th International Conference on Computing, Communications and Security (ICCCS). pp. 1–8. IEEE (Oct 2019). https://doi.org/10.1109/CCCS.2019.8888151

[42] Kang, D., Jun, T.J., Kim, D., Kim, J., Kim, D.: Convgpu: Gpu management middleware in container based virtualized environment. In: 2017 IEEE International Conference on Cluster Computing (CLUSTER). pp. 301–309 (Sep 2017). https://doi.org/10.1109/CLUSTER.2017.17

[43] Kang, J., Park, S.: Algorithms for the variable sized bin packing problem. European Journal of Operational Research **147**(2), 365 – 372 (2003). https://doi.org/10.1016/S0377-2217(02)00247-3, http://www.sciencedirect.com/science/article/pii/S0377221702002473

[44] Kang, Y., Joo, W., Lee, S., Shin, D.: Priority-driven spatial resource sharing scheduling for embedded graphics processing units. Journal of Systems Architecture **76**, 17 – 27 (2017). https://doi.org/10.1016/j.sysarc.2017.04.002, http://www.sciencedirect.com/science/article/pii/S138376211730190X

[45] Kaur, A.: Cloud computing: An asset or a drawback (a survey) **1**, 2455–4227 (06 2016)

[46] Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Pereira, F., Burges, C.J.C., Bottou, L., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 25, pp. 1097–1105. Curran Associates, Inc. (2012), http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[47] Lecun, Y., Kavukcuoglu, K., Farabet, C.: Convolutional networks and applications in vision. pp. 253–256 (05 2010). https://doi.org/10.1109/ISCAS.2010.5537907

[48] Li, X., Qian, L., Ruiz, R.: Cloud workflow scheduling with deadlines and time slot availability. IEEE Transactions on Services Computing **11**(2), 329–340 (March 2018). https://doi.org/10.1109/TSC.2016.2518187

[49] Madougou, S., Varbanescu, A., de Laat, C., van Nieuwpoort, R.: The landscape of gpgpu performance modeling tools. Parallel Computing **56**, 18 – 33 (2016). https://doi.org/10.1016/j.parco.2016.04.002, http://www.sciencedirect.com/science/article/pii/S0167819116300114

[50] Mann, Z.A.: Allocation of virtual machines in cloud data centers - a survey of problem models and optimization algorithms. ACM Comput. Surv. **48**(1) (Aug 2015). https://doi.org/10.1145/2797211

[51] Mell, P., Grance, T.: The NIST definition of Cloud computing. Tech. rep. (Jul 2009), http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

[52] Mitchell, T.: Machine Learning. McGraw-Hill International Editions, McGraw-Hill (1997), https://books.google.it/books?id=EoYBngEACAAJ

[53] Peng, Y., Bao, Y., Chen, Y., Wu, C., Guo, C.: Optimus: An efficient dynamic resource scheduler for deep learning clusters. In: Proceedings of the Thirteenth EuroSys Conference. EuroSys '18, Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3190508.3190517

[54] Peng, Y., Bao, Y., Chen, Y., Wu, C., Meng, C., Lin, W.: Dl2: A deep learning-driven scheduler for deep learning clusters. arXiv (2019)

[55] Reano, C., Silla, F., Nikolopoulos, D., Varghese, B.: Intra-node memory safe gpu co-scheduling. IEEE Transactions on Parallel and Distributed Systems **PP** (12 2017). https://doi.org/10.1109/TPDS.2017.2784428

[56] Resende, M., Ribeiro, C.: Grasp with path-relinking: Recent advances and applications. Operations Research/ Computer Science Interfaces Series **32** (01 2005). https://doi.org/10.1007/0-387-25383-1_2

[57] Resende, M., Ribeiro, C.: Greedy Randomized Adaptive Search Procedures: Advances, Hybridizations, and Applications, vol. 146, pp. 283–319 (09 2010). https://doi.org/10.1007/978-1-4419-1665-5_10

[58] Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Nature **323**(6088), 533–536 (1986). https://doi.org/10.1038/323533a0

[59] Shao, J., Ma, J., Li, Y., An, B., Cao, D.: Gpu scheduling for short tasks in private cloud. In: 2019 IEEE International Conference on Service-Oriented System Engineering (SOSE). pp. 215–2155 (2019). https://doi.org/10.1109/SOSE.2019.00037

[60] Sheikhalishahi, M., Wallace, R.M., Grandinetti, L., Vazquez-Poletti, J.L., Guerriero, F.: A multi-dimensional job scheduling. Future Generation Computer Systems **54** (2016). https://doi.org/10.1016/j.future.2015.03.014

[61] Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition (2014)

[62] Winkler, V.J.: Chapter 7 - security criteria: Building an internal cloud. In: Winkler, V.J. (ed.) Securing the Cloud, pp. 187 – 210. Syngress, Boston (2011). https://doi.org/10.1016/B978-1-59749-592-9.00007-5, http://www.sciencedirect.com/science/article/pii/B9781597495929000075

[63] Wong, A.K.L., Goscinski, A.M.: Evaluating the easy-backfill job scheduling of static workloads on clusters. In: 2007 IEEE International Conference on Cluster Computing. pp. 64–73 (Sep 2007). https://doi.org/10.1109/CLUSTR.2007.4629218

[64] Xiao, W., Bhardwaj, R., Ramjee, R., Sivathanu, M., Kwatra, N., Han, Z., Patel, P., Peng, X., Zhao, H., Zhang, Q., Yang, F., Zhou, L.: Gandiva: Introspective cluster scheduling for deep learning. In: 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18). pp. 595–610. USENIX Association, Carlsbad, CA (Oct 2018), https://www.usenix.org/conference/osdi18/presentation/xiao

[65] Youseff, L., Butrico, M., Silva, D.: Toward a unified ontology of cloud computing. pp. 1 – 10 (12 2008). https://doi.org/10.1109/GCE.2008.4738443

[66] Yu, Y., Li, M., Liu, L., Li, Y., Wang, J.: Clinical big data and deep learning: Applications, challenges, and future outlooks. Big Data Mining and Analytics **2**(4), 288–305 (2019)

[67] Zhang, H., Stafman, L., Or, A., Freedman, M.J.: Slaq: Quality-driven scheduling for distributed machine learning. In: Proceedings of the 2017 Symposium on Cloud Computing. pp. 390–404. SoCC '17, Association for Computing Machinery, New York, NY, USA (2017)

[68] Zhang, Q., Cheng, L., Boutaba, R.: Cloud computing: State-of-the-art and research challenges. J. Internet Services and Applications **1**(1), 7–18 (2010). https://doi.org/10.1007/s13174-010-0007-6

[69] Zhao, H., Wang, J., Liu, F., Wang, Q., Zhang, W., Zheng, Q.: Power-aware and performance-guaranteed virtual machine placement in the cloud. IEEE Transactions on Parallel and Distributed Systems **29**(6), 1385–1400 (2018)

[70] Zhu, J., Li, X., Ruiz, R., Xu, X.: Scheduling stochastic multi-stage jobs to elastic hybrid cloud resources. IEEE Transactions on Parallel and Distributed Systems **29**(6), 1401–1415 (June 2018). https://doi.org/10.1109/TPDS.2018.2793254

[71] Zou, C., Deng, H., Qiu, Q.: Design and implementation of hybrid cloud computing architecture based on cloud bus. In: 2013 IEEE 9th International Conference on Mobile Ad-hoc and Sensor Networks. pp. 289–293 (2013)