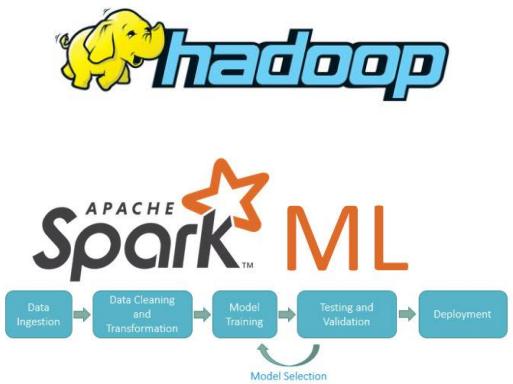
Leveraging low-end Hardware for a Distributed Machine Learning Cluster

By: Derek Manning
Advisor: Dr. Peilong Li

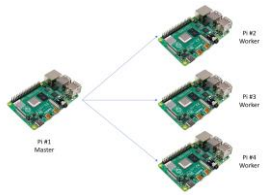
Abstract

In the last decade, the Raspberry Pi computer has evolved from a simple, recreational device for hobbyists and educators to a legitimate option for computationally and memory intensive tasks. The latest Raspberry Pi 4 boasts a 1.5GHz, 64-bit quad-core processor, with the option of 1GB, 2GB, or 4GB models. The field of distributed computing has seen a similar expansion in interest from non-typical users with the advent of Big Data. With open-source frameworks such as Apache Hadoop and Apache Spark, distributed computing can be deployed by anyone capable of learning the structure of the frameworks, assuming they have the proper hardware backbone. While clock speed isn't necessarily a bottleneck for distributed tasks, memory is, especially for machine learning. Node memory sizes of at least several gigabytes are required to support both the distributed framework and dataset demands without error. Until recently, this has meant that each node would have to consist of a several-hundred or thousand-dollar machine. The 4GB Raspberry Pi 4 offers a much cheaper alternative to creating a simple, yet viable machine learning cluster.



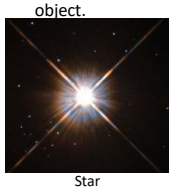
Background

The main reason that Raspberry Pi's have not been widely used with distributed systems has been the limited memory space of all previous models. Prior to the Raspberry Pi 4, the maximum RAM in a Raspberry Pi was capped at 1GB. This is sufficient for hobbyist and educational activities, however more complex tasks require more memory space for optimal results. Previous Raspberry Pi cluster users struggled with poor cluster performance using 1GB memory nodes, as slow performance and potentially out-of-memory errors are triggered by Spark as the dataset size is increased past what <1GB RAM can manage. These issues are solved for functional performance by using exclusively 4GB memory versions for cluster nodes.



Data

The data used in this project is open sourced data from the Sloan Digital Sky Survey (SDSS) project, a long-term project to construct three-dimensional maps of the universe and measure the spectra of different astronomical objects throughout it. Machine learning models were trained on a subset of this data consisting of five broad-band filter responses based on the Thuan-Gunn System (u, g, r, i, z), the measured redshift, and the object class (star, galaxy, or quasar) of each individual object.


Star


Galaxy


Quasar

Results

The machine learning models used for comparison were identically structured neural networks with an input layer of size 6, hidden layer sizes of 64, 64, and 32 respectively, and output layer of size 3. The baseline model used was a Keras-Tensorflow model trained on a single Raspberry Pi node, which was then contrasted with neural networks built using the Spark MLlib with several different node/resource configurations. All models were trained on 400,000 samples over 100 epochs, and then predicted object classes (star, galaxy, quasar) on 100,000 samples set aside for model testing. The primary and secondary metrics being compared between models are model training time and test dataset accuracy, respectively. Table 1 shows the resulting comparison between four different model configurations.

Configuration	Training Time	Testing Accuracy
Keras-Tensorflow	50 minutes 35 seconds	82.5%*
PySpark 2 worker nodes with 2 threads each	56 minutes 17 seconds	96.9%
PySpark 2 worker nodes with 4 threads each	29 minutes 1 second	96.6%
PySpark 3 worker nodes with 4 threads each	18 minutes 53 seconds	96.8%

Table 1: Training Time and Testing Accuracy Comparison

* Keras model exhibited high degree of overfitting due to converging much faster than the PySpark models. Keras offers a much wider variety of loss functions and optimizers, being a much more popular and widely contributed library. With an optimal number of epochs, the Keras model can also achieve accuracy ~96-97%, though the primary focus of this comparison is the training time.

Conclusions

The results obtained from this project were significant in showing that the new, higher memory Raspberry Pi 4 is a legitimate option for distributed computing tasks, not limited to machine learning. Training a neural network using PySpark with 3 workers resulted in a training time over 2.5X faster than an identically structured Keras version, and the accuracy of the PySpark model was on par with even the most highly optimized Keras models. The differences between using PySpark and Keras would also likely be much more pronounced if a much larger dataset, on the order of gigabytes, were used. The SDSS data used with the models in this project was a relatively small dataset (~500MB) for distributed machine learning, and only included six distinguishing features, also relatively small in terms of the dimensionality of the data. It is very likely that projects involving datasets gigabytes in size and with many more features would see a much greater performance difference between a distributed and single-node setup, especially if more worker nodes were implemented.

References

[1] Hadoop. (2010). Apache Software Foundation. https://hadoop.apache.org
[2] Spark: Cluster Computing with Working Sets. (2010). Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, Ion Stoica. HotCloud 2010.
[3] Sloan Digital Sky Survey IV: Mapping the Milky Way, Nearby Galaxies, and the Distant Universe. (2019). Ahumada et al.
[4] Keras. (2015). Francois Chollet. https://keras.io
[5] MLlib: Machine Learning in Apache Spark. (2016). Xiangrui Meng, Joseph Bradley, Burak Yavuz, Evan Sparks, Shivaram Venkataraman, Davies Liu, Jeremy Freeman. Journal of Machine Learning Research (JMLR).

See detailed walkthrough at: https://github.com/dmanning21h/pi-cluster