Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 82 (2016) 90 – 98

Procedia Computer Science

Symposium on Data Mining Applications, SDMA2016, 30 March 2016, Riyadh, Saudi Arabia

Performance of a Low Cost Hadoop Cluster for Image Analysis in Cloud Robotics Environment

Basit Qureshi[a,*], Yasir Javed[a,e], Anis Koubâa[a,b], Mohamed-Foued Sriti[c], Maram Alajlan[d]

[a] *Prince Megrin Data Mining Center, Prince Sultan University, Saudi Arabia*
[b] *CISTER/INESC-TEC, ISEP, Polytechnic Institute of Porto, Porto, Portugal*
[c] *Al-Imam Mohammad Bin Saud University, Saudi Arabia*
[d] *King Saud University, Saudi Arabia*
[e] *eFIT, UNIMAS, Sarawak, Malaysia*

Abstract

With the emergence of cloud robotics, the cloud computing paradigm becomes increasingly attractive to robotics, where the cloud acts as the remote brain of low-cost robots, such as commodity drones. The idea is to offload heavy computations, like image processing, from the robot to the cloud; process it in short time (near real-time) and send back commands to the robot. This paper investigates the performance of a back-end cloud computing framework in deploying robotics-like applications (i.e. image analysis and processing) using low-cost Hadoop clusters. The design of a low-cost mini-data center built with readily available commodity 32-bit ARM boards, i.e. Raspberry Pi 2 Model B, is presented. Furthermore, the performance of RPi-based clusters is extensively tested with different types of data including text, text/image and image, and a comparative analysis against Hadoop cluster running on virtual machines is presented. The Hadoop Image Processing Interface (HIPI) Library was used and also configured to optimally utilize the Pi Cluster resources for improved performance. Results show that the RPi Hadoop cluster lags in performance when compared to Hadoop cluster running on virtual machines, the low cost and small form factor makes it ideal for remote Image analysis in surveillance / disaster recovery scenarios where UAVs can transmit image streams to the Cluster for remote processing.

Peer-review under responsibility of the Organizing Committee of SDMA2016
Keywords: UAVs; HIPI; Hadoop cluster; Image analysis; Distributed processing; Embedded systems

1. Introduction

Nowadays, rapid development in Cloud Robotics research has enabled manufacturers to mass market Unmanned Aerial Vehicles (UAV) often known as drones. These UAVs are capable of flying with a remote controlled device and can capture image and video streams of data. Recently there has been a trend in use of UAVs in surveillance applications where real-time images can be captured. In such applications, detection of suspicious items and objects is a fundamental task that requires intensive computation and processing of objects features and parameters against database of suspicious objects. Since UAVs have limited onboard processing and storage capabilities that restrict their

* Corresponding author. Tel.: +0-000-000-0000 ; fax: +0-000-000-0000.
E-mail address: qureshi@psu.edu.sa

Peer-review under responsibility of the Organizing Committee of SDMA2016
doi:10.1016/j.procs.2016.04.013

abilities in processing and handling such tasks, these could be offloaded to a cloud computing environment capable of storing petabytes of data and processing intensive image processing and analysis applications.

In the light of these limitations, we investigate the use of a low cost cloud infrastructure built with readily available commodity ARM processors that can be used to build clusters. Raspberry Pi (RPi)[1] are a class of embedded computers build with 32-bit ARM processors with on-board memory, Network I/O and Storage. These computers have desirable characteristics such as low-cost, low-power yet high performance boards capable of intensive computation and storage. Michael Laurenzano in[2] characterize the performance and energy of HPC computations drawn from a number of problem domains on current ARM and x86 processors. Pleiter and Richter in[3] study the energy efficiency of embedded system with ARM Cortex-9 processors for scientific numerical applications. Michael Cloutier in[4] investigate the currently available 32-bit low-cost Embedded Computing platforms including Raspberry Pi[1], Beaglebone Black[5] as well as other systems. Abrahamsson in[6] investigate the use of Raspberry Pi to build an affordable and energy-efficient cloud computing cluster with 300 Raspberry Pi computers.

Apache Hadoop[7] is an open-source software framework derived from Google's MapReduce and Google File System (GFS)[8]. Recently, Hadoop became a de-facto standard in the area of BigData analytics. A growing number of Big Data applications are being run on the public cloud service providers. Manikandan et. al.[9] investigate use of Big Data Analytics techniques for classification and clustering of Big Data using MapReduce. Loewen in[10] test performance of virtualization within cloud with variable dataset of different sizes. Mathiya[11] notice that configuring YARN parameters in Hadoop 2.0 greatly improves optimal performance for various applications. Xhafa et.al. in[12] evaluate Data Mining Frameworks under a Hadoop cluster and show that improvements in time efficiency to a certain scale for most mining functions. Unfortunately, there are many limitations to Hadoop supporting image/video analysis applications for the following reasons:

- Lack of image read/write interface: Hadoop provides many interfaces for reading and writing text data but no read/write interface for image data. In particular, object recognition applications may require the information from preceding and subsequent images or frames of videos.
- Existing Image and video processing applications are not compatible with Hadoop framework. Much work needs to be done in allowing popular object recognition libraries such as OpenCV which are coded in C/C++ to execute in Hadoop environment.
- The HIPI Library[13] introduced in 2011 provides an interface for storing images in HDFS using Hib file format. Since Hadoop uses Input Split format for splitting large files based on HDFS block size, the performance of image analysis algorithms running on HIPI desire much improvement.

The contributions of this paper are in three-folds: First, we design and setup a low-cost mini-data center with commodity RPi computers organized into four clusters each with five machines. We discuss the challenges faced to build the cloud infrastructure and to install and configure the cloud applications for big data processing. Second, we evaluate the performance of the RPI clusters for computation and storage intensive applications, in terms of execution time. In particular, we contrast the performance of Hadoop applications running on RPI Cluster with Hadoop running on regular PCs in virtualized environments. Finally, we extend the HIPI library to optimally utilize the computation and storage capabilities of the RPI Clusters. Furthermore, we evaluate the performance of RPI cluster using the extended HIPI Library and compare the results of the library for large scale compute intensive applications for Image analysis-as-a-service.

2. Related Work

Manikandan et. al.[9] investigate use of Big Data Analytics techniques such as Joins, Indexing, graph search and applied these for classification and clustering of Big Data using Map reduce. Loewen in[10] test performance of virtualization within cloud with variable dataset of different sizes. Their results show improved performance when using distributed processing. Mathiya[11] notice that configuring YARN parameters in Hadoop 2.0 greatly improves optimal performance for various applications. They study and analyze customizing parameter configurations setting for the performance tuning of Apache Hadoop jobs and notice improvement in utilization of available hardware resources. Xhafa et.al. in[12] evaluate Data Mining Frameworks under a Hadoop cluster and show that improvements in time effi-

ciency to a certain scale for most mining functions. Very recently researchers have started to address issues of image and video processing on Hadoop platform. Authors in[14] propose an approach for fast and parallel video processing on Map-Reduce based clusters. The researchers implemented face detection and motion detection and tracking algorithms on a Hadoop cluster. Husain in[15] studied the performance of Hadoop cluster with 1, 5 and 10 nodes to extract textual information from annotated video streams of 3GB size. They conclude that executing the text extraction algorithm on the Hadoop cluster was much faster compared to a single machine. To achieve massive Image storage and processing, HIPI framework[13] provides a solution for how to store a large collection of images. HIPI framework is designed to run on Hadoop Distributed File System (HDFS). Currently, HIPI only supports specific image formats, such as, JPEG, PNG and PPM. HIPI is fast becoming popular for fast image storage and retrieval. Recently Wilder[16] proposed an extension to HIPI by providing support for TIFF image format for use in massive satellite image data storage in Hadoop Cluster. Zhao in[17] extend HIPI framework to support storage of video in the MPEG format. The proposed open source Hadoop video processing interface (HVPI) extends Hadoop to support video analytic applications.

Deploying large numbers of small, low-power cores has been gaining traction recently as a system design strategy in high performance computing (HPC). The ARM platform that dominates the embedded and mobile computing segments is now being considered as an alternative to high-end x86 processors that largely dominate HPC. Michael Laurenzano in[2] characterizes the performance and energy of HPC computations drawn from a number of problem domains on current ARM and x86 processors. They study the performance, energy and energy-delay product of applications running on these platforms and conclude a promising future for low-cost and energy efficient ARM processor embedded platforms. Pleiter and Richter in[3] study the energy efficiency of embedded system with ARM Cortex-9 processors for scientific numerical applications. The Glasgow Raspberry Pi Cloud[18] was the first cluster of its kind that is entirely composed of Low cost and small form factor Raspberry Pi Embedded Computers. The PiCloud is implemented with 60 Raspberry Pi Model B computers. Michael Cloutier in[4] investigate the currently available ten, 32-bit low-cost Embedded Computing platforms including RPi, Beaglebone Black[5] etc. Based on performance results extracted from STREAM and from various experiments measuring compute performance, storage, Network I/O etc., the authors recommend Raspberry Pi for building a HPC cluster. Abrahamsson in[6] investigate the use of Raspberry Pi to build an affordable and energy-efficient cloud computing cluster with 300 Raspberry Pi computers.

In this paper we present the design of a low-cost cluster for Hadoop Map reduce framework. Furthermore, we extend the HIPI Framework to store image data and tweak its performance based on the Pi Cluster.

3. The RPi Cluster

The RPi is a System on chip combining a Quad-core ARM Cortex-A7 processor clocked at 900MHz by default (can be over-clocked to 1GHz) with embedded Broadcom GPU and 1024 MB SDRAM (shared with GPU) Memory. There is an SD card slot for storage and I/O units such as USB, Ethernet, audio and HDMI video ports. Table 1 summarizes the hardware specifications of Raspberry Pi Model 2 B. In terms of software, the Raspberry Pi Foundation recommends using NOOBS or Raspbian as operating system. Raspbian is based on the well-known Debian (Linux Distribution) optimized for ARM instruction set, providing better performance for floating point arithmetic operations. Other operating systems such as Ubuntu MATE, OSMC, Microsoft Windows 10 IoT, PINET, Fedora, Google Chromium, UNIX and RISC OS are also compatible. The RPi Model 2 B comes with a default processor speed of 700MHz that can be over-clocked to 1GHz. Although with over-clocking the computing performance increases slightly, the life-span of the processor would decrease. Overall RPi is capable of handling compute and storage load and comes at a very affordable price with low power consumption.

3.1. Design of RPi Cluster

The Bolzano University experiment with RPi[6] inspired us to consider setting up our own RPi cluster. The primary goal of our cluster is to allow remotely located robots / UAVs to transmit data for computation to the RPi cluster, which would in-turn process the data in a relatively short-period of time and transmit back the results to the UAV. This would transfer the computation load from the remote robot and off-load it to the RPi Cluster that is more than capable of handling larger computation loads compared to the robots' limited onboard processor. With this goal we intend to address the following objectives:

- Design and development of a Hadoop based RPI Cluster that is durable and scalable.
- Configuration, development and testing of various computation and storage intensive applications on the RPi cluster consisting of 20 Raspberry Pi's.

Table 1: RPi hardware properties

Property	Details	Property	Details
System on chip	Broadcom BCM2836	Ethernet	Onboard 10/100 Ethernet RJ45
CPU	900 MHz quad-core ARM Cortex-A7	USB	4x USB 2.0 connector
GPU	Broadcom VideoCore IV @ 250 MHz	Video / Video Output	3.5mm jack, HDMI
Memory	1024 MB SDRAM	Storage	SD card supports class 10

In what follows, we present network topology, configuration and installation of Hadoop on RPi Cluster.
Network Topology and Configuration. The RPI cluster uses a star topology with a 24-port Giga-bits-per-second smart managed switch acting as the core of the network as can be seen in Figure 1. The RPi's connect to four 16-port switches realizing the rack-configuration in Hadoop with each switch connecting to the core switch. Currently, five RPi's connect to each switch allowing further scalability of the cluster. The master node as well as the uplink connection to the Internet through a router is connected to the core switch. The current design allows easy scalability with upto 60 RPi connected in the Cluster that can be extended up to 300 RPi's. Currently each RPi is individually supplied by 2.5Amp power supplies that provides ample power for running RPi's in the over-clocked mode.

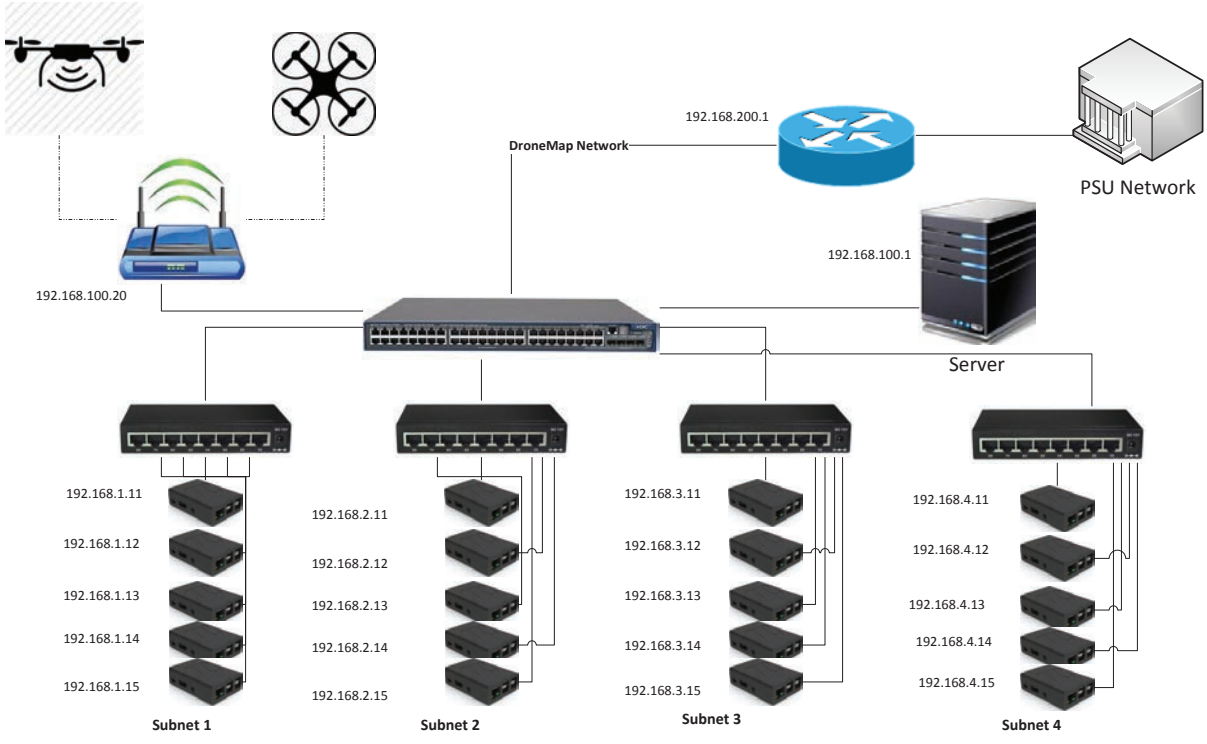


Fig. 1: Network Topology diagram for RPi Cluster

Raspbian and Hadoop Image Installation. The Raspbian OS image is based on Debian that is specifically design for ARM processors. Using Raspbian OS for RPi is easy with minimal configuration settings requirements.

Each individual RPi is equipped with a Class 10, 16 GB SD card capable of up to 80MB/s read/write speeds. We created our own image of the OS which was copied on the SD cards. Additionally Hadoop 2.6.2 is installed on the Image. When ready, these SD cards are plugged into the RPi systems and mounted. The Master node is installed on a regular PC running Ubuntu 14.4 and Hadoop. The master node implements name-node of the Hadoop cluster only.

Hadoop configuration. Hadoop 2.6.2 was installed on the cluster due to availability of YARN which improves the performance of the Map-Reduce jobs in the cluster. To optimize the performance of the RPi Cluster, yarn-site.xml and Mapred-site.xml were configured with 852 MB of resource size allocation. This is due to the reservation of 162MB of available RAM for the operating system as well as the GPU memory bus on the RPi. The default container size on the Hadoop Distributed File System (HDFS) is 128 MB. Each RPi's node was assigned a static IPv4 address based on the four rack configuration and all nodes were registered in the Master node. YARN and HDFS containers and interfaces could be monitored using the web interface provided by Hadoop.

The RPi Cluster was tested extensively for performance using a compute intensive application (Computing the value of Pi) as well as an I/O intensive application (word count) or text files of various sizes.

3.2. Extending the HIPI Library for the RPi Cluster

While Hadoop provides many interfaces for reading and writing text data unfortunately there are no read/write interfaces for images. Each image file stored in HDFS would utilize the containers inefficiently, i.e. a container of default size 128MB is reserved from each image file regardless of its actual size, which could be a few kilo-bytes or close to 128MB. For large number of small image files, which are typically generated in surveillance applications or satellite imagery, Hadoop HDFS storage mechanism is inefficient. To remedy this, HIPI[13] provides a Hadoop image processing interface (HIPI) and propose a structure named HIPI Image Bundle to make Hadoop jobs more efficient. For a large number of small sized images, HIPI provides an interface which allows for the storage of these images files into a Hib indexed archive. A Hib archive is composed of two files `hib` and `dat`. The indexing information for all these images files is available in `.hib` file, whereas the `dat` file stores the image files in an indexed format. The Hib archive allows maximum utilization of HFDS containers whereas decreasing the clutter of multiple image files as well as the inefficient use of HDFS containers.

The Map-Reduce Framework of Hadoop relies on `InputFormat`, `RecordReader` classes to convert input files into key-value pairs and pass these pairs to map function[17], and relies on `OutputFormat` and `RecordWriter` classes to write key-value pairs, which are output from Mapper/Reducer, into an output. Hadoop Map-Reduce Framework splits input file into logical `InputSplits` by calling `getSplits()` method of `InputFormat` class. According to the `InputSplits` information, `jobtracker` schedules map tasks on `tasktrackers` and assign each `InputSplit` to an individual Mapper. Each mapper calls `createRecordReader()` method of `InputFormat` class to obtain `RecordReader` for this `InputSplit`. `RecordReader` reads this `InputSplit` and generates key-value pairs from it. Each key-value pair is passed to map function to process. We modify and re-write InputSplit related methods in the classes provided by Hadoop. The purpose is to split each Hib file based on the number of available YARN interfaces instead of the default split size 128MB. In our implementation with 20 RPi's in the cluster, each Hib file is split into *n* `InputSplits` for all map tasks, where *n* is the number of available YARN containers. This tweaking allowed us to optimally utilize the RPi cluster nodes for individual and parallel jobs increasing the degree of parallelism thus allowing the jobs to complete earlier. Performance analysis of the modified library using the Average Pixel Count (APC) program is presented in the next section.

4. Performance Analysis

In this section we present the results of various experiments carried out to investigate the performance of the RPi Cluster. We evaluated the performance of the cluster in terms of three evaluation metrics, namely, *i)Compute performance, ii) Compute & Intensive text I/O performance and iii) Compute & Intensive image I/O performance.* We run these sets of experiments on the Clusters in three configuration modes and compare the execution times for all experiments. In all three configuration modes, the master node runs only the `namenode` task with `jobtracker` whereas all computation is performed in the `datanodes` running in the RPi's. Each RPi runs a single `datanode` and

is configured to store HDFS blocks with a default replication of 3 blocks. All storage allocation and computation is done exclusively in the data nodes.

Configuration mode 1: In this mode the individual RPi's run on default CPU clock speed at 700 MHz.The default HDFS container size is 128 MB. The max memory allocation for YARN and MapReduce containers is set to 426 MB. This will allow us to investigate the utilization of containers and study the effect of memory utilization versus number of map task allocations in Hadoop.

Configuration mode 2: In this mode the configuration is similar to mode 1 with a few tweaking of parameters for performance. The CPU clock for all RPi's is set to 1000GHz. The max memory allocation for yarn and MapReduce containers is set to 852 MB. All map tasks as well as reduce tasks are allocated the max available container size. This will allow us to study the impact of container size on the performance of map tasks.

Configuration mode 3: In order to benchmark the performance of the RPi Cluster, we build a small cluster entirely running on virtual machines. This cluster is composed of five personal computers with Intel i7 processors clocked at 3GHz each having 4 GB of RAM, running Microsoft Windows 7. These computers are connected in a similar topology to the RPI cluster with a core switch connected to the master node. A 16 port switch connects the four data nodes with the core switch. Each machine runs an image similar to the RPI Cluster with Ubuntu 14.4 as well as Hadoop 2.6.2 installed on the image. Each machine is assigned a static IPv4 address and registered in the master node. The virtual machine is run using VMWare workstation on Windows 7 host operating system installed on the five computers. The Hadoop configuration parameters are set to be similar to Configuration 2.

Table 2: Comparison of the configuration modes

	Configuration 1	Configuration 2	Configuration 3
Master Node	Intel i7 at 3.00 GHZ 64Bit Win 7	Intel i7 at 3.00 GHZ 64Bit Win 7	Intel i7 at 3.00 GHZ 64Bit Win 7
Number of Data Nodes	20	20	4
Data Node Clock Speed	700 MHz	1000 MHz	3000 MHz
OS	Raspbian OS	Raspbian OS	Ubuntu 14.4
Storage (GB)	16 GB	16 GB	40 GB
RAM	856 MB (available)	856 MB (available)	3 GB (available)
Virtual Machine	Only Master Node runs OS in VM	Only Master Node runs OS in VM	All nodes on VMWare-workstation

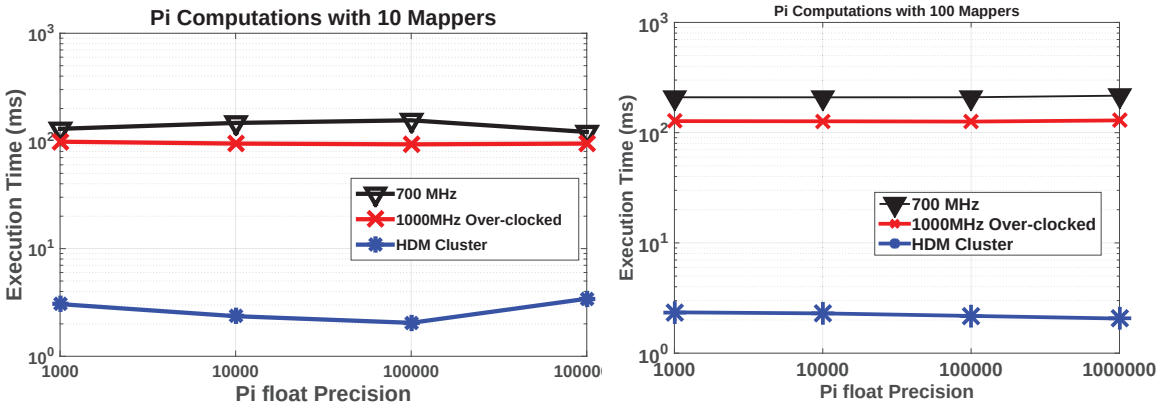
We benchmark the performance of the three configurations based on three criterion.

Criterion A: To investigate the compute performance in terms of execution time, of the clusters. We execute the compute pi program that computes exact m binary digits of the mathematical constant π using a quasi-Monte Carlo method and MapReduce. The precision value m is provided at the command prompt with values ranging from 1×10^3 to 1×10^6 increased at an interval of 1×10^1. Each of these is run against a number of map tasks set at 10 and 100. We study the impact of the value of m versus the number of map tasks assigned and compute the difference in execution time for the completion of these tasks.

Criterion B: To investigate the execution time for the compute as well as Text read/write of the clusters, we execute the word-count program that determines the number of times each word appears in a text file. The program reads text files of 3MB, 30MB and 300 MB each. We compare the amount of time taken in executing the tasks.

Criterion C: To investigate the execution time for the compute as well as Image read/write performance of the clusters, we execute the HIPI librarys APC program that computes the average number of red, blue and green pixel in a Hib archive file. Two archive files are generated from datasets of images taken from Oxford University (OXU) flowers training dataset[19] and Caltech CUB-200 Birds datasets (CUB)[20]. Both of these datasets are composed on high-resolution JPEG format images. Two Hib archive files were created with sizes 60MB and 2.5GB respectively. Although the size of dataset in[20] is only 670MB, many image files were copied to expand the Hib file to 2.5GB. A fat jar file embedding all libraries with dependencies in Java, was created with modified APC program. The fat jar was copied on the master nodes in all clusters for execution. In this criterion, we focus on the execution time in terms of seconds to run the APC program.

Figure 2a shows the comparison of the average time consumed by the π computation program in 30 iterations with 10 map tasks. The program was executed in the Cluster configuration mode 1, 2 and 3. The performance of mode 1 is slower compared to mode 2 with RPi's over-clocked to 1000 MHz. The performance of the cluster in configuration



(a) (b)

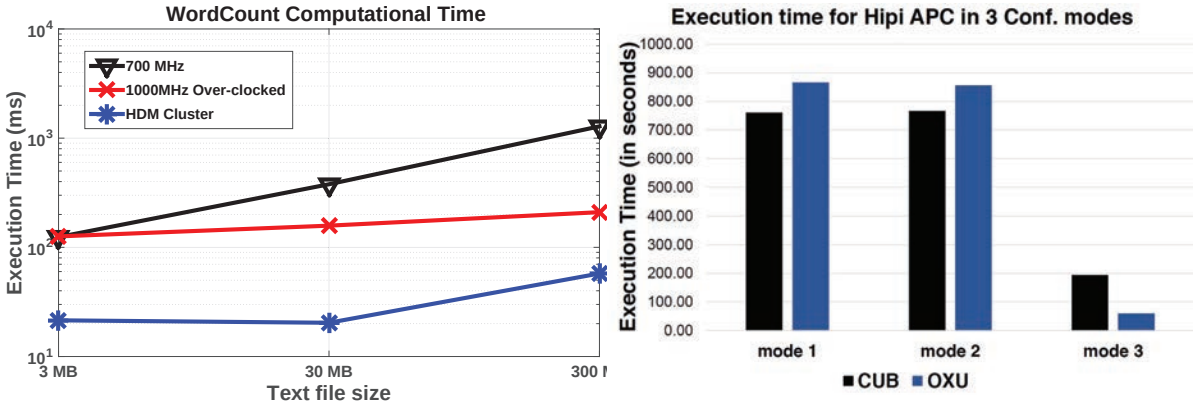
Fig. 2: Comparison of π program with m=10^3, 10^4, 10^5 and 10^6 with (a) 10 Mappers and (b) 100 Mappers

mode 3 is much better as compared to the RPi cluster. Figure 2b shows the comparison for all the three configuration for π computation program with 100 map tasks. As can be seen from the figure, the performance is slightly slower with 22% increase in average execution time when comparing configuration mode 1 and mode 2. This clearly shows that over-clocking the RPi's improves the compute performance.

Figure 3a shows the average time consumption in executing the word count program in 30 iterations. With the text file size 3 MB, the performance of configuration mode 2 with over-clocked RPi's is almost 50% better when compared to mode 1. As the file size increases to 30 and subsequently to 300MB the performance decreases with mode 2 better by only 27% and 19% respectively. This clearly shows the impact of the size of tasks on the performance of the cluster. Similarly, we also noted that the performance of HDM cluster also decreases under larger loads of computation.

Figure 3b shows the comparison of the execution time for image datasets using the HIPI Library APC program. The OXU dataset is executed using the APC program defined in the HIPI Library whereas the CUB dataset is executed using the APC program in the modified HIPI Library. Consequently in HDFS only 1 container is created for OXU due to the InputSplit parameter being lower than the HDFS container size, i.e. 60MB file size is smaller than the 128MB Split size, whereas 20 containers of size 128MB each were created from CUB dataset (2.5 GB files). As expected, the execution time for CUB is significantly larger than OXU dataset in mode 3. This is due to the fact that small number of data nodes in the mode 3 cluster forces HDFS to create replicated blocks of data in all nodes. More or less each node possesses a large number of unique containers needed for processing that are readily available and need not be transported on the network, consequently decreasing the network traffic. The comparative execution time of the two datasets in mode 3 is at an acceptable level.

When comparing the two datasets in RPi clusters in mode 1 and mode 2, we observe that the difference in execution time of the APC program in standard HIPI versus the modified HIPI library is significant. First we execute the OXU dataset on the clusters in mode 1 and 2, as can be seen in Figure 3b, the difference is negligible. Due to the replication factor set to 3, the impact of InputSplit size in case of mode 1 and mode 2 with OXU dataset, results in only one container which would be replicated to 3 nodes. Since only three nodes contain the copy of the file, only three nodes could execute it in parallel, whereas other nodes wait for the next map task, therefore reducing the performance of the cluster. On the other hand, the CUB dataset which was executed using the modified version of the library performs in a similar fashion in modes 1 and 2 of the RPi Clusters. It is worth noticing, when comparing the execution time for OXU and CUB datasets, despite the fact that the CUB dataset file is considerably larger than OXU, the execution time for CUB is lower compared to OXU. A reasonable explanation for this observation is that the CUB dataset executed using the modified program generates 20 InputSplit with each RPi assigned a map task. This allows map tasks to execute on their respective containers in the RPi's in parallel providing better execution time. This clearly proves our hypothesis that modifying the InputSplit yields significant performance improvement in Hadoop environment when working with large image files.



(a) (b)

Fig. 3: (a). Time analysis of the Comparison of Criterion B with wordcount program executing on text files with sizes 3,30 and 300MB. (b)Execution time Comparison of Criterion C with CUB and OXU datasets

5. Conclusions and future work

In this paper we investigate the configuration and deployment of a Hadoop cluster using the low-cost Raspberry Pi computers. We presented the design of a cluster based on 20 RPi's that is scalable and can be extended to 300 nodes. We also extended the HIPI Library to optimally utilize the RPi Clusters resources for large scale Map-Reduce based image processing and analysis. We benchmarked the performance of the RPi cluster against three configurations and study the impact of various benchmarking programs including Pi computation, word-count and Average Pixel count. Results show that the performance of RPi cluster when over-clocked is significantly improved for less compute intensive tasks. As for heavy compute and I/O intensive tasks the performance of the two configurations is negligibly different. On the other hand, the suggested modifications of InputSplit parameters to the APC program in the HIPI Library yield significant improvements in performance with large datasets. These results prove that the RPi Hadoop cluster lags in performance when compared to Hadoop cluster running on virtual machines using traditional PCs, however the low cost and small form factor makes it ideal for remote Image analysis in surveillance / disaster recovery scenarios where UAVs can transmit image streams to the Cluster for remote processing. We intend to further study the impact of tweaking the performance of various parameters for providing Image-analysis-as-service and object recognition in surveillance applications using the low cost RPi cluster.

Acknowledgements

This work is supported by the DroneMap project entitled DroneMap: A Cloud Robotics System for Unmanned Aerial Vehicles in Surveillance Applications under the grant number 35-157 from King AbdulAziz City for Science and Technology (KACST). This work is partially supported by Prince Sultan University.

References

1. Raspberry pi.
URL https://www.raspberrypi.org/
2. M. A. Laurenzano, A. Tiwari, A. Jundt, J. Peraza, W. A. Ward, R. Campbell, L. Carrington, Euro-Par 2014 Parallel Processing: 20th International Conference, Porto, Portugal, August 25-29, 2014. Proceedings, Springer International Publishing, Cham, 2014, Ch. Characterizing the Performance-Energy Tradeoff of Small ARM Cores in HPC Computation, pp. 124–137. doi:10.1007/978-3-319-09873-9_11.
URL http://dx.doi.org/10.1007/978-3-319-09873-9_11
3. D. Pleiter, M. Richter, Energy efficient high-performance computing using arm cortex-a9 cores, in: Green Computing and Communications (GreenCom), 2012 IEEE International Conference on, 2012, pp. 607–610. doi:10.1109/GreenCom.2012.91.

4. M. F. Cloutier, C. Paradis, V. M. Weaver, Design and analysis of a 32-bit embedded high-performance cluster optimized for energy and performance, in: Proceedings of the 1st International Workshop on Hardware-Software Co-Design for High Performance Computing, Co-HPC '14, IEEE Press, Piscataway, NJ, USA, 2014, pp. 1–8. doi:10.1109/Co-HPC.2014.7.
URL http://dx.doi.org/10.1109/Co-HPC.2014.7
5. Beagle board black.
URL http://beagleboard.org/BLACK
6. P. Abrahamsson, S. Helmer, N. Phaphoom, L. Nicolodi, N. Preda, L. Miori, M. Angriman, J. Rikkila, X. Wang, K. Hamily, S. Bugoloni, Affordable and energy-efficient cloud computing clusters: The bolzano raspberry pi cloud cluster experiment, in: Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on, Vol. 2, 2013, pp. 170–175. doi:10.1109/CloudCom.2013.121.
7. Apache hadoop.
URL https://hadoop.apache.org/
8. S. Ghemawat, H. Gobioff, S.-T. Leung, The google file system, SIGOPS Oper. Syst. Rev. 37 (5) (2003) 29–43. doi:10.1145/1165389.945450.
URL http://doi.acm.org/10.1145/1165389.945450
9. S. Manikandan, S. Ravi, Big data analysis using apache hadoop, in: IT Convergence and Security (ICITCS), 2014 International Conference on, 2014, pp. 1–4. doi:10.1109/ICITCS.2014.7021746.
10. G. Loewen, M. Galloway, S. Vrbsky, On the performance of apache hadoop in a tiny private iaas cloud, in: Information Technology: New Generations (ITNG), 2013 Tenth International Conference on, 2013, pp. 189–195. doi:10.1109/ITNG.2013.32.
11. B. Mathiya, V. Desai, Apache hadoop yarn parameter configuration challenges and optimization, in: Soft-Computing and Networks Security (ICSNS), 2015 International Conference on, 2015, pp. 1–6. doi:10.1109/ICSNS.2015.7292373.
12. F. Xhafa, D. Ramirez, D. Garcia, S. Caballe, Performance evaluation of data mining frameworks in hadoop cluster using virtual campus log files, in: Intelligent Networking and Collaborative Systems (INCOS), 2015 International Conference on, 2015, pp. 217–222. doi:10.1109/INCoS.2015.82.
13. C. Sweeney, L. Liu, S. Arietta, J. Lawrence, Hipi: a hadoop image processing interface for image-based mapreduce tasks, Chris. University of Virginia.
14. H. Tan, L. Chen, An approach for fast and parallel video processing on apache hadoop clusters, in: Multimedia and Expo (ICME), 2014 IEEE International Conference on, 2014, pp. 1–6. doi:10.1109/ICME.2014.6890135.
15. M. Husain, S. Meena, A. Sabarad, H. Hebballi, S. Nagaralli, S. Shetty, Counting occurrences of textual words in lecture video frames using apache hadoop framework, in: Advance Computing Conference (IACC), 2015 IEEE International, 2015, pp. 1144–1147. doi:10.1109/IADCC.2015.7154882.
16. W. Nina, R. Cruz, J. Serrano, J. Cuba, Y. Huaynacho, A. Mamani-Aliaga, Y. Yari, P. Yanyachi, A new approach to the massive processing of satellite images, in: Computing Conference (CLEI), 2015 Latin American, 2015, pp. 1–6. doi:10.1109/CLEI.2015.7359991.
17. X. Zhao, H. Ma, H. Zhang, Y. Tang, Y. Kou, Hvpi: Extending hadoop to support video analytic applications, in: Cloud Computing (CLOUD), 2015 IEEE 8th International Conference on, 2015, pp. 789–796. doi:10.1109/CLOUD.2015.109.
18. F. P. Tso, D. White, S. Jouet, J. Singer, D. Pezaros, The glasgow raspberry pi cloud: A scale model for cloud computing infrastructures, in: Distributed Computing Systems Workshops (ICDCSW), 2013 IEEE 33rd International Conference on, 2013, pp. 108–112. doi:10.1109/ICDCSW.2013.25.
19. M.-E. Nilsback, A. Zisserman, Delving deeper into the whorl of flower segmentation, Image and Vision Computing 28 (6) (2010) 1049 – 1062. doi:http://dx.doi.org/10.1016/j.imavis.2009.10.001.
URL http://www.sciencedirect.com/science/article/pii/S0262885609002145
20. P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, P. Perona, Caltech-ucsd birds 200.