Open and low consumption platform for analysis of distributed data

J Y Bolaño-Gonzalez1,2, and G A Serna-Lopez1,2
1 Grupo de Investigación en Gerencia y Aplicación de la Ciencia y la Tecnología, Servicio Nacional de Aprendizaje, Medellín, Colombia
2 Red Tecnoparque Nodo Medellín, Servicio Nacional de Aprendizaje, Medellín, Colombia

E-mail: jbolanos@sena.edu.co, sernal@sena.edu.co

Abstract. Parallel programming is a computing model in which the computations are run on multiple processors simultaneously. In this work, a parallel computing system is implemented through the network connection of a set of Raspberry Pi Cluster 3 boards, using 4 nodes (one master and three slaves) that acts as a single computer with improved processing speed. A Linux based operating system, Raspbian is used in the present work, our purpose is to provide an economical, scalable, powerful, mobile architecture and especially low energy consumption as an alternative to the use of high performance computing to execute Python code distributed from the Jupyter notebook platform. The main contribution of this work is the configuration of the cluster to be able to execute some learning machines.

1. Introduction

Machine learning algorithms provide support for decision making by extracting knowledge from large data sources; however, the enormous volumes and speed with which information is generated makes its processing require a robust infrastructure, which demands considerable investments in Data Centers. With this, it is common for inconveniences related to the need to acquire high performance hardware, guarantee adequate temperature (cooling), physical space and energy consumption, among others. This could lead to significantly high economic costs [1].

Among the Frameworks that make it possible to apply machine learning, are Tensor Flow and Keras. The first is a Google open source library that is based on deep neural network [2]. This makes possible the implementation of various machine learning algorithms such as neural networks, stochastic gradient, regression and support vector machines (SVM) [3] and can be executed in distributed systems [2]. On the other hand, Keras, was also developed for neural networks and this is a Phyton library; however, it is currently connected to Tensor Flow and provides the main advantage for its execution to possess minimum knowledge [4]. Parallel computing through clusters (consisting of a few to many nodes) to implement open and low-cost platforms for education and experimentation in the area of artificial intelligence, guarantees access to infrastructure at scale that can be conceived as alternatives to the use of high performance computing [5].

This experiment consisted of a set of single-board computers interconnected in a network. These are economical, energy efficient, of considerable performance and power [5] called Raspberry Pi 3. As a result, the capacity and scalability were demonstrated when executing Python code distributed from the Jupyter Notebook platform [1,6].
2. Experimental procedure

This section discusses the steps necessary to configure the cluster that allowed you to run the Python code distributed from the Jupyter Notebook platform.

Initially Jupyter was installed on the controller node, see Figure 1. For this case, memory limits were taken into account, which are undoubtedly a problem when executing very heavy algorithms. It was installed on all the nodes that are part of the cluster to launch work on them. The controller node is configured to serve as an access point and provides a transparent connection with the rest.

![Figure 1. Jupyter notebook running on node and remote connection.](image)

For the parallelism functionality, the package called IPython Parallel (ipyparallel) was used [7]. Most distributions do not have it available within their package management systems, therefore, it is necessary to install it with the manager pip3 [8].

In Figure 2, the ipyparallel is structured in four sections: the engine, the hub, the planner and the client. The engines perform the actual execution of the code from their program; the code is executed on the client and parallel calls are made; the hub and the scheduler are the parts that allow the client and the server to communicate with each other.

![Figure 2. Ipyparallel architecture [7].](image)

In the configuration of ipyparallel there are two approaches to start the controllers and engines throughout the cluster: SSH protocol [2] and MPI [10]. For this project it was decided to use SSH, which is the simplest configuration. To start the controller and the motors, ipcluster was used together with SSH. To run the engines on several nodes, the ipcluster_config.py file was configured on the line of code called SSHEngineSetLauncher [10].
In Figure 3. The engines are specified in a dictionary by IP of the node and the number of engines that will run on that node. The controller node with a single motor, the rest are work nodes, each with 4 motors.

For the validation process, an algorithm that creates a neural network was used to classify sports images. A dataset called MNIST-Sports was used [10]. The dataset has between 5000 and 9000 images of each sport. The images are in diminuto size of 21 × 28 pixels in color and are a total of 77,000. Subsequently, the data set was divided into 80-20 for training and for testing respectively. In turn, the training set also subdivided it into another 80-20 for training and validation in each learning iteration (EPOCH). Figure 4. The neural network used was the convolutional neural networks (CNN) [11], which is an algorithm used in Machine Learning to give the ability to "see" the computer. Thanks to this, you can classify images, detect various types of tumors automatically, teach driving autonomous cars and a host of other applications.

3. Results and discussion
Initially, different algorithms found in the literature that use machine learning (ML) were implemented and validated, this to identify their properties and select the one that best meets the requirements of the application. The algorithms studied were based mainly on the hardware requirements necessary for their execution, as well as artificial intelligence techniques. In this section, the steps that were carried out to carry out this project are mentioned.

After starting the cluster, the correct operation was verified. Figure 5 shows the cluster of 4 running nodes on the Jupyter platform running on the main node and monitored remotely from a web browser through the Multi-Tabbed PuTTY (MTPuTTY) program [12], which allows the SSH connection to the nodes of the cluster.
In Figure 6, you can see the list of nodes configured in the cluster, which are going to process the distributed data of the selected dataset for the ML algorithm execution test on Raspberry Pi (RPI) platforms.

The connection test of the nodes configured in the cluster was performed by means of a ping, see Figure 7; which is a diagnostic tool in computer networks that checks the communication status of the local host with one or more remote computers on a network that are running. It can be seen that all nodes respond satisfactorily to the command.
Finally, after checking the cluster configuration we proceeded to execute the Python code of the selected example through the following steps: Figure 8.

Regarding the results obtained, taking into account that the objective of the project was to determine the ability of the RPI platform to execute distributed code for data analysis using ML libraries, it was determined that the RPI platform can run code with these characteristics, taking into account some limitations such as memory, and type of operating system. Figure 9 shows the result of the execution of the algorithm in a single node, we can appreciate the execution time of 1132 seconds, compared to the execution of the same code in the same conditions in the configured cluster, see Figure 10, the time of Execution presents a considerable reduction in runtime in 331 seconds. All this comparable to the execution of the algorithm in a mid-range laptop, which gives us an execution time of 236 seconds, see Figure 11.

![Figure 8. Sample execution process.](image)

![Figure 9. Execution of the algorithm in a single node.](image)

![Figure 10. Execution of the algorithm in the cluster using the four nodes.](image)
4. Conclusions
Clusters implemented with Raspberry Pi 3 are efficient when emulating high performance computers. This type of architecture is interesting therefore it is very useful when working with codes, applications and distributed data, and with the ability to scale if you need more resources.

Taking into account the results at runtime of the algorithm we can take into account for future work, the expansion of the cluster with more nodes in order to reduce the execution time to a value below that presented in Figure 11.

Another future work would be the implementation of the cluster in an RPI 4 system close to reaching the market and with maximum memory capacity. As well as being able to explore other alternatives of neural networks with other types of data.

References
[1] C M Crúz Salazar, Data 2015 Implementación de Hadoop 2.7 sobre una placa Raspberry pi v3 Revista de la Facultad de Ciencias de la UNI, REVCIUNI 18(1) 46
[2] Kiepert J 2013 Creating a raspberry pi-based beowulf cluster (USA: Boise State University)
[3] Quimis Barzola W D, Ramirez Reyes L J 2019 Comedero de aves inteligente: sistema de clasificación, monitoreo y reporte de especies de aves por medio de un comedero construido con un ordenador de placa reducida y análisis de imágenes usando machine learning (Ecuador: Universidad de Guayaquil)
[4] Espinoza Villafuerte N 2019 Diagnóstico automatizado para la clasificación de café trillado con broca mediante procesamiento de imágenes con Deep Learning (Peru: Universidad Peruana Unión)
[5] Soaffran J, Garcia G, Souza M A, Penna P H, Castro M, Góes L F, Freitas H C 2016 A low-cost energy-efficient raspberry pi cluster for data mining algorithms (Germany: European Conference on Parallel Processing)
[6] Houseux P, Hale J S, Borda S P 2017 Accelerating Monte Carlo estimation with derivatives of high-level finite element models Computer Methods in Applied Mechanics and Engineering 318 917
[7] Kumar R A 2019 Face recognition using raspberry Pi-3 in IoT (India: Blue Eyes Intelligence Engineering and Sciences Publication)
[8] Kyuchukova D, Hristov G, Zahariev P, Borisov S 2015 A study on the possibility to use raspberry pi as a console server for remote access to devices in virtual learning environments (Portugal: International Conference on Information Technology Based Higher Education and Training)
[9] Doucet K, Zhang J 2019 The creation of a low-cost raspberry pi cluster for teaching (Canada: Proceedings of the Western Canadian Conference on Computing Education)
[10] Hirohara M, Saito Y, Koda Y, Sato K, Sakakibara Y 2018 Convolutional neural network based on SMILES representation of compounds for detecting chemical motif BMC Bioinformatics 19(19) 526
[11] Jasmin M, Angelo Virgin G, Jeyalakshmi G, Hemalatha B 2019 Design and implementation of WIFI controlled robotic rover based on raspberry-pi and Arduino International Journal of Recent Technology and Engineering 8(1) 22