Bulletin of the *Transilvania* University of Braşov • Vol 13(62), No. 2 - 2020
Series III: Mathematics, Informatics, Physics, 705-714
https://doi.org/10.31926/but.mif.2020.13.62.2.26

HIGH PERFORMANCE COMPUTING FOR MACHINE LEARNING

Árpád KERESTÉLY[*,1]

Abstract

Efficient High Performance Computing for Machine Learning has become a necessity in the past few years. Data is growing exponentially in domains like healthcare, government, economics and with the development of IoT, smartphones and gadgets. This big volume of data, needs a storage space which no traditional computing system can offer, and needs to be fed to Machine Learning algorithms so useful information can be extracted out of it. The larger the dataset that is fed to a Machine Learning algorithm the more precise the results will be, but also the time to compute those results will increase. Thus, the need for Efficient High Performance computing in the aid of faster and better Machine Learning algorithms. This paper aims to unveil how one benefits from another, what research has achieved so far and where is it heading.

2000 *Mathematics Subject Classification:* 68T01, 68T02.

Key words: distributed, data parallelism, model parallelism, Hadoop, Spark, machine learning, data science, data mining, high performance computing.

1 Introduction

High Performance Computing and Machine Learning are two different topics. They developed analogous of each other driven by different necessities. Still the two joined at some point and now it is hard to think of one, without the other lurking in the background. Before diving into detail, let's look into some basic concepts.

[1*] *Corresponding author*, Faculty of Mathematics and Informatics, *Transilvania* University of Braşov, Romania, e-mail: arpad.kerestely@unitbv.ro

1.1 High Performance Computing

"High Performance Computing (HPC) most generally refers to the practice of aggregating computing power in a way that delivers much higher performance than one could get out of a typical desktop computer or workstation in order to solve large problems in science, engineering, or business." [23] Aggregating computing power can be done to result in:

- single *supercomputer*: composed of powerful central processing units (CPUs), graphical processing units (GPUs), big memory (RAM), huge (and fast) storage space; unaffordable to most companies (not to mention researchers)

- *cluster of computers*: many high end but affordable computers - called *nodes* - interconnected through a network, that act as a single computer to the end user

The most commonly used are the clusters of computers, as a cluster can easily be extended with additional nodes (computers) to quickly gain more computing power or storage space. A typical node in a cluster has about 1 to 4 CPUs each having 2, 4 or 8 cores, which can be used to compute more tasks locally or to speed up the execution of one task. A cluster can have from 2 nodes, used by small companies, to 256 nodes, used by medium and big companies. Data-centers typically have thousands of nodes in a cluster, but also have many clusters, depending on the tasks at hand or the purpose of the data-center. The nodes are usually connected and communicate through Ethernet connection to act as one computer. By the nature of this architecture, nodes can join a cluster even if they are not physically in the same location, although that is not desired, as there could be latency issues. Another advantage of having a cluster on nodes to act as supercomputer is that it can be remotely or even automatically scaled on demand, which is not possible with single supercomputers.

High performance systems can normally reduce the execution time of an algorithm if the algorithm itself can run in a parallel fashion, meaning the problem can be decomposed into multiple sub problems, the results of which combined will give the solution to the initial problem. While this is generally true, it is not always enough for an algorithm to run faster: memory access, disk I/O and inter-node communications should be taken into account when decomposing a problem.

High performance computing also refers to using specialized dedicated hardware for certain problems. An example is the usage of the GPU to render high demanding 3D scenes, animations and movies which would normally run very slow on the general purpose CPU. Lately there has been an opening to using the GPU for more than just graphical processing. Thus, for example, Nvidia developed PhysX to run Newtonian physics simulations, or CUDA to enable general purpose processing on the GPU, making the GPU termed as a GPGPU (General Purpose Graphical Processing Unit). Other companies like Intel or AMD developed their own technologies that are similar to those that Nvidia offers.

1.2 The intersection of Machine Learning and High Performance Computing

A strong bond exists between Machine Learning algorithms and High Performance Computing, even if at first glance it is not that obvious. Looking at the evolution of Machine Learning algorithms, we could say that some key components existed long before the first computers were "born". When computing stations began spreading, hope had reborn in Machine Learning algorithms, but soon vanished as the required computing power to run them, was unimaginable. This is how "Artificial Neural Networks", which were first published in the '80s, gained significant popularity only after 25 years, when the computational power had caught up so something significant could be made with them, and in reasonable amount of time. So why does Machine Learning need High Performance Computing?

Generally speaking, most Machine Learning algorithms need to "train" (we can call this fit, or find the best parameters, or compute some distance) before being able to predict (or generalize) the results of unseen input. Training takes time, for some algorithms, it takes more than for others. One fact is proven: the more the training data (or training iterations), the slower the train session will be. Another fact is proven too: that the more training iterations and data an algorithm gets the more chance it has to give better results. Putting these together, one can easily guess that often the performance of a learning model is sacrificed if favor of reduced training time, by reducing the amount of time an algorithm will train.

Another thing to have in mind is that most Machine Learning algorithms have some hyper-parameters that need to be tuned empirically so that the algorithms can yield the best results for the problem at hand. This in turn requires rerunning the training process several times, until the best hyper-parameters are found. This whole process is very long and sometimes ends prematurely, with not exploring the whole training set and hyper-parameters domain, thus not achieving the best possible results the algorithms could have had.

High Performance Computing can push the Machine Learning algorithms to new limits, by aiding in this costly process of training and beyond that. But there's a catch. Today's high performance systems rely heavily on the fact that algorithms can run in parallel. This said, Machine Learning can benefit from high performance computing only if its algorithms can be decomposed into parallelizable subroutines. Sometimes this step is obvious as in the case of finding the best hyper-parameters, but at other times it requires architectural redesigns of an algorithm.

In the following sections, this paper aims to provide a survey of the existing work that has been done in both High Performance Computing and Machine Learning, to provide personal recommendations and best practices on when and how to use High Performance Computing and Machine Learning, and finally, in the conclusion section, to wrap up and provide a forecast on where the High Performance Computing and Machine Learning might end up next.

2 Literature Overview

In the digitalized era we live in, large amounts of data are generated by the healthcare [20], government and economic systems. Some of the notable sources of data are the "record keeping, compliance and patient related data, physician notes, Lab reports, X-Ray reports, case history, diet regime, list of doctors and nurses in a particular hospital, national health register data, medicine and surgical instruments expiry date identification based on RFID data"[9]. These data help by providing "patient centric services, detecting spreading diseases earlier, monitoring the hospital's quality and improving the treatment methods" [9]. "A Survey Of Big Data Analytics in Healthcare and Government" [9] suggests a Big Data Ecosystem architecture that receives data streams through "Apache Flume" [2] and can import data from structured datastores such as relational databases with the usage of "Apache Sqoop" [7]. The data acquired this way is stored in a "Hadoop Distributed File System (HDFS)" which is a module of the bigger "Apache Hadoop" [3]. Analyzing and processing the data is done, using machine learning algorithms, in a parallel manner using "Map-Reduce" (which is again part of Apache Hadoop) and "Apache Hive" [5] which was formerly part of Hadoop, but it is now a standalone project and it is used to manage data using SQL. For storing multi-structured data, "Apache HBase" [4] is used, which is similar to Google's "Bigtable" [11] but is built on top of HDFS. "Apache Storm" [8] helps in processing streaming data in real time. To produce reports and gain a better knowledge of data, "Intellicius" and "Hunk" are used.

On the early days of Hadoop, the authors of "MapReduce: Distributed Computing for Machine Learning" [17] tested different types of machine learning algorithms with the Map-Reduce framework. They defined three categories of machine learning algorithms: "Single-pass Learning", "Iterative Learning" and "Query-based Learning with Distance Metrics" which behave slightly different on the Map-Reduce framework, with the first being the fastest and the last being the slowest. The execution time being proportional to the amount of queries made to the data available on the HDFS.

"Large-Scale Machine Learning based on Functional Networks for Biomedical Big Data with High Performance Computing Platforms" [15] outlines that Map-Reduce built on top of HDFS has some drawbacks, which are crucial when working with machine learning algorithms. One of the major drawbacks is that Map-Reduce architecture is designed to reload data from disk at each Map-Reduce processing function. Thus, it relies heavily on the disk I/O speed which is usually very slow compared to the speed of in-memory operations. Machine learning algorithms can't be efficient using Map-Reduce in its native form. "Spark" [6] overcomes this limitation by reducing the disk I/O operations and offering an in-memory solution, while keeping the fault tolerant behavior of Map-Reduce. The speed gain is claimed to be 100x the speed of Map-Reduce (see figure 1). Spark can work with HDFS as well as with other file systems and can be accessed through Java, Scala, Python, R, and SQL. "Offers over 80 high-level operators that make it easy to build parallel apps. And you can use it *interactively* from the Scala,

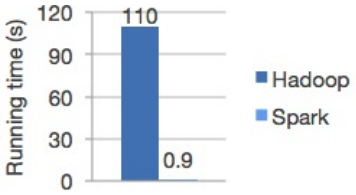

Figure 1: "Logistic regression in Hadoop and Spark" [6]

Python, R, and SQL shells." [6]. The authors of the paper [15] demonstrate the usage of "Functional Networks" [10] on a Spark system.

Recent work in the field of machine learning focused heavily on two types of algorithms: Convolutional Neural Networks (with all its flavors) and Deep Neural Networks. Because of the heavy processing and high memory footprint, these algorithms can't benefit from a framework like Spark, as much of the time would be wasted exchanging information between the computing nodes. To achieve faster communication speed between the memory and the computing unit, these algorithms have been moved to the Graphical Processing Unit (GPU). The result was an increase in speed from 10 to nearly 60 percent, compared to a run on CPU (using Singe Instruction Multiple Data (SIMD)) [12]. While GPUs are very powerful at parallel computing - mostly because they are composed of many small processing units that reside, physically, close to the GPUs dedicated memory - they have their limitations. The biggest limitation of the GPUs is their dedicated memory limit. While this limitation can be overcome by using RAM, research has shown that it comes with a significant performance penalty [12]. The trend in the aforementioned machine learning algorithms is to grow in size by pulling in more hidden layers, more neurons. As this has proven to help the networks generalize better, it is also a barrier that prevents training and running them on GPUs, because of the memory constraints. The authors of "DaDianNao: A machine-learning supercomputer" [12] suggest and implement dedicated hardware for CNNs and DNNs that overcomes the memory limitations, by providing a cluster of dedicated memories, thus opening the way for bigger, sophisticated networks. By being dedicated hardware for these types of algorithms, specific operations are also built in the hardware which boost the overall performance (as seen in Figure 2).

While these results are astonishing, the machine learning researchers community can't use them right away without having the hardware.

"Deep learning with COTS HPC systems" [13] offers a more accessible solution to the same problem, namely running deep neural networks with over 1 billion parameters. To be able to feel the magnitude of 1 billion parameters, let's assume that each parameter is a double which occupies 8 bytes, thus the size of 1 billion parameters is roughly 8 GB. With common end-user GPUs having mostly 4 GB of dedicated memory, the limitation becomes clear. Even in the event of having a GPU with 8 GB of memory the limitation is still there; we can't use a network of more than 1 billion parameters. One possibility would be to use a

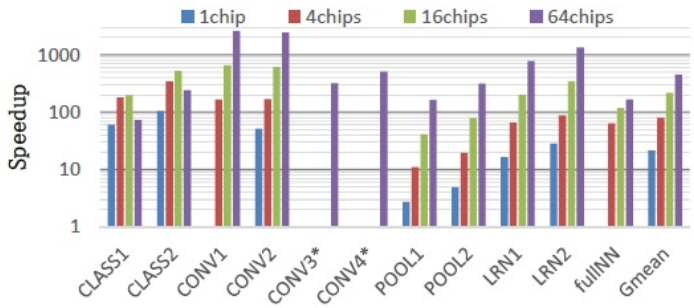

Figure 2: "Speedup w.r.t. the GPU baseline (inference)." [6] The X axis represents the different types of layers. # of chips represents the amount of chips clustered together to form the final hardware piece.

distributed computing infrastructure such as "DistBelief" [14], which trained a neural network using 16000 CPU cores (in 1000 machines) in a few days. Sadly, this kind of infrastructure is not available to most researchers. The solutions offered by [13] can make the same network train on only 3 machines, each having 4 NVIDIA GTX680 GPUs, in less time. The authors also show that in a few days they can train a network of 11 billion parameters using 16 GPU equipped machines. While the approach of having GPUs connected on a network seems intuitive, the overhead created by transferring knowledge through Ethernet connection between GPUs prevented significant improvements from being done. The authors of [13] succeeded in this chapter by using an FDR Infiniband connection between the machines, which can send messages in microseconds compared to Ethernet connection that needs a few milliseconds. "DistBelief" has evolved since it was first published, into the newer "Tensorflow" presented in [1]. This new framework can be used to run Machine Learning algorithms on a cluster of dedicated computers, but also on mobile devices, using their CPUs or GPUs, thus unifying the computation power under a single hood while providing a friendly API.

Although extra hardware can help a Machine Learning algorithm train and run faster, sometimes that is not enough. As it can be seen in Figure 3 from "Imagenet classification with deep convolutional neural networks" [21], but also in [13], specific architectural changes are needed sometimes so that an algorithm may run in parallel.

3 Personal recommendations

From the many possibilities that are out there, one may be confused as to what would be the best solution to go on with. The short answer is that there is no best solution. Most factors depend on the problem at hand. Nevertheless, this section will try to give some personal recommendations as to where to start.

The first and most important factor when deciding on the Machine Learn-

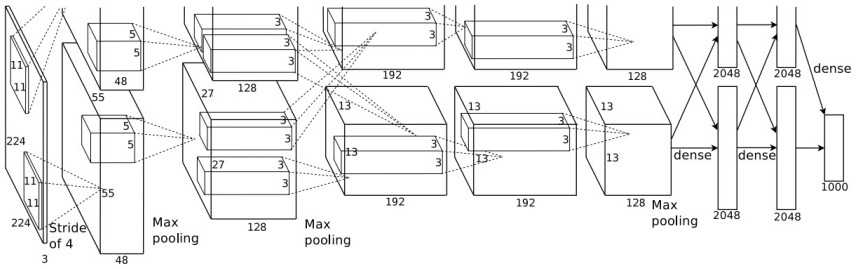

Figure 3: A deep convolutional neural network redesigned to work in parallel, on two GPUs (forced by memory constraints). Figure from [21]

ing algorithm and the High Performance Computing system is the nature of the dataset that is used. Important characteristics of a dataset are its features (attributes) and size. The features of a dataset can be: the pixels of an image, a timeseries, a text or the clinical data of a patient in the form of a table. The feature type determines the ML algorithm to use. For image classification some form of CNN should be used, timeseries could be handled by some form of curve fitting or RNN, for text data the NLP algorithms are the best suited, and so on. Regarding the size of the dataset there is one rule: if it fits into memory (RAM or VRAM), than one should go for a library that supports running ML algorithms on one computer in multi-threaded mode. If the dataset does not fit in memory, one should look for a distributed solution. There is also the third, the border case, when the data does not fit in memory, but the algorithm can still compute the results in memory; while this case can have it's benefits, it also has many shades which won't be covered in this paper. When searching for a good algorithm regardless of the size of data, one should start experimenting on a subset of the data.

When we think about distributed systems, one of the first options to consider is the Apache catalog. By setting up some modules on available hardware, or by renting servers that offer an already set up environment. It is important to note that the chosen ML algorithm influences the choice of the distributed system, or vice-versa. For example in Apache Spark's MLlib there is no existing implementation for K-Nearest Neighbors (KNN) or for Convolutional Neural Network (CNN) algorithms (current version: 3.0.0.). Looking deeper down in Spark's MLlib, we can observe that for ANNs, the *Adam* solver is not available to choose. A recommendation when choosing a library, be it distributed or not, would be to do a quick research beforehand on: available ML algorithms, exposed hyperparameters and the possibility of extending the library.

The preferred programming language can also narrow down the possible choice of Machine Learning algorithms and HPC system combinations. For example, to work and extend algorithms in Weka, one would need to know Java; to work natively with Spark, one should know Scala. For Python users there is good news though: most of the libraries offer an API to work directly in Python; this comes

from the fact that the Machine Learning community's preferred programming language is Python. The recommendation on this topic would obviously be the following: learn Python!

Other factors exist also but are more specific to a chosen direction. For example the renting costs of a server, or the availability of GPUs on the hardware at hand, or what operating system to choose when setting up a Hadoop-Spark cluster. Considering these, part of the research should always focus on what and how others did in the same situations.

4 Conclusions

The fact that High Performance Computing and Machine Learning coexist is certain, from the high number of papers that are available on this topic. More important is that Machine Learning is fueled by High Performance Computing, thus reaching new heights every now and then. Machine Learning, on the other hand, creates new opportunities for HPC practitioners to come up with new ideas and solutions, thus pushing the boundaries of what we perceive as high performance systems.

As a Machine Learning researcher, it is important to know that a wide range of solutions exist to make any algorithm run more efficiently. For example, using Hadoop to connect multiple computers to act as one brings the advantage of having access to a variety of smaller components with which to extend the functionality of a big data, machine learning application. Using the GPU's power to tame some of the most challenging machine learning algorithms can be a solution; but maybe connecting more GPUs or even having a cluster of machines, each having a couple of GPUs can be a solution too.

It is important also to know the data that needs to be processed, because structured and unstructured data are stored and processed differently, nevertheless some algorithms require and use both to compute their results. Thus, having some background in the big data domain can help overcome some difficulties.

Where will the High Performance Computing and Machine Learning go is not clear, but certainly they will always go hand in hand, as Machine Learning algorithms are not easy tasks for a computing system and where Machine Learning exists, there will most likely be a need for High Performance Computing. And who knows, some day, analogous of how the Graphical Processing Unit (GPU) came to exist in our personal computers, extrapolating the research done by [12], we could be seeing specialized hardware for machine learning tasks in our personal computers in the coming 10-15 years.

References

[1] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M. and Kudlur, M., 2016. *Tensorflow:*

A system for large-scale machine learning. In 12th USENIX symposium on operating systems design and implementation (OSDI 16) (pp. 265-283).

[2] *Apache Flume*. URL: https://flume.apache.org/

[3] *Apache Hadoop*. URL: http://hadoop.apache.org/

[4] *Apache HBase*. URL: https://hbase.apache.org/

[5] *Apache Hive*. URL: https://hive.apache.org/

[6] *Apache Spark*. URL: https://spark.apache.org/

[7] *Apache Sqoop*. URL: https://sqoop.apache.org/

[8] *Apache Storm*. URL: https://hortonworks.com/apache/storm/

[9] Archenaa, J. and Anita, E.M., 2015. *A survey of big data analytics in healthcare and government.* Procedia Computer Science, 50, pp.408-413.

[10] Castillo, E. and Hadi, A.S., 2014. *Functional networks.* Wiley StatsRef: Statistics Reference Online.

[11] Chang, F., Dean, J., Ghemawat, S., Hsieh, W.C., Wallach, D.A., Burrows, M., Chandra, T., Fikes, A. and Gruber, R.E., 2008. *Bigtable: A distributed storage system for structured data.* ACM Transactions on Computer Systems (TOCS), 26(2), pp.1-26.

[12] Chen, Y., Luo, T., Liu, S., Zhang, S., He, L., Wang, J., Li, L., Chen, T., Xu, Z., Sun, N. and Temam, O., 2014, December. *Dadiannao: A machine-learning supercomputer.* In 2014 47th Annual IEEE/ACM International Symposium on Microarchitecture (pp. 609-622). IEEE.

[13] Coates, A., Huval, B., Wang, T., Wu, D., Catanzaro, B. and Andrew, N., 2013, February. *Deep learning with COTS HPC systems.* In International conference on machine learning (pp. 1337-1345).

[14] Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Mao, M., Ranzato, M.A., Senior, A., Tucker, P., Yang, K. and Le, Q., 2012. *Large scale distributed deep networks.* Advances in neural information processing systems, 25, pp.1223-1231.

[15] Elsebakhi, E., Lee, F., Schendel, E., Haque, A., Kathireason, N., Pathare, T., Syed, N. and Al-Ali, R., 2015. *Large-scale machine learning based on functional networks for biomedical big data with high performance computing platforms.* Journal of Computational Science, 11, pp.69-81.

[16] Freund, K., 2017. *What's Hot At SC17: The Synthesis Of Machine Learning & HPC.* Forbes. URL: https://www.forbes.com/sites/moorinsights/2017/11/14/whats-hot-at-sc17-the-synthesis-of-machine-learning-hpc/

[17] Gillick, D., Faria, A. and DeNero, J., 2006. *Mapreduce: Distributed computing for machine learning.* Berkley, Dec, 18.

[18] *Introduction to High-Performance Machine learning @SURFsara.* URL: https://events.prace-ri.eu/event/693/attachments/626/. 2018.

[19] Kadupitige, K., 2017. *Intersection of HPC and Machine Learning.* Digital Science Center.

[20] Kerestely, Á., Sasu, L.M. and Tăbîrcă, M.S., 2018. *Machine Learning in Healthcare: An Overview.* Bulletin of the Transilvania University of Brasov. Mathematics, Informatics, Physics. Series III, 11(2), pp.273-278.

[21] Krizhevsky, A., Sutskever, I. and Hinton, G.E., 2017. *Imagenet classification with deep convolutional neural networks.* Communications of the ACM, 60(6), pp.84-90.

[22] Vose, A., 2017. *The Intersection of Machine Learning and High-Performance Computing.* URL: https://www.cray.com/blog/intersection-machine-learning-high-performance-computing/.

[23] *What is high performance computing?.* URL: https://insidehpc.com/hpc-basic-training/what-is-hpc.