Feasibility of Raspberry Pi 2 based Micro Data Centers in Big Data Applications

Nick Schot
University of Twente
P.O. Box 217, 7500AE Enschede
The Netherlands
n.j.schot@student.utwente.nl

ABSTRACT

Many new data centers have been built in recent years in order to keep up with the rising demand for server capacity. These data centers consume a lot of energy, need a lot of cooling equipment and occupy big stretches of land. Energy efficiency of data centers is becoming an increasingly hot topic. Researchers and companies continuously look for ways to bring down energy consumption.

This paper takes a look at the possibilities and benefits of using cheap, low-power and widely supported hardware in a micro data center with big data as its main focus. For this purpose, the Raspberry Pi 2 Model B was used as a basis. It was analyzed for performance, scalability, energy consumption and manageability in a data center environment.

The result was a fully functional distributed Hadoop setup with a very low power consumption and moderate performance. A highly concurrent, low power setup in a small 1U form factor was proposed as an alternative to traditional rack servers.

Keywords

Micro data center, Raspberry Pi, big data, Hadoop

1. INTRODUCTION

In recent years, many new data centers have been built in order to keep up with the rising demand for server capacity. The servers within these data centers consume a lot of energy, need a lot of cooling equipment and occupy big stretches of land. Currently, one of the biggest problems data centers are facing is ensuring efficient and cheap cooling of all hardware with the lowest possible power usage [7]. The hardware used in data centers is often expensive, power hungry and in need of active cooling. The cost of cooling equipment often matches or exceeds the cost of the server hardware. Companies and researchers are constantly looking for ways to improve the efficiency and cost of data centers. A few low power pioneering clusters [2], [6], [8], [27] based on first generation Raspberry Pi hardware have been built. These resulted in highly concurrent and cheap clusters with low power consumption, but with relatively low performance as the bottleneck.

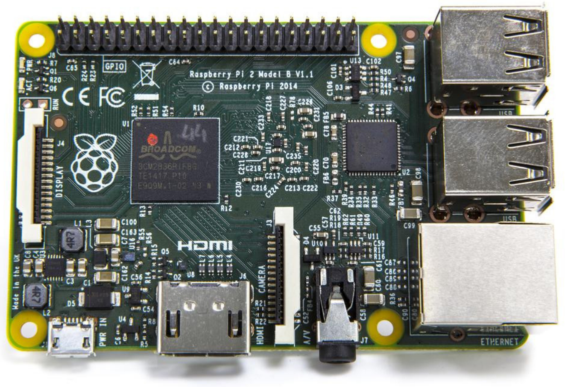

Figure 1. Raspberry Pi 2 Model B

Table 1. Raspberry Pi 2 Model B specifications

System on Chip	Broadcom BCM2836
CPU	900MHz quad-core ARMv7 Cortex-A7
GPU	Dual Core VideoCore IV Multimedia Co-Processor
RAM	1GB LPDDR2 SDRAM
Ethernet	Onboard 10/100 Ethernet RJ45 jack
USB	Four USB 2.0 Connectors
Storage	Micro SD card slot

Recently, the Raspberry Pi 2 Model B (RPi2, see figure 1) has been released [28]. This release gives new opportunities for the Raspberry Pi as the foundation of a feasible micro data center. The performance and energy consumption were improved in the new version while the price remained the same.

According to [5, p.6], the main design criteria of a flexible future proof data center are availability, scalability, security, performance and manageability. Three of those criteria have been tested for in this paper on a small RPi2 cluster: scalability, performance and manageability. For a data center to be scalable, it must support fast and seamless growth without major disruptions. It must also support new services without overhauling the infrastructure of the data center. Finally the data center may not have a single point of failure. [5, p.117]

23^{th} *Twente Student Conference on IT* June 22^{st}, 2015, Enschede, The Netherlands.

In order to find out whether or not the RPi2 is up to the job, the main research question has been defined as: In which ways is a cluster of Raspberry Pi 2 Model B boards a feasible alternative to traditional servers in big data applications? This paper has been split up into several sections defined by the following subquestions which will need to be answered:

- What is the performance and scalability of the Raspberry Pi 2 in a cluster setup?
- What is the energy consumption of a cluster of Raspberry Pi 2 boards?
- How can a cluster of Raspberry Pi's be efficiently put in a traditional rack server?
- How is the performance of the Raspberry Pi 2 Model B cluster when compared to traditional servers?

This paper evaluates the possibilities and benefits of using cheap, low-power and widely supported hardware in a micro data center. It will also propose a way to allocate this hardware in a standard rack server and draw comparisons with an existing data center setup.

2. RELATED WORK

Multiple Raspberry Pi clusters have been built and evaluated for mainly research and educational purposes [2], [6], [8], [27]. Most of them are still based on older Raspberry Pi Model B(+) boards which offer significantly lower performance than the newer RPi2. The creators of these clusters have designed racks and casings ranging from Lego constructions to cases designed by Industrial Engineering students, but these were not specifically meant for usage within data centers where rack servers are the standard.

2.1 Iridis-pi cluster

The Iridis-pi cluster [8] was designed as a portable, low power, affordable, passively cooled cluster aimed at educational applications. It consists of 64 Raspberry Pi Model B boards combined with 16GB SD cards. They installed a traditional Message Passing Interface (MPI) which provides a standard for implementing parallel computing on clusters.

2.2 Glasgow Raspberry Pi cluster

The Glasgow Raspberry Pi cluster [27] consists of 56 Raspberry Pi Model B boards. They developed a custom piece of management software (PiCloud) running on top of LXC virtualized containers. They also experimented with Hadoop and Software Defined Networking.

2.3 Bolzano Raspberry Pi cluster

The Bolzano Raspberry Pi cluster [2] consists of 300 Raspberry Pi Model B boards and was put together as an affordable energy-efficient computing cluster. Applications such as a green research testbed and as a mobile data center are evaluated.

2.4 Raspberry Pi in a server rack

Raspberry Pi Colocation [24] proposed a rack setup with a custom power board in which about 500 Raspberry Pi's would fit in a 42U rack.

3. BACKGROUND

3.1 Hadoop

Apache Hadoop [29] is a well known open source framework which offers the necessary components for the distributed processing of large distributed data using simple programming models like map/reduce. It has been designed to scale well from one to thousands of machines. Hadoop offers high-availability options for detecting and recovering from failures in both hard- and software.

Map/reduce [9] as implemented in Hadoop is a programming model first developed by Google to allow for simple distributed processing of large data sets. A map/reduce program consists of two steps. The map step performs filtering and sorting. The reduce step can then do further computations on the output of the maps, which is usually a summarizing operation. Depending on the program the map and/or reduce tasks can be parallelized.

Hadoop 2 had some big improvements in the map/reduce area [12]. YARN was introduced as a new resource management layer. YARN handles workload management and monitoring, manages high availability features and allows for more programming models next to just map/reduce.

A basic Hadoop installation consists of three main parts: HDFS, YARN and the JobHistoryServer.

HDFS is the Hadoop distributed file system and consists of a couple of processes. The NameNode is the main process which keeps track of where all files are distributed and replicated. It is the main access point for all clients and processes and runs on the master node. The SecondaryNameNode keeps a recent backup of the NameNode so the NameNode can be restored if it might go down. The DataNode processes run on the remaining slave nodes and handle data storage and retrieval.

YARN consists of a ResourceManager, which manages all jobs in the system, and on each slave node a NodeManager. The NodeManager process handles the execution of jobs allocated to a slave node.

Finally, the JobHistoryServer keeps track of all completed jobs and their logs.

3.2 Current server hardware

Big server manufacturers like Dell, HP and SuperMicro offer all kinds of servers aimed at a Hadoop use case. They currently range from thousand dollar rack servers to thousands of dollars per rack server. Most of the time a single 1U rack server has two or more dedicated CPU's and four hard drives.

Hadoop usually runs on a multitude of 1U rack servers containing eight or more storage drives. 1U rack servers are relatively cheap, but bring a lot of space and energy overhead compared to more expensive, but more efficient solutions like blade servers which are usually 10U. Blade servers house vertically placed blades combined with a single power supply and network access for all blades combined. This makes them more space efficient, offering a higher density of servers than traditional 1U servers. Hadoop setups can start with just a single server and be scaled to thousands of servers.

3.3 The Raspberry Pi 2 Model B

The Raspberry Pi 2 Model B [23] is a small, cheap yet feature packed computer. It is based on the Broadcom BCM2836 SoC which offers a 900MHz quad-core ARMv7 CPU combined with 1 GB of RAM and can currently be bought for $35. Detailed specifications can be found in Table 1.

4. SYSTEM DESCRIPTION

In this paper, a RPi2 based cluster is presented as an alternative to the current status quo of multiple 1U rack servers. The goal is to create a cheap, highly concurrent, low power and cheaply cooled alternative to standard rack servers in big data applications. Thus, the hardware must be able to run the Hadoop software library and successfully complete distributed computations with it.

A small proof of concept cluster consisting of eight RPi2 boards was created. A lightweight Debian distribution based on Raspbian [25] was used (DietPi [10]) as the operating system. DietPi was chosen because it is a very lightweight distribution, with only 11 running processes and 12MB of RAM usage after boot. The operating system, applications and storage were all placed on a 16GB UHS-I micro SD card. Since there were only eight boards, the network topology consisted of just a 16-port 100 Mbit switch with a fast enough backplane to handle the full bandwidth of eight RPi2 boards concurrently so that it would not be a bottleneck. A simple router was attached to provide the cluster with internet connectivity. The total cost of used hardware and cables was € 535.

Hadoop 2.6.0 with YARN compiled natively for the RPi2 (ARMv7 hard float) [4], [17] has been configured on top of DietPi. Hadoop was used in conjunction with the ARM HF version of Oracle Java 7 as provided from the Raspbian repositories. Since there are only eight nodes, Hadoop has been configured such that one master node runs the NameNode, Secondary NameNode, ResourceManager and the JobHistoryServer. Each of the seven slave nodes runs a NodeManager and a DataNode. YARN has been configured so that two YARN containers can run concurrently on a single slave node. With seven slave nodes and a little storage overhead this results in about 13 GB of free space on each node which makes the cluster total out at about 91 GB of distributed storage. HDFS has been configured so that two replicas of each file exist in the system. This effectively amounts to about 45 GB of usable storage.

5. RESULTS

5.1 System benchmarks

System benchmarks were done to test basic performance in storage, CPU, memory and networking. Apart from the networking tests, all tests were done on a single RPi2.

5.1.1 SysBench

For basic system benchmarks, the SysBench suite [26] has been used. SysBench tests file I/O performance, scheduler performance, memory allocation and transfer speed, POSIX threads implementation performance and database server performance. It serves as a tool to quickly determine system performance without setting up any complex software. SysBench runs on a big variety of hardware, which makes it ideal for comparison purposes. The file I/O, CPU and memory benchmarks were used to test the RPi2.

The CPU benchmark calculates prime numbers up to a given maximum. The maximum used for the benchmark was 10000. The duration of the benchmarks are noted in Table 2. When comparing these results with the Raspberry Pi Model B(+), the RPi2 is roughly six times faster when running with four threads [19].

Table 2. SysBench CPU

Benchmark	Duration (s)
cpu, 1 thread	442
cpu, 2 threads	224
cpu, 3 threads	149
cpu, 4 threads	112

Table 3. SysBench Storage & Memory

Benchmark	Transfer speed
random read	9.9718 MB/s
random write	1.2604 MB/s
random read/write	3.4046 MB/s
sequential read	17.7400 MB/s
sequential write	6.3972 MB/s
sequential rewrite	13.0000 MB/s
sequential memory read	207.5000 MB/s
sequential memory write	177.0200 MB/s

The SD card storage was tested by running the random and sequential storage tests. 4 GB of test data was prepared with SysBench. The benchmarks were run with a maximum execution time of 300 seconds. The memory test sequentially read and wrote 512 MB of data to memory.

Table 3 shows that the write performance of the SD cards is quite low. Read performance is below what was expected from the SD card, which promised 40 MB/s for sequential read operations but achieved barely half of that speed. The RPi2's memory is sequentially read at 207 MB/s while its write speed is 177 MB.

5.1.2 iperf3

iperf3 [15] has been used to test network performance. The RPi2 uses a 100 Mbit Ethernet connection which is connected to the rest of the system via a combined USB 2.0/Ethernet chip [16]. Due to the overhead caused by this, the Raspberry Pi Model B(+) could not achieve maximum Ethernet performance. With the improved performance of the RPi2 it might have overcome this problem. Furthermore, iperf3 was used to find out whether the network, the storage or the memory is a bottleneck by reading/writing from/to the different mediums [11]. This is important as Hadoop shuffles a large amount of data around the network and might write directly to disk if the data is too big to fit in free memory. To find out if there is a bottleneck, 60 second iperf3 benchmarks were run from memory to memory, memory to disk and from disk to memory.

Table 4. iperf3

Test	Avg bandwidth	Cwnd
Mem -> mem	93.4 Mbit	133 KB
Mem -> disk	24.3 Mbit	133 KB
Disk -> mem	94.2 Mbit	133 KB

From the results in Table 4 it can be concluded that the write performance of the Raspberry Pi 2 and/or the SD card is a bottleneck with only 3 MB/s. This number is in line with the results from the SysBench write tests which were between 1.26 MB/s and 6.4 MB/s for random and sequential writes respectively. The memory to memory and disk to memory results show that the Raspberry Pi can achieve good performance on the 100 Mbit Ethernet port ending up around 94 Mbit. The congestion window size ended up at 133 Kilobyte for every test after the initial slow-start.

Table 5. iperf3 connections

Connections	Avg aggregated bandwidth
4	94.2 Mbit
32	94.7 Mbit
128	96.5 Mbit

The effect of multiple parallel connections was also tested. Results are shown in Table 5. The amount of connections did not affect the average aggregated bandwidth which stayed around 95 Mbit. The RPi2 seems to handle a lot of concurrent connections well.

5.2 Hadoop benchmarks

A selection of Hadoop benchmarks was made to cover the most important aspects of a Hadoop cluster. The benchmarks which were run are part of the HiBench benchmark suite [13], [14] and the standard Hadoop test suite and cover HDFS throughput, CPU bound computation and generic computation on distributed big data.

5.2.1 *DFSIO-E*

DFSIO Enhanced tests the HDFS throughput of a Hadoop cluster by sampling the read and written bytes handled by the map tasks. It generates a defined number of tasks which perform writes and reads simultaneously. The average I/O rate and throughput of each map task are recorded together with the aggregated throughput of the HDFS cluster.

Table 6. DFSIO-E

Type	Throughput	Avg aggregated throughput	Standard deviation
read 1	5.52 MB/s	43.66 MB/s	5.68 MB/s
write 1	2.26 MB/s	17.57 MB/s	9.42 MB/s
read 2	6.69 MB/s	55.35 MB/s	4.44 MB/s
write 2	2.07 MB/s	15.54 MB/s	6.95 MB/s
read 3	6.08 MB/s	39.66 MB/s	7.13 MB/s
write 3	1.94 MB/s	11.64 MB/s	7.49 MB/s

In Table 6 the results of three DFSIO-E runs are shown. The benchmark was configured to use eight files of 1 GB each handled by eight map tasks. Because of YARN scheduling, not all nodes were necessarily used at all times. This means that the average aggregated throughput is not necessarily the actual full throughput the cluster is capable of, explaining the big differences in column 3 of Table 6. On average the cluster had a read throughput of 6.1 MB/s per map task and a write throughput of 2.09 MB/s. The reason these are lower than the SysBench and iperf3 benchmarks is that multiple map tasks were running on a single RPi2. As then two map tasks are reading or writing concurrently to the same SD card, this lowers the average performance per map task.

Yarn会根据任务规模决定使用节点数量

？？？

5.2.2 *TeraSort*

TeraSort is a benchmark which measures sort speed on large distributed files. TeraSort consists of three programs. TeraGen is a map/reduce program which generates the data which is used by TeraSort. The result is a multiple of 100 byte rows. Each map generates a range of 100 byte rows resulting in the requested file size. TeraSort is a standard map/reduce sort program with a custom partitioner. TeraSort forces a replication factor of one on the output files of the reducers, instead of the cluster default. TeraValidate makes sure the output of TeraSort is globally sorted.

For the TeraSort benchmark, different node setups were used. The amount of slots mentioned in Table 7 has to be subtracted by one overhead slot for the application master to get the actual containers available for map and reduce jobs. slot是指container的数量

As there is not a lot of storage available on the cluster, TeraSort was not run with a TeraByte of data but with 1 GB, 7 GB and 10 GB of data generated by TeraGen. The 7 GB number was chosen because there are seven DataNodes available in the cluster. After each run, the data of the previous run was removed.

Table 7. TeraSort

Nodes	5	8	5	8	5	8
Slots	8	14	8	14	8	14
Maps	16	16	64	64	80	80
Reduces	8	8	8	8	8	8
Data (GB)	1	1	7	7	10	10
Total (s)	366	230	3584	1747	-	3041
Avg map (s)	70	72	144	141	-	261
Avg shuffle (s)	70	88	-	698	-	830
Avg reduce (s)	48	49	1741	406	-	550

The results from the TeraSort benchmark can be found in Table 7.

An inherent problem to a smaller cluster showed up in the 7 GB run on five nodes and is caused by one of Hadoops optimizations for bigger clusters. When a map task finishes on a node, Hadoop starts a reduce task on that same node since the necessary data is already there. The nodes are configured to run two concurrent tasks. With seven nodes available, this gives a total of 14 containers of which one is the Application Master. If there is a job with more map tasks than the amount of available containers, part of the tasks will run sequentially. If a container frees up, a new map or reduce task gets assigned to the container.

The problem is that as soon as the first batch of map tasks finishes, reduce tasks get started on the nodes. This leaves only few containers available for the relatively high amount of map tasks which still need to be completed. Meanwhile, the reduce tasks will have a lot of idle time, because more input data from the map tasks becomes available at a very low pace. In this case, adding more nodes would solve this problem as enough slots will be available to allocate enough map jobs to keep everything speedy. This will bring the total running time closer to the average map time. Since the 7 GB run allocated 64 map tasks, and after the first run of 13 maps, some reduces were already started, it took a total of 1747 seconds to complete all jobs. The average reduce time is high mostly because the reduce tasks were waiting for new input data.

hadoop的优化（节点完成map即转为reduce，导致做map的节点减少而reduce的空等待）导致在节点少的情况下浪费资源，完成时间和maap时间相差很大

The shuffle time is the time it takes to get the required data as output by a map task to the correct reducer. As there are usually many more map tasks then there are reduce tasks, this is a vital number for fast Hadoop operations. The reducers were able to retrieve data from other nodes with a reported speed of about 11 MB/s, which is to be expected from a 100 Mbit Ethernet connection. This also means Hadoop is, at least most of the time, writing into memory as iperf3 showed that the write speed to the SD card is much lower over the network.

The last problem showed up for the first time when TeraSort ran with 10GB of data on 5 nodes. It never finished because of the amount of crashing DataNodes and reduce jobs which never completed. If Hadoop assigns two reduce tasks to a single node and those nodes have, relatively speaking, a lot of data to process, the reduce tasks will use too much memory when writing their results to HDFS causing the DataNode process to get kicked out of memory and with that causing the reduce task to fail. Hadoop may then decide to start two copies of the same job to the cluster. This only amplifies the problem with a small cluster, as more reduce tasks are running at the same time, making the chance that two are running on a single node significantly higher. This problem can most likely be solved by changing the YARN configuration so that only one reduce task may run on a single node.

5.2.3 Pi

Pi is a map/reduce program from the default Hadoop examples. It estimates pi using a quasi-Monte Carlo method [20] with a given number of samples and maps. This benchmark is CPU bound. It does not read/write anything to HDFS. Thus it benchmarks the raw compute performance of the RPi2 in a Hadoop environment.

The Pi benchmark was run with a setup of five nodes with eight containers and a setup of eight nodes with 14 containers. Pi was calculated with 1.000.000.000 samples per map. Adding more maps or samples makes the estimation of pi more accurate.

Table 8. Pi

Containers	8	8	14
Maps	6	12	12
Total (s)	996	1975	996
Avg map (s)	976	981	975
Avg shuffle (s)	13	978	13
Avg reduce (s)	2	2	2

From the pi benchmarks as seen in Table 8 it became clear that the average shuffle time also depends on the availability of the data for the reducers. The pi benchmark generates very small intermediate data which, if all maps can be allocated, takes only 13 seconds of shuffle time which is mostly overhead time from Hadoop due to hard coded polling intervals.

The runs with 8 available containers instead of 14 show the impact of a setup where there are fewer available slots than there are maps to be run. As can be seen from the comparison with six maps and 12 maps with enough available nodes, the total duration depends on the speed with which individual maps are finished and not on the amount of maps if enough slots are available.

5.3 Energy consumption and cooling

5.3.1 Energy consumption

The energy consumption of the RPi2 was measured with a simple setup. A prototyping PCB with two USB connectors and some jumper wires was used to allow for a multimeter (Elro M990) to connect for voltage and current measurements of a single RPi2. This way the actual power usage of the RPi2 is measured, because the (in)efficiency of our power supply is not taken into account.

The power consumption was measured under several workloads to find out what effect different kind of operations have on the power consumption of the RPi2. SysBench was used to consistently stress different parts of the board.

Table 9. Power usage

Benchmark	**Current**	**Voltage**	**Wattage**
idle	315mA	4.86V	1.53W
cpu, 1 thread	340mA	4.84V	1.65W
cpu, 2 threads	365mA	4.79V	1.75W
cpu, 3 threads	392mA	4.77V	1.87W
cpu, 4 threads	415mA	4.78V	1.98W
storage write	395mA	4.77V	1.88W
storage read	442mA	4.77V	2.11W
memory	440mA	4.79V	2.11W

As shown in Table 9, the power usage seems to max out at a little over 2 Watt under load. When doing only CPU intensive computation it even stays under 2 Watt, while an idle board uses roughly 1.5 Watt.

5.3.2 Cooling

By default, the RPi2 is a passively cooled board without any heat sink or fan. It has been designed to run cool enough under normal room temperatures.

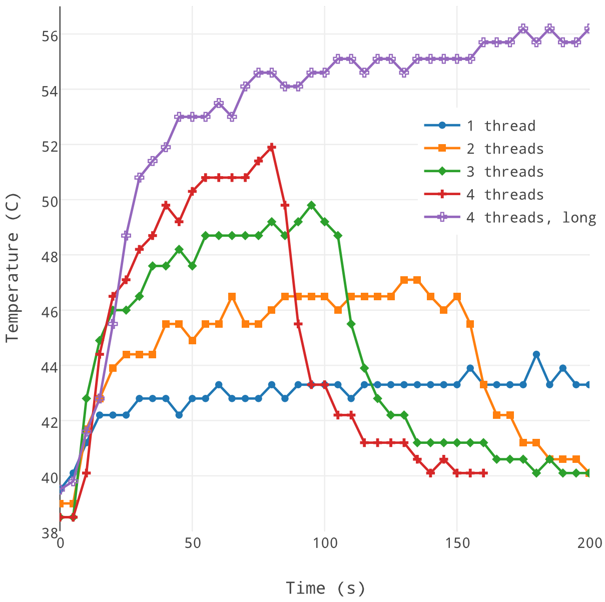

Figure 2. Sysbench CPU temperatures

CPU temperature measurements were taken under SysBench CPU stressing with different numbers of threads. The results are shown in figure 2. The temperature was taken directly from the RPi2 itself. The second run with four threads was configured to run longer as the first run shown finished too quickly for the CPU to warm up. The room in which the measurements were done had a temperature of 21 degrees Celsius. The RPi2 seems to heat up to just above 55 degrees Celsius under full CPU load. With the current Hadoop configuration, the most common workload would be two CPU threads for which the temperature stays around 46 degrees Celsius. This is a good working temperature, considering that those temperatures are idle temperatures for x86 CPU's. As expected, the CPU cools down quickly after the benchmark finished.

5.4 The Raspberry Pi in a rack server

For the RPi2 to be useful in an enterprise environment, it must be able to fit well in standardized server racks. Hardware breaks all the time in data centers, so it should be easily accessible and replaceable. One big disadvantage of current Raspberry Pi models is the placement of the power connector and Ethernet connector. The connectors are placed perpendicular to each other which makes it harder to create a compact easy to manage way to place the boards in a confined space.

Most standard data center rack consists of 42U of space. As defined by the EIA-310 standard a single U is 44.50mm high [1]. A 1U rack's inside dimensions are defined to be 450mm wide, 44.43mm high and at most 739.775mm deep [22]. Each 1 or more U server is usually 0.787 mm less high than the EIA-310 standard defines to allow for easy removal of servers without friction from other servers [21].

The RPi2 is 85.60mm wide, 56mm deep and 21mm high. It has four standard mounting holes for screws or spacers to fit through.

The rack must, next to RPi2 boards, contain a power supply with sufficient ports and power to handle all boards under full load. The casing must also contain a couple of fans to generate some airflow. The power supply has an estimated size of 100mm wide, 40mm high and between 190-250mm deep. when looking at standard 1U power supplies. 5V DC power supplies in this form factor are readily available with sufficient output power.

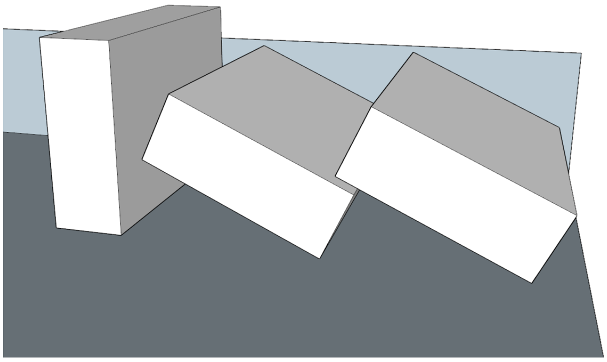

Figure 3. Vertical and tilted RPi2 in a 1U server

The most efficient way to place the RPi2 in a small contained space is to tilt it 90 degrees on the long edge with the power connector facing downwards. This way it can be connected to power on the bottom of the rack, and to Ethernet on the side, which would allow for the easiest access to a single board. Unfortunately, as can be seen in Figure 3, a tilted RPi2 is just a little higher than a standard U, so a bigger 1.5U rack should be used to achieve this option.

The second and easiest option is to make stacks of two RPi2 boards with brass spacers in between. As there is 44.43mm of inside vertical space and a single RPi2 is 21mm high, a stack of two RPi2's fits perfectly in the available space. By extending the spacers on the bottom, the tiny stack can also be easily secured on the bottom of the rack server. The only downside to this approach is the accessibility of the RPi2 boards. The board on the bottom is a little harder to access as either the top one or both RPi2 boards have to be removed. If the maximum rack depth is used, 12 RPi2 boards fit next to each other. This gives 24 boards for a single row. Keeping space for all cables and connectors, 4 rows fit in the width of a rack server. This gives a total of 96 boards for a 1U rack server. As this is excluding the power supply, 16 to 24 RPi2 boards will need to be removed to make enough room. The estimated amount of RPi2 boards is then 72-80 for a 1U rack with a power supply. This amount of boards does result in a big amount of cables in a small compartment. For better manageability some more RPi2 boards need to be removed. The resulting layout with 80 RPi2 boards and a power supply in the back can be seen in Figure 4.

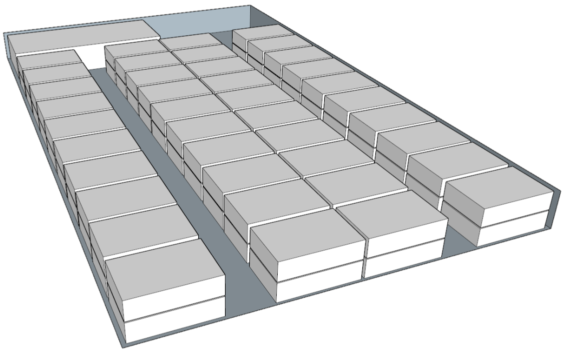

Figure 4. Proposed RPi2 rack layout

Another option is to still put the RPi2's on their side, but tilted so they still fit in a 1U rack. This way all RPi2 boards are easily removable and space is still used efficiently as the boards can overlap each other. The down side is that the case in which they will have to reside will need special triangular spacers on the bottom in order to secure the boards to the case. The maximum height of the tilted boards should be 40mm so a little room is left for the power connector. This gives a required angle of 25 degrees making the board 55mm wide horizontally. The effective angle and overlap is shown in Figure 3. Because of the low angle, practically no overlap between the RPi2's can exist. This removes the main advantage of this option, as it is easier to put the RPi2's horizontally in the rack.

A 1U switch can house a maximum of 48 Ethernet ports. This means a 2U switch will be needed to provide a 1U rack of RPi2 boards with enough Ethernet connections. Due to the high amount of connections required in the case and the limited amount of available space, flat Ethernet cables are likely needed.

5.5 In comparison

At the University of Twente resides the CTIT cluster. Hadoop runs on 32 Dell R415 servers. Some of the Hadoop benchmarks that ran on the RPi2 were run on the CTIT cluster too for comparison purposes. As the RPi2 cluster contains only eight nodes at this time, a comparison can be drawn by looking at the average duration of maps and reduces. If there were more nodes available, more maps and reduces can run concurrently and thus complete the task faster.

The CTIT cluster has far more container slots than the RPi2 cluster. This is why the slots are not mentioned in Table 10 and Table 11 as they are not relevant for this benchmark. Enough slots were available to allocate all map/reduces at once.

Table 10. TeraSort (CTIT)

Maps	16	64	80
Reduces	8	8	8
Data (GB)	1	7	10
Total (s)	22	49	67
Avg map (s)	7	10	11
Avg shuffle (s)	4	19	24
Avg reduce (s)	2	15	21

Table 11. Pi (CTIT)

Maps	6	12
Total (s)	40	40
Avg map (s)	32	32
Avg shuffle (s)	3	3
Avg reduce (s)	0	0

Table 7 and Table 10 show that the CTIT cluster is roughly ten times faster when sorting 1 GB of data. Three of the map tasks had to wait until the rest was complete, so the actual time is likely smaller. The average map task also took roughly ten times longer on the RPi2 cluster. The runs with more data were a lot slower on the RPi2 cluster because not enough container slots were available to run all map tasks simultaneously. The average map took 24 times longer on the RPi2 cluster when sorting 10 GB of data. This higher ratio could be the result of the low write speed to the SD card when more data has to be handled.

10倍

当数据量大时，有20倍

The results in Table 8 and Table 11 show that the amount of maps does not influence running time for the pi benchmark if enough container slots are available. Thus we can directly compare the results between the two systems. The CTIT cluster took 40 seconds to complete the benchmark with an average map time of 32 seconds. In comparison the RPi2 cluster took 996 seconds to complete with an average map time of 975 seconds. From this we can than calculate that, for this CPU bound single thread per map benchmark, the processing cores in the CTIT cluster are roughly 30 times faster than the processing cores from the RPi2.

计算能力相差30倍

The Dell R415 server used by the CTIT cluster contains two AMD Opteron 4386 processors [3] which each have a TDP of 95 Watt. The R415 contains a 480 Watt power supply [18]. Included are four hard disks for storage which consume about 8 Watt each. Under full load this means the system uses roughly 250 Watt including some overhead for the motherboard and RAM. With this power usage, about 110 RPi2's could be placed while having roughly the same maximum power consumption.

A RPi2 cluster cannot necessarily achieve the same performance as the CTIT cluster by adding more boards. It depends on the specific map/reduce program whether it can be parallelized enough to achieve a similar performance. When averaging the found ratios to 20, 140 RPi2 boards should achieve performance much closer to the CTIT cluster.

6. CONCLUSION

The Raspberry Pi 2 Model B was successfully used to create a fully functional distributed Hadoop setup. A highly concurrent, low power setup in a small 1U form factor was proposed as an alternative to traditional rack servers.

The RPi2 setup delivered moderate performance in combination with Hadoop. The biggest bottleneck in the current setup seemed to be the SD card (reader) and more specifically the random write performance; which is extremely low at 1.26 MB/s. The rated throughput of 100 Mbit for the Ethernet connection was easily achieved even with a lot of concurrent connections. The scalability of the RPi2 in a cluster setup is mostly dependent on the assigned task. As Hadoop has been run on thousands of nodes, the scalability of the soft- and hardware should not be a problem. Programs with bigger map/reduce jobs like TeraSort ran only 10 to 24 times slower than on the CTIT cluster. However, single core performance stays behind being about 30 times slower than the CTIT cluster in the pi benchmark.

The RPi2 requires very little power while operating under load. A single RPi2 consumes only 2 Watts when under full load and thus also remains quite cool at 55 degrees Celsius. Combined with the estimated amount of 72-80 RPi2's in a 1U rack, this results in a highly concurrent 1U rack server while using only 160 Watts under full load. The manageability suffers from the amount of required Ethernet and power cables and the need of a 2U switch to connect all RPi2 boards.

7. FUTURE WORK

7.1 Zookeeper

In addition to the few built-in high-availability options from Hadoop, there is the Apache Zookeeper project which provides a distributed configuration service, synchronization service, and naming registry for distributed systems. Zookeeper also provides lower level high-availability support to fill in the gaps left by Hadoop. An example of this is the master/NameNode. The NameNode does not get automatically restarted or reassigned to another node even though there is a backup available on the Secondary-Namenode. Zookeeper would be a logical addition when building and testing any Hadoop based or other distributed computing clusters.

7.2 Network and/or USB storage

The SD-card turned out to be a big bottle neck in writing performance, other storage mediums could be tested. USB storage is the next logical step. Performance should be a lot better. The theoretical bandwidth limit of USB 2.0 is 480 Mbit/s. Final performance will mostly be dependent on the performance of the used USB stick, SSD or hard drive.

Another possible addition would be to uses distinct DataNodes combined with SD or USB storage, though this would dramatically increase network traffic as data will always have to be retrieved from another node than the one which needs it.

7.3 Compute module

The Raspberry Pi Compute Module is a stripped down Raspberry Pi Model B in a SODIMM form factor. If a new compute module based on the RPi2 were to be released, this could be used in conjunction with a custom motherboard to greatly increase the amount of RPi2's which can be placed inside a rack server. As direct access to the SoC is given, it would also mean the possibility of adding faster storage controllers with improved throughput as a result.

8. REFERENCES

[1] *19-inch rack (EIA-310).* [Online]. Available: https://www.server-racks.com/eia-310.html (visited on 03/23/2015).

[2] P. Abrahamsson, S. Helmer, N. Phaphoom, L. Nicolodi, N. Preda, L. Miori, M. Angriman, J. Rikkila, X. Wang, K. Hamily, and S. Bugoloni, "Affordable and Energy-Efficient Cloud Computing Clusters: The Bolzano Raspberry Pi Cloud Cluster Experiment," in *2013 IEEE 5th International Conference on Cloud Computing Technology and Science*, vol. 2, IEEE, Dec. 2013, pp. 170–175, ISBN: 978-0-7695-5095-4. DOI: 10.1109/CloudCom.2013.121.

[3] *AMD Opteron 4300 Series Processor.* [Online]. Available: http://products.amd.com/en-us/OpteronCPUDetail.aspx?id=825 (visited on 06/01/2015).

[4] *Apache Hadoop 2.6.0 - Native Libraries Guide.* [Online]. Available: http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/NativeLibraries.html (visited on 04/21/2015).

[5] M. Arregoces, M. Portolani, and C. Press, *Data Center Fundamentals.* 2003, ISBN: 1587050234.

[6] J. Bausch, *Supercomputer built using Raspberry Pi and Legos*, 2012.

[7] *Center of Expertise for Energy Efficiency in Data Centers.* [Online]. Available: https://datacenters.lbl.gov/ (visited on 03/31/2015).

[8] S. J. Cox, J. T. Cox, R. P. Boardman, S. J. Johnston, M. Scott, and N. S. O'Brien, "Iridis-pi: a low-cost, compact demonstration cluster," *Cluster Computing*, vol. 17, no. 2, pp. 349–358, Jun. 2013, ISSN: 1386-7857. DOI: 10.1007/s10586-013-0282-7.

[9] J. Dean and S. Ghemawat, "MapReduce," *Communications of the ACM*, vol. 51, no. 1, p. 107, Jan. 2008, ISSN: 00010782. DOI: 10.1145/1327452.1327492.

[10] *DietPi.* [Online]. Available: https://github.com/Fourdee/DietPi.

[11] *Disk Testing using iperf3.* [Online]. Available: http://fasterdata.es.net/performance-testing/network-troubleshooting-tools/iperf-and-iperf3/disk-testing-using-iperf/ (visited on 06/04/2015).

[12] *Hadoop 2 vs. Hadoop 1: Understanding HDFS and YARN.* [Online]. Available: http://www.tomsitpro.com/articles/hadoop-2-vs-1,2-718.html (visited on 05/30/2015).

[13] *HiBench: A Representative and Comprehensive Hadoop Benchmark Suite.* [Online]. Available: https://software.intel.com/sites/default/files/blog/329037/hibench-wbdb2012-updated.pdf (visited on 05/06/2015).

[14] *HiBench - the Hadoop Benchmark Suite.* [Online]. Available: https://github.com/intel-hadoop/Hibench (visited on 03/23/2015).

[15] *Iperf/iperf3.* [Online]. Available: http://fasterdata.es.net/performance-testing/network-troubleshooting-tools/iperf-and-iperf3/ (visited on 06/04/2015).

[16] *LAN9514-JZX.* [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/9514.pdf (visited on 06/04/2015).

[17] *Native Hadoop 2.6.0 Build on Pi.* [Online]. Available: http://www.instructables.com/id/Native-Hadoop-260-Build-on-Pi/ (visited on 04/21/2015).

[18] *PowerEdge R415 1U Rack Server.* [Online]. Available: http://www.dell.com/us/business/p/poweredge-r415/pd (visited on 03/31/2015).

[19] *Putting the New Raspberry Pi 2 to the Test | element14.com.* [Online]. Available: http://www.element14.com/community/docs/DOC-73977 (visited on 06/04/2015).

[20] *Quasi-Monte Carlo Sampling.* [Online]. Available: http://statweb.stanford.edu/~owen/reports/siggraph03.pdf (visited on 05/26/2015).

[21] *Rack Mount Servers.* [Online]. Available: https://eracks.com/rackmount-servers/ (visited on 06/01/2015).

[22] *Rack Mounting Depth.* [Online]. Available: https://www.server-racks.com/rack-mount-depth.html (visited on 06/01/2015).

[23] *Raspberry Pi 2 Model B*, 2015. [Online]. Available: http://www.raspberrypi.org/products/raspberry-pi-2-model-b/ (visited on 03/22/2015).

[24] *Raspberry Pi colocation.* [Online]. Available: https://www.raspberrypi.org/raspberry-pi-colocation/ (visited on 06/01/2015).

[25] *Raspbian.* [Online]. Available: https://www.raspbian.org/ (visited on 06/01/2015).

[26] *SysBench benchmark suite.* [Online]. Available: https://github.com/akopytov/sysbench (visited on 03/23/2015).

[27] F. P. Tso, D. R. White, S. Jouet, J. Singer, and D. P. Pezaros, "The Glasgow Raspberry Pi Cloud: A Scale Model for Cloud Computing Infrastructures," in *2013 IEEE 33rd International Conference on Distributed Computing Systems Workshops*, IEEE, Jul. 2013, pp. 108–112, ISBN: 978-1-4799-3248-1. DOI: 10.1109/ICDCSW.2013.25.

[28] E. Upton, *Raspberry Pi 2 on sale now at $35*, 2015. [Online]. Available: http://www.raspberrypi.org/raspberry-pi-2-on-sale/ (visited on 03/22/2015).

[29] *Welcome to Apache™ Hadoop®!* [Online]. Available: https://hadoop.apache.org/ (visited on 03/31/2015).

Appendix A.

How to: install & configure native Hadoop on the Raspberry Pi 2 Model B

This guide mostly contains shell commands and tasks in order to successfully install and configure a basic Hadoop 2.6.0 YARN on a cluster of Raspberry Pi 2 Model B boards.

Hadoop 2.6.0 compiled natively for the Raspberry Pi 2 and the required configuration files for a cluster setup can be found on: https://github.com/nickschot/dacs-pi

Prerequisites for native Hadoop

```
apt-get install libssl-dev libsnappy-dev
```

Basic Hadoop installation

Create a Hadoop user

```
addgroup hadoop
adduser --ingroup hadoop hduser
adduser hduser sudo
```

Setup SSH for Hadoop

```
su - hduser
ssh-keygen -t rsa -P ""
cat ~/.ssh/id_rsa.pub >>~/.ssh/authorized_keys
chmod 700 ~/.ssh
chmod 600 ~/.ssh/authorized_keys
```

Unpack Hadoop

```
sudo tar -vxzf Hadoop-260.tar.gz -C /usr/local
cd /usr/local
sudo mv hadoop-260 hadoop
sudo chown -R hduser:hadoop hadoop
```

Add Hadoop environment variables

Add the following lines to `/home/hduser/.bashrc`

```
export HADOOP_INSTALL=/usr/local/hadoop
export PATH=$PATH:$HADOOP_INSTALL/bin
```

Reboot and test

```
sudo reboot
hadoop version
```

Basic Hadoop configuration

The basic Hadoop configuration should happen on each node in the cluster. Preconfigured configuration and environment files are supplied in the configuration folder in the repository.

Configure environment

```
nano /usr/local/hadoop/etc/hadoop/hadoop-env.sh
```

Set: `export HADOOP_HEAPSIZE=768`

```
nano /usr/local/hadoop/etc/hadoop/yarn-env.sh
```

Set: ``` `YARN_HEAPSIZE=768` ```

Configuration files

Edit `core-site.xml`, `mapred-site.xml`, `hdfs-site.xml` and `yarn-site.xml` (see configuration folder in the repository for preconfigured files).

Basic folder setup

Create tmp folder

```
sudo mkdir -p /fs/hadoop/tmp
sudo chown hduser:hadoop /fs/hadoop/tmp
sudo chmod 750 /fs/hadoop/tmp/
```

Format working directory

```
/usr/local/hadoop/bin/hadoop namenode -format
```

Starting Hadoop

```
/usr/local/hadoop/sbin/start-dfs.sh
/usr/local/hadoop/sbin/start-yarn.sh
/usr/local/hadoop/sbin/mr-jobhistory-daemon.sh
start historyserver
```

Go to http://IP_ADDR:50070 to see the Hadoop status page.

Going multinode

Give each node a hostname in `/etc/hostname` (eg. node01, node02 etc.)
Comment the line containing 127.0.1.1 with a # in `/etc/hosts`

Add the hostname of the node which should be secondary namenode to masters configuration file in `/usr/local/hadoop/etc/hadoop/masters` on the master node (one per line).

Add the hostnames of nodes which should be data/task nodes to the slaves configuration file in `/usr/local/hadoop/etc/hadoop/slaves` on the master node (one per line).

Reboot the nodes: `sudo reboot`

Make sure `/fs/hadoop/tmp` is empty on all nodes and reformat the datanodes from the master node with `hdfs namenode -format`

Now start Hadoop from the master node and check the status of the nodes on http://IP_ADDR:50070

Dynamically adding slave nodes

Make sure the new node(s) are configured the same as the other nodes (up until the namenode format). First, add the new nodes hostname to the slaves file on the master node. Then start the datanode by running the following command on the new node: `/usr/local/hadoop/sbin/hadoop-daemon.sh start datanode`

Check if it's correctly added to the cluster on the Hadoop status page. After that start the Yarn nodemanager by executing: `/usr/local/hadoop/sbin/yarn-daemon.sh start nodemanager`