ARM Wrestling with x64: A Study of Web and Terasort Workloads

Sarthak Sharma[1,*] and Yogesh Simmhan[2]
[1]Department of Computer Science and Engineering, Birla Institute of Technology (BIT), Mesra, India
[2]Department of Computational and Data Sciences, Indian Institute of Science, Bangalore 560012, India
Email: sarthakaqua96@gmail.com, simmhan@cds.iisc.ac.in

***Abstract*—In the last decade, ARM processors have dominated the mobile device market due to their favorable computing to energy ratio. In this age of Cloud data centers and Big Data analytics, the focus is increasingly on power efficient processing, rather than just high performance computing. The ARM architecture has started to make in-roads into server-grade processors, with AMD recently releasing the A1100-series processor based on a 64-bit ARM architecture. In this paper, we study the effectiveness of the ARM64 architecture in competing with the entrenched x64 architecture for web and Big Data applications in the data center. Specifically, we study the performance of these two processors on comparable servers configurations to run a web server and a Terasort benchmark using Apache HTTPD Service and Apache Hadoop platform, respectively. These early results indicate the ARM64 server performs comparably with the x64 server for small and medium workloads, but is unable to keep up for larger workloads, while having half the rated power envelope.**

I. Introduction

Mobile and Cloud computing have transformed computing in the 21^{st} century, with millions of servers currently hosted at Cloud data centers and billions of smart phones in the hands of consumers. At the same time, these two classes of computing devices have been supported by two different categories of processor architectures. x64 processors (also called x86-64) have traditionally held sway over Cloud servers, with Intel and AMD offering 64-bit versions of their x86 instruction sets based on a CISC architecture, and with support for hardware virtualization. ARM's RISC-based processor architectures, on the other hand, have dominated mobile platforms, including smart phones, tablets, and embedded Internet of Things (IoT) devices, typically running a 32-bit instruction set.

The RISC architecture natively offers a lower power envelope relative to CISC processors while having more limited performance. At the same time, x64 processors have been increasing the number of cores per processor to overcome the power-wall that limits their growth in single-core clock speeds. As a result, the performance difference between individual ARM and x64 cores has been narrowing.

ARM recently introduced their 64-bit instruction set, ARM64 (also called AArch64), to increase the memory addressing available to their processors. In January 2016, AMD released the first ARM64 System on Chip (SoC), the A1100 series processor with 8-cores (code named *Seattle*), using ARM's Cortex-A57 micro-architecture. Since energy consumption by servers forms a big fraction of the operational cost for Cloud data centers, ARM64 with its lower energy footprint and server-grade memory addressing has started to become a viable platform for servers on the Cloud. This is particularly compelling given the scale-out (rather than scale-up) workloads common to Cloud applications, and the growing trend of containerization as opposed to virtualization.

However, given their recent emergence, there has been limited empirical study of the effectiveness of ARM64 processors for Cloud workloads to confirm these possibilities. In this paper, we study the performance of two servers with similar configurations but running x64 and ARM64 processors, for a web server and a Big Data workload. These are representative Cloud applications running on Apache HTTPD and Apache Hadoop platforms respectively. The web server is tested using the Apache Bench workload while Hadoop is benchmarked using the Terasort workload. These offer early insights into the impact of ARM servers within Cloud data centers to support Big Data applications. *To our knowledge, this is the first paper to evaluate the AMD A1100 64-bit ARM server for data-intensive workloads.*

II. Related Work

ARM32 processors were designed for mobile and embedded devices, such as smart phones and Raspberry Pis, and have also been used in network switches in the data center. Their focus has traditionally been on power efficiency, as opposed to server-grade processors that focus on performance. There have been a few studies of ARM32 processors, and their comparison with x86 or x64 processors as we discuss below.

[1] uses a low cost testing system to systematically compare several ARM and x86 devices. It analyzes the system's power efficiency and performance for a web server, database server and floating point computation, thereby comparing the CPU, power utilization and temperature. The results conclude that ARM proves to be 3 to 4 times more power efficient when it comes to the performance per energy comparison although its performance starts to deteriorate once we increase the size of the workload compared to the x86 CPU which performs consistently. It however does not evaluate the performance of the ARM processors for Big Data workloads, nor consider ARM64, both of which we address.

*Based on work done as an intern at the DREAM:Lab, CDS, IISc

The authors of [2] run various Query processing (TPC-C and TPC-H benchmarks on MySQL configurations) and Big Data (K-means, TeraSort and WordCount, among others) benchmarks on the ARM Cortex-A7 little/ARM Cortex-A15 big cores, and Intel Xeon server systems. They evaluate execution time, energy usage and total cost of running the workloads on these self-hosted ARM and Xeon nodes. Their results show that ARM takes more time to perform MapReduce computations (e.g., Pi estimator) compared to others, when implemented in Java. However, this time is reduced significantly by the C++ implementation of the same.

HPC workloads have been verified for the ARm architecture we well. [3] evaluates the performance and energy on various HPC platforms like CoolEmAll RECS, which consists of 18 nodes based on Intel Core i7-3615QE, Intel Atom N2600 or AMD G-T40N processors, Bull BCS, featuring Intel Xeon E7 processors, and Boston Viridis with ARM Cortex A9 processors. They perform benchmarks like Phoronix Test Suite, CoreMark, Fhourstones, High-Performance Linpack along with a few others. Similarly, [4] also focuses on HPC workloads and characterizes the performance and energy of ARM and x86 platforms for various applications from the HPC domain. The benchmarks are characterized in terms of their performance and power on several ARMv7 (32-bit) and x86 processors. Their results show that the performance to energy efficiency ratio of the ARM system varies by up to an order-of-magnitude according to the characteristics of the application which is being run on the machine. HPC workloads often require strong scaling and scale up, rather than Big Data applications that require scale-out, and is our target domain. As before, the performance analysis of ARM64 servers is not considered by them either.

The energy saving potential of the ARM Cortex-A9 MP-Core 32 bit processor has earlier been compared to conventional Intel Xeon processors by performing a web server benchmarking [10], using autobench, on Apache http web servers. ARM proves to be highly energy efficient with the Dual Core Cortex-A9 having up to 11.1 times greater efficiency when compared to the E5430, while running the HTTP server enabling total of 12.7 in cost saving.

In summary, the novelty our novelty of our work lies in our study of ARM64 processors, which is a recent release compared to the ARM32 variant, and also our focus on data intensive workloads representing data center applications that this processor can enable.

III. System Model

The ARM Cortex-A57 is the first 64-bit server-grade processor design to implement the ARMv8-A architecture, with support for Symmetrical Multiprocessing (SMP) and an out-of-order superscalar pipeline [1]. AMD's A1100 processor series is the first commercially available chip implementing ARM Cortex-A57, released in Jan, 2016 [2]. For our evaluation, we use SoftIron's Overdrive 3000 server, which is an Enterprise-class developer system that based on AMD A1170 processor having * cores at 2 GHz, 16 GB RAM, 1 TB HDD an Gigabit Ethernet. The server comes natively installed with OpenSUSE Linux distribution, Apache web server and OpenJDK 64-bit ARM edition that are used by our workloads.

The x64 server used in our evaluation is a single node in a cluster that is similarly configured, with an AMD Opteron 3380 processor with 8 cores rated at 2.6 GHz, a higher RAM capacity of 64GB, 256 GB SSD and Gigabit Ethernet. It runs CentOS 7 Linux distribution as well. These are listed in the Table I below.

Table I: Configuration of ARM64 and x64 Nodes

	ARM64	**x64**
CPU	1 AMD Opteron A1170 processor, 8× ARM64 A57 cores, 2.0 GHz	AMD Opteron 3380 processor, 8× x64 cores, 2.6 GHz
Memory	2 × 8 GB DDR3 RAM	8 × 8 GB DDR3 RAM
L2/L3 Cache	4MB/8MB	8MB/8MB
Rated TDP	32W	65W
Disk Storage	1 TB (HDD)	256 GB (SSD)
Swap	512 MB	16 GB
OS	OpenSUSE Tumbleweed	CentOS 7
Linux Version	4.5.4-1-default	3.10.0-229.1.2

IV. Web and Big Data Workloads

We consider two categories of workloads in this study: (1) web requests served by Apache HTTP server, and (2) TeraSort using Apache Hadoop. The first workload is a transactional benchmark that evaluates the ability of the servers to handle thousands of concurrent web requests by clients over the network. The TeraSort workload addresses processing of large data volumes using the MapReduce programming model popular for Big Data applications. These workloads are representative of Cloud data center applications and help evaluate the relative merits of the ARM and x64 servers for data intensive applications, albeit not in a fully distributed setting. We next we describe the setup of these Big Data workloads, and analyze their results in the next section.

A. ApacheBench HTTP Server

ApacheBench [12] is a benchmarking tool for HTTP web servers. It creates synthetic client requests for a HTTP web server based on specified parameters, and helps benchmark the performance of the web server. ApacheBench allows us to conduct both *performance* and *load testing* of the server – the former informs us of the latency of client requests under different workloads, while the latter tells us how far we can push the system in terms of requests per second [**?**]. While it was originally designed to test the Apache HTTP web server, the fact that it generates transactional requests that are sent to a given HTTP URL means that it is compatible with other HTTP servers as well.

We use ApacheBench to generate transactional workloads that can benchmark Apache HTTP Web Servers running on the ARM64 and x64 servers. This workload evaluates

[1] https://developer.arm.com/products/processors/cortex-a/cortex-a57

[2] www.amd.com/en-us/products/server/opteron-a-series

their ability to handle 1000's of concurrent connections, and small requests which need rapid response. In that sense, this mimics applications with high data *velocity* and low latency requirements, which is one of the three V's of Big Data [**?**].

ApacheBench offers various configuration parameters [13], including PUT and POST operations, basic authentication, SSL certificates, custom headers, and so on. However, we limit our workload to performing a fixed number of HTTP GET requests from multiple concurrent users. The ApacheBench tool itself runs on a single system thread and simulates the behavior of concurrent client requests. A sample commandline request is shown below:

```
ab -r -n 10000 -c 10 -v 1 -k -H
"Accept-Encoding: gzip, deflate"
  http://www.arm.example.com/
```

ApacheBench (`ab`) is configured to request an HTML page that is 256 bytes from the Apache HTTPD servers running on ARM and x64 servers respectively. We run `ab` from the head node of our cluster that has dual 16 core AMD Opteron 6376 processor at 2.3 GHz, 128 GB RAM and Gigabit Ethernet, whose high resource capacity that ensures that the client does not become the bottleneck. Experiments are performed by varying the numbers of concurrent clients simulated by `ab` per run between 1 to $20,000$, with each client requesting the same webpage 100 times in sequence. Thus, this simulates between 100 to $2,000,000$ webpage requests in each experiment run. $20,000$ was the upper-bound of concurrent clients which the `ab` tool could support.

B. TeraSort on Apache Hadoop

Sorting benchmarks have been widely used to evaluate relational databases [14] and Big Data platforms that deal with large data volumes [**?**]. Typically, they evaluate the time taken by the system under test to sort a large input corpus of random numbers or text. *TeraSort* [15], [16] is one such tool that is used to evaluate platforms like Apache Hadoop and Apache Spark. We use the TeraSort implementation that is shipped with the Apache Hadoop MapReduce v2.6.4 framework [**?**] to evaluate the performance of Hadoop on the ARM64 and x64 servers, and this captures the *volume* dimension of Big Data.

We deploy Apache Hadoop v2.6.4 (pseudo-distributed version) in a single-node cluster setup on the ARM and the x86 servers, with the Hadoop Distributed File System (HDFS) configured to have a $3\times$ replication factor. We run the *TeraGen* input generator which is a Map-only job that generates a given number of random rows of data. The field to be sorted in each row has a range of 10 bytes, while additional fillers are used to bring the length of each row to 100 bytes. This corpus is generated *a priori* and stored onto HDFS present in the two servers.

The TeraSort MapReduce job runs on each server and sorts the input data as follows. For each input row, the Map method emits the bytes to be sorted as the key of its output. This is passed to a custom partitioner that samples and partitions the value range into as many buckets as the number of Reducers. The framework then implicitly sorts each partition as part of the Shuffle and Sort phase of MapReduce. The Reducer then stores the sorted keys it receives back to HDFS. A separate MapReduce job, *TeraValidate*, checks if the output of the TeraSort application is indeed sorted.

MapReduce framework is configured with 3,900 MB of memory for each Map and Reduce task, one virtual core assigned to each Map and Reduce task, and the heap size for the child JVMs set to 3,500 MB. 1,000 MB of memory is given to the AppMaster. Each worker in the above configuration uses 3900 MB memory with 2 Cores. Hence 4 such workers run on the ARM and the x64 servers considering that both have 8 cores each, and the ARM server only has 16 GB of RAM. Also, the ARM server has a 1TB hard disk whereas the x64 server has a 256GB SSD. Hence, the size of the data to be sorted was limited by the capacity of the smaller disk, when we consider the $3\times$ replication factor of HDFS. We varied the size of the input from 10^7 bytes (0.01 GB) to 5×10^{10} bytes (50 GB).

V. Results

A. ApacheBench HTTP Server

For each experiment run of `ab` on the ARM64 and x64 servers with different number of concurrent clients, we monitor and plot the following: (a) the client throughput (total requests per second), (b) the total time to complete all requests by the client, (d) the number of faults or failed requests by the client, and (d) the percentage CPU and memory used on each server.

The requests per second (Fig. 1a) shows a sweetspot for ARM64 and x64 with 10 users, each sending 100 requests, and cumulatively supporting requests at the rate of $30,000$ /sec. This is likely because of too few requests (1000) to offer a meaningful measurement or some cache effect. However, as we increase the number of users beyond that to 100, there is a sharp drop in the rate supported at about 2000 /sec, and a growth in the rate as the number of users increase. x64 is able to grow to a HTTP request rate of about $19,000$ /sec for 10,000 users and flatten out. The ARM server matches the x64's rate until 100 users, after which it has a slower request rate. However, even with 20,000 users, it achieves 70% of the rate supported by x64 at $13,000$ /sec.

The corresponding plot with the end to end time taken by each experiment (solid line) is shown in Fig. 1b. The dashed line depicts an idealized linear plot between the time taken and the number of users. As the number of users grow exponentially on the X axis, we see that ARM64 takes $1.5 - 2.0\times$ longer to complete the run compared to x64. ARM64 is closer to the linear relation in terms of the number of users and the total time taken, whereas the x64 server can be seen to grow marginally super-linearly with more users.

In Figs. 1c and 1d, the CPU utilization for the servers grow as the number of users rise, and it saturates at about 40% median and 80% peak usage for both servers, though x64 shows a slightly higher usage. The memory % however is

modest at under 20%, indicating that this is a compute-bound workload.

`ab` also reports the number of requests that failed, possibly due to timeout of the server forcibly closing connections to save resources. While this grows marginally with the number of users. This load-testing shows that ARM and x64 support comparable loads with below 0.5% faults until 10,000 users, but ARM's faults spike at 20,000 users to 2.5%.

B. TeraSort on Apache Hadoop

We measure and plot the time taken for the TeraSort MapReduce job and the CPU and Memory usage for each experiment. Fig. 2a shows the runtime for TeraSort to execute for data sizes ranging from 0.01 GB–50 GB. It is evident from that the performance of both these nodes is comparable for smaller data sizes till about 1 GB but the difference starts to appear for larger data sets. ARM takes nearly twice as long to sort 5 GB of data and $10\times$ slower for 50 GB of data than x64 server. There are several possible reasons for this out-performance by the x64 server. The CPU usage of the x64 server (Fig. 2c) is close to 100% for data sizes of 5 GB and larger, indicating CPU is the limiting factor while ARM (Fig. 2b) does not go beyond a median of 65% indicating other bottlenecks. The x64 server uses an SSD storage while ARM has a HDD storage, which can explain a slower I/O access by the ARM server that is a disk bottleneck. Another possible reason is the lower memory capacity of the ARM server. The memory usage % for ARM and x64 servers are comparable at 35% for larger data sizes, but the x64 server has $4\times$ larger installed memory, While the MapReduce workers in both servers are limited to similar heap spaces, the x64 may benefit from a larger memory space for its critical shuffle and sort phases. These indicate the need for additional experiments to narrow down the analysis.

VI. Conclusion

In this paper, we have presented early results on comparing the performance of the first commercially available ARM64 server with an x64 server with similar specifications. The servers were evaluated for representative transactional and Big Data workloads. The results indicate that the ARM server shows comparable performance for small and medium workloads, but under-performs the x64 server for larger workloads. Specifically, it comes within 70% of the x64 server for the HTTPD workload and supports about 13,000 web requests per second. For MapReduce, it has comparable performance until 1 GB of data being sorted, but is unable to scale beyond that, due to a combination of slower HDDs and lower memory capacity. We should note that rated power envelope of the ARM processor is 32 Watts, and half as much as the x64 CPU's 65 Watt, as shown in Table I.

As future work, we propose to overcome some of the limitations of our study by having similar memory and disk characteristics for both servers so that the comparison is between more evenly matched servers, other than for the processor. We also plan to explicitly study the energy characteristics. The data intensive workloads we plan to study in future will include common algorithms like PageRank for MapReduce, as well as graph and stream platforms such as Apache Giraph and Storm, and querying over NoSQL databases. In the absence of multiple homogeneous servers, these studies will however be limited to single-node, pseudo-distributed evaluations. There is also value in benchmarking the behavior of such workloads in a virtualized or containerized environment.

References

[1] R. V. Aroca and L. M. G. Gonalves, "Towards green data centers: A comparison of x86 and arm architectures power efficiency," *jpdc*, 2012.

[2] D. Loghin, B. M. Tudor, H. Zhang, B. C. Ooi, and Y. M. Teo, "A performance study of big data on small nodes," *vldb*, 2015.

[3] M. Jarus, S. Varrette, A. Oleksiak, and P. Bouvry, "Performance evaluation and energy efficiency of high-density hpc platforms based on intel, amd and arm processors," *lsds*, 2013.

[4] M. A. Laurenzano, A. Tiwari, A. Jundt, J. Peraza, W. A.Ward, Jr., R. Campbell, and L. Carrington, "Characterizing the performance-energy tradeoff of small arm cores in hpc computation," *europar*, 2014.

[5] N. Rajovic, A. Rico, N. Puzovic, C. Adeniyi-Jones, and A. Ramirez, "Tibidabo: Making the case for an arm-based {HPC} system," *Future Generation Computer Systems*.

[6] K. Neshatpour, M. Malik, M. A. Ghodrat, A. Sasan, and H. Homayoun, "Energy-efficient acceleration of big data analytics: Applications using fpgas," *bigdata*, 2015.

[7] E. S.Chung, J. D.Davis, , and J. Lee, "Linqits: Big data on little clients," *icsa*, 2013.

[8] B. M. Tudor and Y. M. Teo, "On understanding the energy consumption of arm-based multicore servers," *sigmetrics*, 2013.

[9] S. Vijaykumar, S. G. Saravanakumar, and M.Balamurugan, "Unique sense: Smart computing prototype for industry 4.0 revolution with iot and bigdata implementation model," *ijst*, 2015.

[10] O. Svanfeldt-Winter, S. Lafond, and J. Lilius, "Cost and energy reduction evaluation for arm based web servers," *dasc*, 2011.

[11] J. Zhang, S. You, and L. Gruenwald, "Tiny gpu cluster for big spatial data: A preliminary performance evaluation," *icdcsw*, 2015.

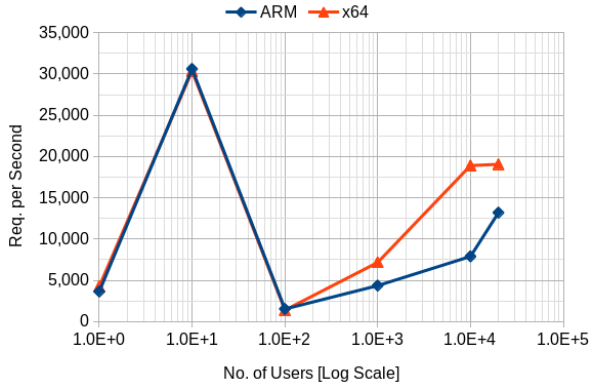
[12] "Apachebench description (wikipedia)," https://en.wikipedia.org/wiki/ApacheBench.

[13] "Official apachebench documentation (apache)," https://httpd.apache.org/docs/2.4/programs/ab.html.

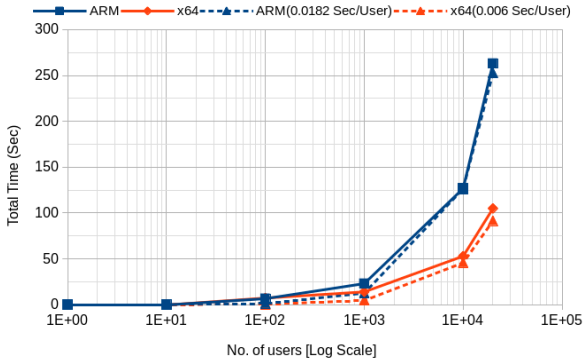
[14] Dina Bitton, et al., "A measure of transaction processing power," *Datamation*, vol. 31, no. 7, pp. 112–118, Apr. 1985.

[15] "Terasort-yahoo," http://sortbenchmark.org/YahooHadoop.pdf.

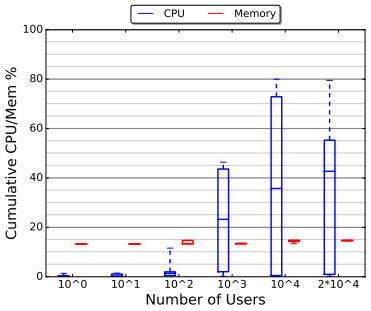
[16] "Sort benchmark home page," http://sortbenchmark.org, visited in July, 2016.



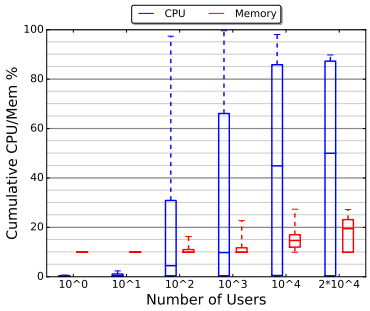
(a) Requests per second



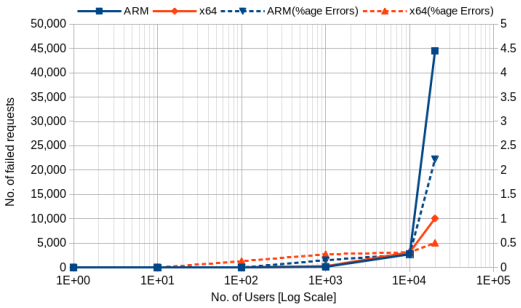
(b) Total time taken



(c) CPU/Memory Usage (ARM64)

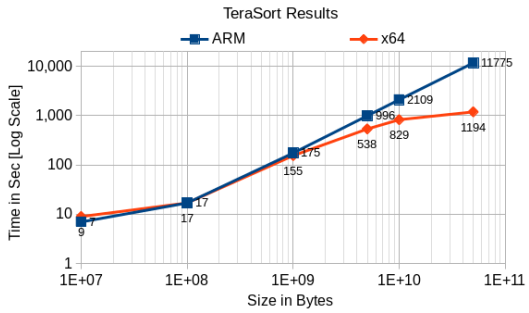


(d) CPU/Memory Usage (x64)

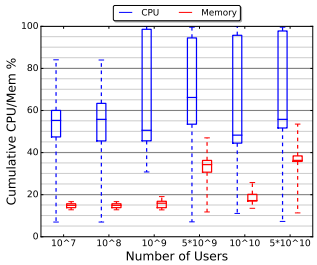


(e) Number and % requests that fail

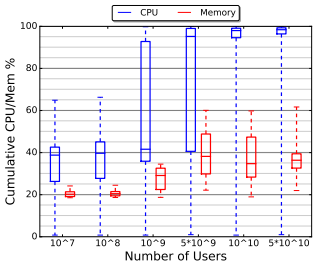
Figure 1: Performance of HTTPD service using `ab` workload by ARM and x64 servers



(a) Runtime for TeraSort



(b) CPU/Memory Usage (ARM64)



(c) CPU/Memory Usage (x64)

Figure 2: Performance of TeraSort MapReduce workload on ARM64 and x64 servers