Cloud Benchmark Testing of Cassandra on Raspberry Pi for Internet of Things Capability

THESIS

Daniel P. Richardson, Capt, USAF

AFIT-ENG-MS-17-M-065

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENG-MS-17-M-065

CLOUD BENCHMARK TESTING OF CASSANDRA ON RASPBERRY PI FOR INTERNET OF THINGS CAPABILITY

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Computer Engineering

Daniel P. Richardson, B.S.E.E.

Capt, USAF

March 1, 2017

AFIT-ENG-MS-17-M-065

CLOUD BENCHMARK TESTING OF CASSANDRA ON RASPBERRY PI FOR INTERNET OF THINGS CAPABILITY

THESIS

Daniel P. Richardson, B.S.E.E.
Capt, USAF

Committee Membership:

Lt Col John Pecarina, PhD
Chair

Maj Alan Lin, PhD
Member

Dr. Kenneth Hopkinson
Member

Table of Contents

Page

List of Figures

Figure Page

Figure Page

Figure Page

Figure		Page

List of Tables

Acknowledgements

Daniel P. Richardson

AFIT-ENG-MS-17-M-065

Abstract

This paper explores the distributed NoSQL database Cassandra's performance limitations in an IoT using limited hardware, namely the Raspberry Pi 2. Our aim is to use Cassandra's reliable and efficient data distribution to enable distributed exploits on real-time streaming data. At the time of this writing, the proposition of Cassandra on IoT-like hardware carries some development risk for a would-be application developer. This work not only demonstrates that actual operation of Cassandra is possible on Raspberry Pi, but also varies the conditions of operation to serve the expectation management of the variations inherent in new, creative, and cutting-edge applications. This work uses the Yahoo Cloud Services Benchmark (YCSB) not only for its popularity but to catalyze infusion of this research with existing and future research that will fully characterize IoT in the realm of controlled, flexible and resilient data storage. To represent IoT, we vary the memory on a virtual machine among 1GB, 2GB, and 4GB as well as implement Cassandra on the Raspberry Pi platform. This work demonstrates the feasibility and expected performance drops when porting a distributed database like Cassandra from powerful, stationary nodes to less powerful, but more flexible nodes.

CLOUD BENCHMARK TESTING OF CASSANDRA ON RASPBERRY PI FOR INTERNET OF THINGS CAPABILITY

I. Introduction

1.1 Background and Motivation

Moore's Law implies a trend that information technology and electronics generally become cheaper, powerful, and physically smaller as time progresses, which has led to the ensuing ubiquity of the IoT. Open source software and limited hardware like the Raspberry Pi embody this trend, both in terms of of lowering requisite eclectic expertise as well as price. Databases, key to IoT, are no exception, and it is of interest to explore the trade space involved in porting distributed database technology among such limited hardware. At the time of this writing, distributed databases like Cassandra are normally associated with large capacity nodes.

1.1.1 Potential Challenges.

Using Cassandra on nodes like the Raspberry Pi series in IoT is not without its challenges. First, while software developers may have the luxury of assuming increasingly powerful processing units and limitless memory, an IoT application cannot take such resources for granted. The Raspberry Pi 2 Model B, which will be used in this thesis, has 1GB of RAM available [2]. Storage for a single node depends on the SanDisk (SD) card, whose disk storage can vary from 8 GB to 256 GB and whose Input/Output (I/O) data rates can from 2 MB/s to 30 MB/s at the time of this writing. These amounts pale to the 768GB memory and 74.4 TB storage for, say, the

Lenovo Thinkserver RD650 [3] or some other server on the market. Prior to making a server purchase, it may of interest to predict the effect of varying the computing power of underlying hardware for some software of interest.

Second, using a node like the Raspberry Pi in IoT can imply a requirement for a finite (primary cell) or periodic (secondary cell) power source. Larger nodes, of course, consume their fair share of power, which comes with its own set of constraints. The prospect of deploying a node in new and unusual places is part of the motivation of porting software to low-power nodes like the Raspberry Pi series.

The prospect of nodes implying less power consumption, less weight, smaller dimensions, and/or maybe even a better price compared to an alternative all do their part to chip away barriers that creative minds would otherwise face in deploying applications for the betterment of mankind.

1.1.2 Potential Benefits.

Giving up computing power for less in power consumption, weight, dimensions, and price also has potential positive implication for innovation management.

1.1.2.1 Potential Benefits Particular to Supply Chain Management.

In supply chain management, especially in government, there is perpetual interest in items that qualify as Commercial-Off-The-Shelf (COTS). This interest, naturally, ties directly to cost and economies of scale. The more purposes something can serve, the greater demand is likely to be in the free market. This not only save on the actual unit price, but also engineering and administrative costs that may be incurred by triggering the full acquisition process of an application specific integrated circuit. The defense acquisition process is notorious for being overly burdensome, and limited hardware like the Raspberry Pi can represent the potential for rapid or general schedule acquisition.

There is another supply chain benefit as well. Despite security measures, it can be difficult to completely conceal a supply chain from an interested and skilled third party. The more application-specific a device is, the more insight it may give an unsavory adversary knowledge of mission parameters, whether it is physically captured, or if technical requirements leave a leaky paper trail. Purchasing limited hardware rather than application specific hardware has the potential to effect diffusion of such insight.

1.1.2.2 Potential Benefits Particular to Wireless.

There is no shortage of options for networking different hosts together, but the ability to exchange a wired networking medium for a wireless networking medium enables increased mobility and flexibility in placement. Wired networking options require cables that may incur installation costs and risk damage when exposed to many environments of interest. Allowing for wireless technology may be a critical enabler for some applications that have yet to be seen. However, along with these potential benefits comes with the risk of decreased or degraded performance. It is helpful to know in advance how similar endeavors have fared.

1.2 Problem Statement

This paper seeks to gain insight into how modern distributed databases operate in an IoT environment and what actions or configurations may be required or recommended to be in place to ensure feasible operation.

1.3 Research Goals and Hypothesis

1.3.1 Characterization of IoT and the IoT device.

Echoing [4], there is no ”standard identification” of IoT or an IoT device, although invoking the phrase can identify some archetypes, such as home automation systems or facilities management systems. All of these incorporate transducing some process over time, such as water pressure, steam pressure, temperature, etc, and may provide indication, such as a thermometer or dashboard, and/or actuation, such as actuation of a furnace or pump. This concept and its relationship to limited hardware is further addressed in the Chapter II.

The Raspberry Pi is able to receive digital data, audio signals, and video signals, and thus represents some of the computation and storage that could be attached or otherwise networked to one of these transducers. This question of how the Raspberry Pi fits into the IoT will be addressed in Chapter II through examination of current published work as well as commercial specifications.

The Raspberry Pi has no shortage of competitors. These will be briefly examined in Chapter II. In addition to the Raspberry Pi, Chapter III will evaluate a virtual machine set up to emulate limited hardware. Thus, both the Raspberry Pi and a virtual machine simulator will serve as representatives of IoT devices in this thesis.

Databases are key, and distributed databases offer flexibility and robustness that an alternative may not imply. Section 2.1.2 explains the current use of Cassandra and the benefits of porting distributed databases from a thinner client to a thicker client.

While IoT may imply the internet protocol (IP) at the network layer, there is no shortage of options at the data link and physical layers. Chapter II will discuss some networking considerations and explain some of the background behind varying between data link layers.

1.3.2 Feasibility of Distributed Database on Limited Hardware.

There is the question of whether a given distributed database will work on a given hardware platform or not. Any one-off results at a nonzero measure of performance indicate feasibility, but replicating operation and slightly varying the conditions gradually increases the confidence that such feasibility can be extended to yet untested environments. With the distributed database being represented by Cassandra, and limited hardware being represented by virtual machines as well as the Raspberry Pi, the experiments in this work aim toward refining the answer to this question.

Section 2.2.2 explains the reasoning behind selecting Cassandra as the representative of a distributed database. Section 2.2.3 addresses the reasoning behind the selection of YCSB.

Finally, this thesis aims to answer the following questions:

- *Does the amount of RAM affect Cassandra's performance?*

  Chapter III describes the empirical method used to determine an effect with respect to varying memory.

- *Platform Testing, Scalability Testing - What are the implications of using limited hardware?*

  Varying hardare platforms implies varying a variety of factors closely related to performance: Central Processing Unit (CPU), RAM, networking interfaces, and hard disk interfaces. Chapter III describes the empirical method used to measure the effect of porting Cassandra between a virtual machine the Raspberry Pi, as well as measuring the effect of cluster size. A common selling point for distributed databases is an ability to accept additional nodes for storage, as opposed to say, more storage in-situ.

- *Link Layer Testing - What effects on performance result from varying networking schemes?*

  Chapter III describes the empirical method used to measure the effect of switching between Ethernet links and 802.11a/b/g/n wireless links, as well as the scalability implied given 802.11a/b/g/n links.

- *Sensitivity Testing - How does variation in the configuration of a distributed database affect performance?*

  Different applications may imply different configurations associated with distributed databases, including compression strategies and replication factors. Chapter 3.1 does some investigating into how these can affect performance.

1.4 General Approach and Research Activity Overview

The general approach to this will be to implement a scientific methodology for understanding the effect of inherent aspects of IoT networks on factors that limit performance of distributed databases. Of particular interest are the effects of low memory and processor speed, limited bandwidth and scalability on IoT networked devices. In general, this study follows a template that includes varying configuration and environment settings, performing stress testing, measuring results, and interpreting the results to form a conclusion.

This paper will apply the YCSB benchmarking tool to gauge performance changes over variation in the following: keyspace configuration, network configuration, platform choice, and node scaleup.

1.5 Expected Contributions

Aggregating lower-cost, lightweight hardware spawns a lingering question of possibilities and performance due to lower barriers to proliferation. With distributed database Cassandra representative of application, and the Raspberry Pi a representative of low-cost hardware, we explore the performance of a distributed database over Raspberry Pi networked clusters. This paper asserts the following contributions:

- *Methodology*

  First, this paper develops a methodology to leverage existing tools [5] to evaluate a NoSQL distributed database in IoT. This will expand on such work as [1] and [5] developing a test methodology to explore the limits of Cassandra, and how its performance is affected by the number of nodes, nature of hardware, and links.

- *Performance Implications of Limited Hardware*

  Second, this paper reports results that suggest and characterize a differential between a simulated environment for IoT using virtual machines and an internal network and a physical one using Raspberry Pi's and an Ethernet network.

- *Performance Implications of Link Layer Variance*

  Third, this paper reports results that suggest a predictable performance cost associated with exchanging an 802.11a/b/g/n wireless links with those of a wired Ethernet network.

1.6 Organization

Chapter II paints the relevant parts of the landscape that have emerged as a result of much interest in distributed databases, limited hardware, and IoT. We also present the gaps of the current literature to describe the technical goals for the current work. Chapter 3.1 demonstrates some sensitivity testing. Chapter III presents a methodology to examine and evaluate the performance implications of varying hardware, network, and cluster size. Chapter IV describes the results and evaluation of the methodology presented in Chapter III. In Chapter V, this study is brought to a conclusion and future work is discussed.

II. Background and Related Works

2.1 Characterization of IoT and IoT devices

This section will seek to define where in IoT distributed databases such as that which is tested here, can augment or enable applications.

2.1.1 IoT Model.

See Figure 1 for where Cassandra is often used at present in IoT, as a back-end database for time-series sensor data. Figure 1 depicts a token sensor that sends data to a router, which in turn sends the data over a connection to a back-end database. It is here, at the back-end database, where Cassandra is put to use. The data is split and replicated over several servers with plentiful memory and storage where it can be queried from afar.

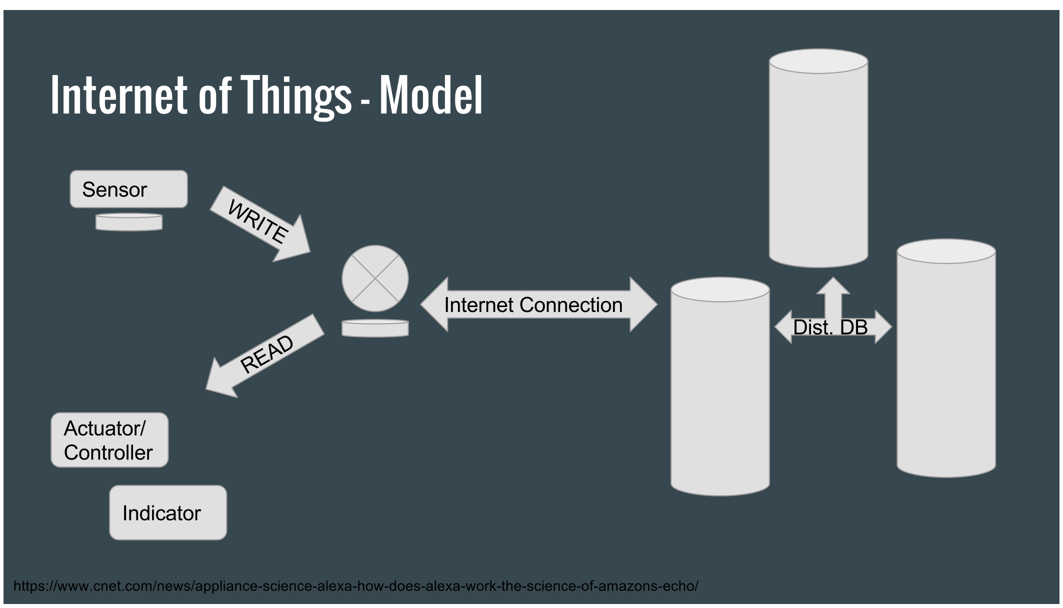

Figure 1. IoT This represents what for some might be a nominal "IoT" application. Some measurement of interest, such as temperature is sensed, that raw data is stored and sent afar, and if desired, the user may also query for some kind of feedback or indication.

2.1.2 Application Space.

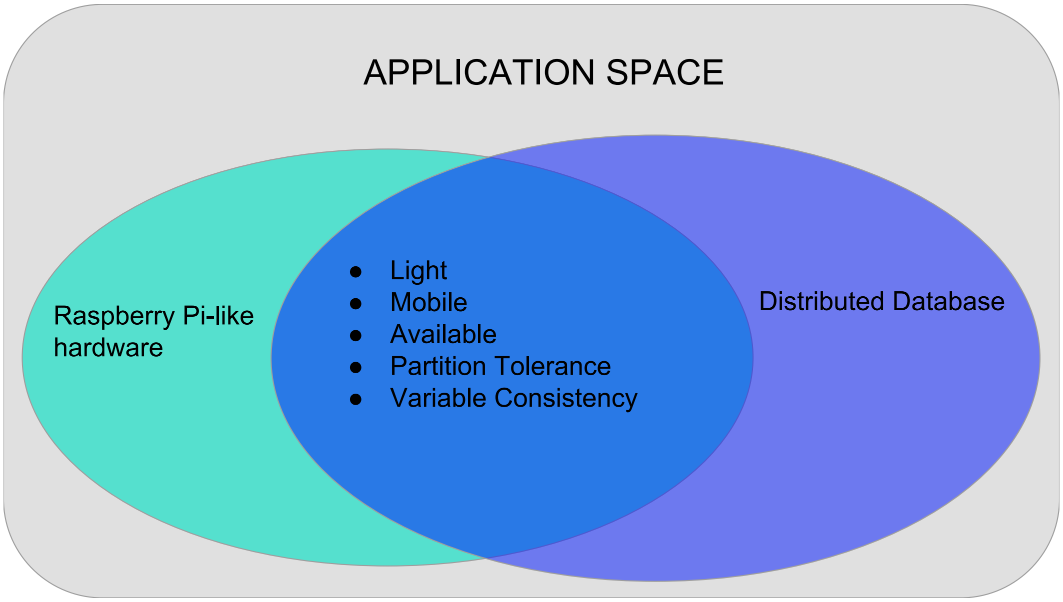

Figure 2. Venn Diagram of Application Space. The application space of interest takes advantage of some combination of lightness in weight, mobility, availability, partition tolerance, and variable consistency.

At the time of this writing, back-end database operating on powerful, but stationary nodes dominate documented use cases of Cassandra, as well as other distributed databases. This paper explores the idea of moving toward a more in-situ database, where the sensing nodes may also contain serve as mechanisms for storage and do not require a pipeline to another datacenter, a pipeline that from an operational perspective, represents a single point of failure, or from a security perspective, a threat to privacy. These ideas are depicted in Figures 2 and 3. Figure 2 paints in words what applications might seek to benefit the most from this research. Figure 3 is a modified version of Figure 1, where the connection to the back-end servers is either degraded or non-existent, and some version of in-situ storage must take its place to the best extent possible. There are obvious trade space among hardware capabilities and cluster size, but what is not obvious is how the database will actually perform.

The next few sections map this general concept to some existing applications and

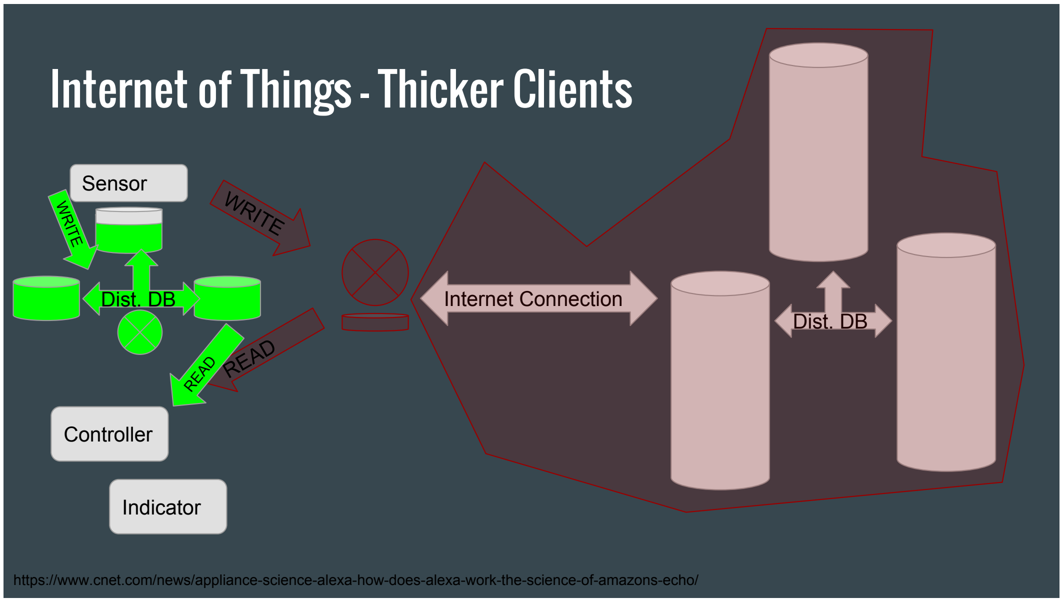

Figure 3. IoT Application with Thicker Clients...Depending on the application, using a distributed database on thicker clients in-situ may suitably replace a remote database operating in a remote location. From an operational standpoint, the pipeline may be unreliable, resulting in a loss of data. From a security standpoint, this pipeline may be subject to undesired third-party monitoring.

research that may benefit from such distributed database research.

2.1.3 WiFi Collection, Mapping and Analysis.

2.1.3.1 WiFi Sniffing, Collection.

WiFi (802.11) sniffing, or war-driving, has been explored by a number of enthusiasts, both in and out of the academic realm. These techniques for WiFi sniffing, at least on the front-end, are well-documented and thus represents a low barrier to entry for initial operations, often just requiring a specific WiFi chipset, such as the ALFA card, and open-source software. One example that has been well-documented goes by "Snoopy" [6]. Snoopy provides a framework for collecting WiFi data and observing with another handy piece of software, Maltego. Of course, a multitude of other software exists. For example, one can sniff wireless traffic by using software Aircrack-ng [7] or Airsnort [8]. However, although sensing the data is easy, the subject of storage has not been fully explored with respect to the selling points mentioned above. Investigating the limits of storage operations is key to unlocking realistic aspirations and application development.

2.1.3.2 WiFi Mapping.

It may worth noting one off-shoot of the WiFi sniffing mentioned above: WiFi mapping. There is much interest in WiFi mapping: it implies extra assurance to existing technologies that may be overly taken for granted. There has been much work done with respect to WiFi mapping. Argos [9] describes a similar system of a distributed system, but there is no mention of Cassandra or any distributed database, which may serve as an improvement on such a sensor network. Wigle.net [10] is an aggregate map that distributes a smart-phone application to collect GPS coordinate-Access Point pairs, but relies on a central database, and has an unreliable user base and irregular sampling and update frequencies (Relying on the public will often do that, unfortunately.). Heat Mapper [11] is partially free and commercial software that can generate a heat map for a small room or office. Wi2Me [12] performs this mapping as well, with an emphasis on performance and data throughput. It uses an instance of SQLite to store the traces on the individual's smart-phone, but again, none of these make use of a distributed database like Cassandra as part of the sensor network. Once again, this is a realm that may benefit integration with distributed databases, expanding the range possible node types, or both. However, there are limits that have to be anticipated. This paper aims to add a piece to that expectation management.

2.1.3.3 WiFi/Wireless Crowd Detection.

There is also considerable interest in crowd detection and related data gathering. Privacy implications notwithstanding, this kind of data can give a marketing-type a sense if certain areas are popular or versus other areas, or if certain paths are well-traveled or not. The way WiFi broadcasts Service Set Identifier (SSID)s in plain text, crowds can even be characterized as to whether individuals connect to similar access points or not. The data can indicate whether you have a crowd of people from a certain country, a crowd of people who know each other or maybe just a crowd of strangers. It can add assurance to more primitive forms of tracking, such as person-by-person registration. Informally, some have even reported to make art exploiting this mechanism [13], and not surprisingly, there are numerous claims that members of the public are routinely tracked via commercial entities via WiFi [14]. There have been more academic efforts to track crowds as well, notably a university campus and a music festival in [15] and an airport in [16]. In [15], data travels through Global System for Mobile Communications / Groupe SpecialMobile (GSM) to a central node, but does not use any kind of distributed database, like Cassandra. There are commercial entities that claim to track crowds and report data, namely "Bluemark" [17]. Here as well, a central server is utilized to collect the data. Users then reportedly log into the web to view the metrics. According to their marketing literature, they do use Raspberry Pi, but not for data storage. Paper [18] used this product line for their tests. Although WiFi utilizes Media Access Control (MAC) addresses supposedly unique to each device, research has found that this leaves more to be desired for many applications, namely crowd-tracking. For instance, some devices have been reported to change their MAC addresses [18]. There exist a number of papers that explore characterizing mobile devices and their users at the individual level [19, 20, 21, 22, 23, 24, 25] and propose techniques to better prepare data for analysis [18]. What is clear in the subtext of all these papers, is that these types of application research can only benefit from a wider choice in nodes, and a wider choice storage options as well as a reasonable amount of foresight into their performance.

In all of these, however, distributing the stored data among the nodes, like with Cassandra is either not used or not mentioned. Many utilize a central server that represents a single point of vulnerability.

2.1.4 CBIR and others and summary statement on application space.

The motivation for WiFi mapping can also be extended to other types of mapping, such as Content-based image retrieval (CBIR). Paving the way for feasible, desirable in-situ data storage increases the portability and development of these and many more applications.

The experiments that follow read and write random data in the context of a generic schema, but the experiments were done with the following in mind: A schema supporting any of these applications could be substituted in instead, and ideally with predictable results.

2.1.5 IoT Devices and Raspberry Pi.

The Raspberry Pi 2 (Model B) [2] is a low-cost computer designed sold from the United Kingdom. It can be described as a motherboard about the size of a 3x5 index card and has been available since February 2015. This experiment is interested in the Raspberry Pi 2 as a representative of the low-cost hardware domain, which implies low cost, low power consumption, and low in terms of size and weight.

This author's interest in the Raspberry Pi 2 is that its ARM Cortex-A7 processor and 1GB RAM [2] makes it a key representative of the low-cost hardware domain and IoT. The Raspberry Pi 2 has cost as low as 35 USD [26]. It is lightweight and has limited power consumption [2]. Constraints on size, weight, power, and cost can all be barriers to entry for applications seeking computing nodes.

Expanding this experiment, to say the BeagleBone black [27], is in touch with the spirit of this experiment but outside the scope of this paper and is reserved for future work. Various analogues and/or competitors to the Raspberry Pi are enumerated in Table 1.

Restricting the database schema to time series data, [31] claims to have achieved about 4 million inserts on a relational database on a Raspberry Pi 2 B+, one of the same models used in this work. However, the database in [31] lacked the capability and flexibility as Cassandra, and due to its custom nature could not be tested by the YCSB to be as robustly studied against other databases.

2.2 Benchmarking Distributed Databases for IoT

2.2.1 Small Cluster Computing on Raspberry Pi.

Considering the educational purpose of Raspberry Pis, it is no surprise to find academic interest in Raspberry Pi clusters. For instance, [32] evaluates a cluster of 8 Raspberry Pi nodes using SysBench.

Existing Raspberry Pi clusters, built to serve as a "practical balance" [33], such as in [34] and [33], suggest the value of Raspberry Pi nodes compared to large, traditional servers both in terms of power construction and actual purchasing price. In [35], Cassandra is used to store videos for a video streaming application on after a "a lot of configuration", albeit the configuration parameters were unspecified. However, [35] shows a high index of suspicion that Cassandra can be used in a small cluster environment.

2.2.2 Cassandra.

A distributed database offers a trade space among consistency, availability, and partition tolerance. Mostly this means that a system of nodes can still operate as expected even if there is a loss of one or two nodes. There is no shortage of distributed

Table 1. Raspberry Pi Alternatives, Depending on Application

Name	CPU	RAM	DISK	Price
Banana Pi M3 [28]	Octa-core 1.8GHz CPU	2 GB RAM	8 GB eMMC flash storage	73.00
CHIP [29]	R8 1GHz	512 MB	4 GB	9.00
VoCore	360 MHz MIPS CPU	32 MB	8 MB Flash	20.00
Arduino INDUSTRIAL 101 []	Atheros AR9331 Processor	64 MB	16 MB Flash	40.00
NanoPi 2 Fire []	Samsung S5P4418 quad-core ARM Cortex-A9, 1.4 GHz	unk	1 GB MicroSD card	22.99
NanoPC-T3 []	Samsung S5P6818 octa-core ARM Cortex-A53 up to 1.4 GHz	1-2GB of RAM	8GB of flash storage	60.00
Intel Edison with Kit for Arduino	Dual-core, dual-threaded Intel Atom CPU with a 32-bit Intel Quark microcontroller	1GB of RAM	4GB of flash storage	92.00
cloudBit []	Freescale i.MX23 ARM926EJ-S processor	64MB of RAM	4GB SD Card	59.95
Parallella [30]	16-core Epiphany RISC SOC	unk	unk	unk
Zynq SOC	(FPGA + ARM A9)	1GB SDRAM	Micro-SD storage	99
PixiePro	Freescale i.MX6Q Soc Quad Core ARM Cortex-A9 up to 1GHz	2GB of RAM	SD Card	129.95
Raspberry Pi	900 MHz quad-core ARM Cortex-A7 CPU	1GB RAM	MicroSD Card	35.90

databases to choose from, but Cassandra is widely used and is known to have a high write throughput [36]. Cassandra specifies the latest versions of both Java 8 and Python 2.7. In turn, Java can run on Windows, Mac OS X, Linux, and Solaris [37].

Cassandra is a widely used distributed Not Only Structured Query Language (NoSQL) database with many use cases [38] and boasts a high write throughput. Not only has Cassandra been reportedly been used in practice [39], but has been, using the Yahoo Cloud Services Benchmark [5], formally evaluated in scholarly literature against other databases such as MongoDB and proposed as the NoSQL database of choice in the Internet of Things and distributed sensor networks [40].

If possible, it is important to get realistic expectation from available specifications whether or not the database of interest, Cassandra, would be supported by the node of interest, in this case Raspberry Pi. At the time of this writing, Cassandra specifies it is supported by the latest versions of both Java 8 and Python 2.7 [37]. In turn, Java can run on Windows, Mac OS X, Linux, and Solaris. Raspbian, a Linux-based operating system, can be run on Raspberry Pi. In addition, there have been credible claims of Cassandra being used on Raspberry Pi [41, 42], but to the author's knowledge, no white paper with the details exists.

This author's interest in Cassandra lies in the fact that Cassandra is a distributed database used in practice for cloud computing. Although it describes itself as a NoSQL database, the interface allows for Structured Query Language (SQL) commands and has a Python Application Program Interface (API). Cassandra allows for configuration of distributed systems parameters, such as replication factor, but detailed knowledge of distributed systems protocols is not critical for operation.

The aim of this paper is to examine Cassandra's performance coupled with a simplified Wi-Fi collection and analysis application, where the nature of link nodes may be less reliable than wired Ethernet.

From an experimental standpoint, the distributed nature of a Cassandra "keyspace" lies in four parameters [43]: cluster size (the number of nodes), replication factor (configured in software), write level (configured in software), and read level (configured in software). As alluded to in the Methodology section, these factors will be held constant for this experiment.

2.2.3 Benchmarking Distributed Databases.

Benchmarks are the common parlance for a way to test a computing system's capabilities. There has been a lot of interest in testing distributed databases, databases that cover multiple nodes.

The "cassandra-stress" [44] tool is used to initially probe for sensitivity to configuration changes. Documented use includes [42], which, could be used as an anchor for methodology if desired. This tool is developed along with, and is optimized for Cassandra. However, it falls short if one desires to compare two different databases for the same task.

Instead, we explore a benchmark in this thesis that has been used to characterize scaling of Cassandra in other computing environments. Paper [36] presents the YCSB, highlighting "scaleup" and "elastic speedup" as parameters for benchmarking. It provides a survey of five databases: PNUTS, BigTable, HBase, Cassandra, and Sharded MySQL. As might be expected, Cassandra has the ability to be tuned based on the application, data distribution, and workload type. In [36], they claimed to "[tune] each system as best [they] know how." In contrast to boasting optimized performance, this paper, in the interest in repeatability and expectation management, will rely on default parameters and makes it a point to identify any tuning parameters that have been modified from the default. It is also worth noting that the version of Cassandra has evolved from year 2010, the time [36] was published.

The YCSB provides six pre-defined workloads and also allows one to determine a custom workload.

Cassandra was shown in [40] to be favorable to write-heavy workloads compared to another database in the domain. Notable about this paper is that the paper scales the node's RAM down to 2GB, compared to higher powered machines in other papers such as the Cooper paper [36] or as specified on the website [37]. Although it is not explicitly mentioned as an interest in the paper, this shows a transition of using Cassandra for lower powered machines. One thing that is not clear in [1] is how cache effects are accounted for. If unaccounted for, a cache effect may result in the initial run resulting in longer execution times than subsequent runs, all other factors being constant. The key cache is set at 100 MB and the row cache at 0. In contrast, this paper clears the data from (or truncates) the table of interest. [1] mention that each 7200 rpm with no stated limits on hard drive space. Moving into the realm of in-situ storage, this paper takes a significant deviation in limiting the hard disk space to 8 or 16 GB.

2.3 Networking Considerations

Although this paper takes an empirical, top-down approach to evaluation, some networking concepts may be useful to parse out, and predict what they mean to this

Table 2. This table describes the predefined workloads available from the YCSB

Workload	Description	Breakdown
A	Update heavy workload	50/50 reads and writes
B	Read mostly workload	95/5 reads/write
C	Read only	100% read
D	Read latest workload	95/5 reads/writes
E	Short ranges	95 scans / 5 inserts
F	Read-modify-write	50 read / 50 read-modify-write

work. This work only varies parameters at the link level. All terminology follows from [45].

This section will address delay in a local area network implied by the physical layer. Explanations are paraphrased from [45]. Variation in the physical layer may imply variation in end-to-end delay of a message composed of the following: Queuing Delay, Propogation Delay, Transmission Delay, and Processing Delay. Propogation delay is the most predictable, and reflects on the physical properties of the transmission medium. Transmission delay is dependent upon the message length and the transmission rate. In these experiments, the Maximum Transmission Unit (MTU) is set to 1500 on the router, so the message length would not change between Ethernet and 802.11a/b/g/n. However, the transmission rates do differ between Ethernet and 802.11a/b/g/n. Queuing delay, the outgoing delay, and processing delay, the incoming delay are not as predictable.

Ethernet is the dominant technology in modern wired LANs. Ethernet uses carrer sense multiple access with collision detection (CSMA/CD) and often uses switches.

The 802.11 protocol avoids collisions one of two ways. For shorter data frames, after sensing a channel idle for a specified interval called the Distributed Inter-frame Space (DIFS), the source sends the data frame. Then, it waits for an Acknowledge (ACK). For longer data frames, instead of sending the data frame right away, the source might send an Request-To-Send (RTS). This threshold can be set on the router, and in the case of these experiments it is set high, such that the RTS is never employed.

A key difference between the 802.11 protocol, or really any wireless protocol, is can be referred to as the 'hidden terminal problem' [45]. If nodes are spread out, it is possible for one node to miss interference caused by another node. A transmission source waits for an acknowledgement or retransmits. Contrasted with CSMA/CD, which does not require an acknowledgement from the receiver, this has the potential to cause greater latency.

III. Methodology

3.1 Cassandra Pilot Tests

This section will demonstrate some of the variance that is possible by varying configuration of a cassandra keyspace itself. These configurations may differ from application to application.

Cassandra comes with its own stress-testing tool, denoted 'cassandra-stress' [44] in the tarball installation. By all appearances, it is designed to give an operator an idea of how Cassandra performs over time on a given hardware setup. But, it also allows an operator to vary a limited set of configuration parameters with each command line execution. Naturally, this stress testing tool does not allow one to compare Cassandra to a competitor, hence the YCSB in the main section of this paper, but in theory Cassandra-stress may also serve as a check on the YCSB.

3.1.1 Experimental Setup.

The set-up follows the guidelines established in the main methodology. The set of experiments described in this section explored how Cassandra performs. Here a cluster of 3 Raspberry Pi 2 nodes was used. The tarball version of Cassandra version 3.9 was downloaded and installed. The stressor application cassandra-stress was executed from a laptop also connected to the LAN.

3.1.2 Variance in Nature of the Links with Compression Algorithms.

The plot in Figure 5 shows the effect of varying compression strategies for a given configuration on a pure write load as load-tested through the cassandra-stress module. In the left, wireless 802.11 links were used. On the right, wired Ethernet links were used. A one-way analysis-of-variance (ANOVA) may be able to test whether there is a

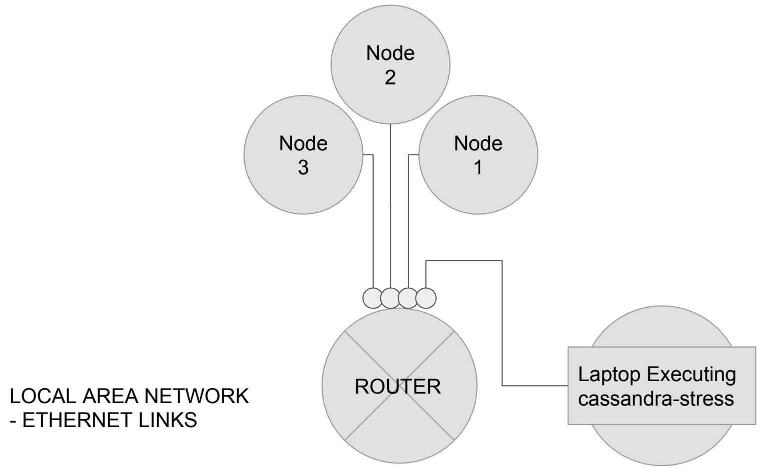

Figure 4. Experimental Setup for cassandra-stress Tests

significant differential between either of these two means, but from visual inspection, one could say that for a given configuration and a write-heavy load, varying the compression strategy does not have a large effect on performance as far as writes per second.

3.2 Overview of Similar Experiment

Although this work has a slightly different aim than [1]'s stated purpose, this paper aims to follow [1]'s methodology closely enough as to anchor its results to a cross-section of similar work that has been done. This work assumes that an in-situ storage application in the realm of IoT implies a small database, in this case represented by 1 million records, as opposed to 10 million or 100 million or more. Although [1] seems to imply there is feasibility for large database with many, many nodes given the right balance, this work focuses more on the initial impact to performance of introducing limited hardware in order to lighten costs or actual physical weight for an application that would see this as a benefit.

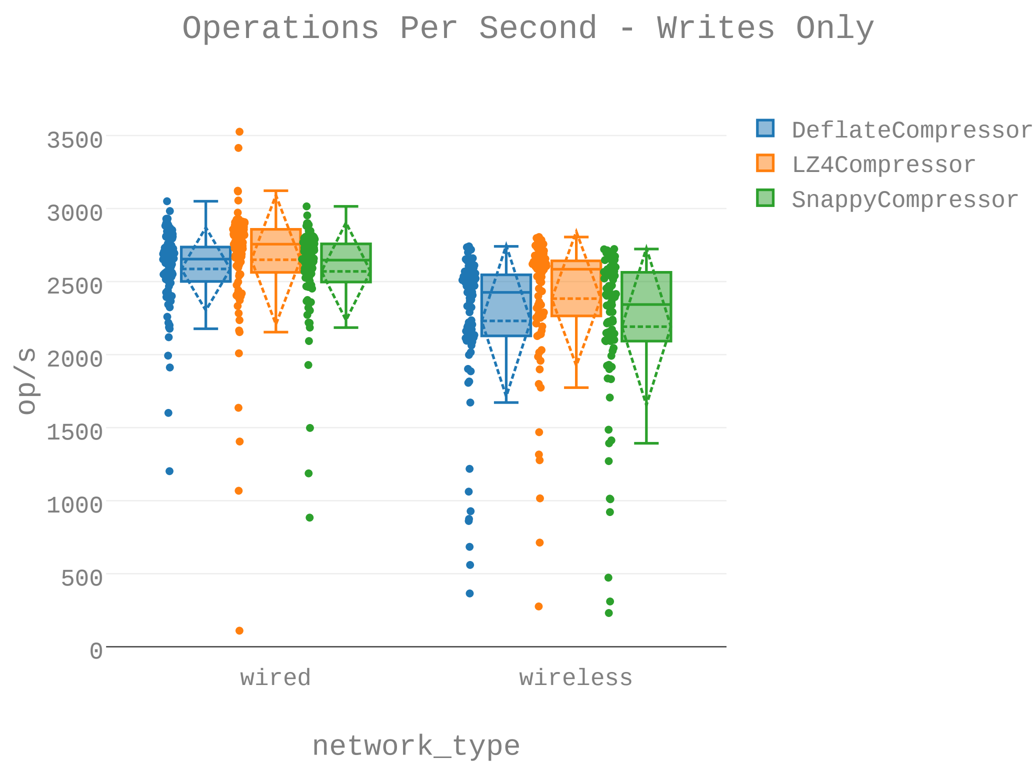

Figure 5. Varying Compression Methods: Writes ... Results are grouped by wired and wireless LAN configurations. From top to bottom, the solid horizontal lines of each box represent the maximum, 75 percentile, median, 25 percentile, and minimum. The dashed horizontal line represents the mean. The oblique dashed lines represent the standard deviation. The plotted points to the left of each box represent the actual data points.

Using standard workloads A, C, and E from the popular YCSB, the authors of [1] examined and evaluated Cassandras scalability over database [sizes] and cluster sizes [1]. The authors found that this trend, depicted in Figures 7, 8, and 9 did not necessarily hold true across database sizes, that in fact for larger database sizes of 10 million and 100 million records, 3 node clusters performed better than both a single node cluster and a 6-node cluster. The authors concluded that for sufficiently small databases, which is the likely case for IoT, more nodes imply more time to execute, which overwhelms any advantageous parallelism that may ensue with increasing nodes.

Because there are so many variables that can be at work, this work aims to anchor

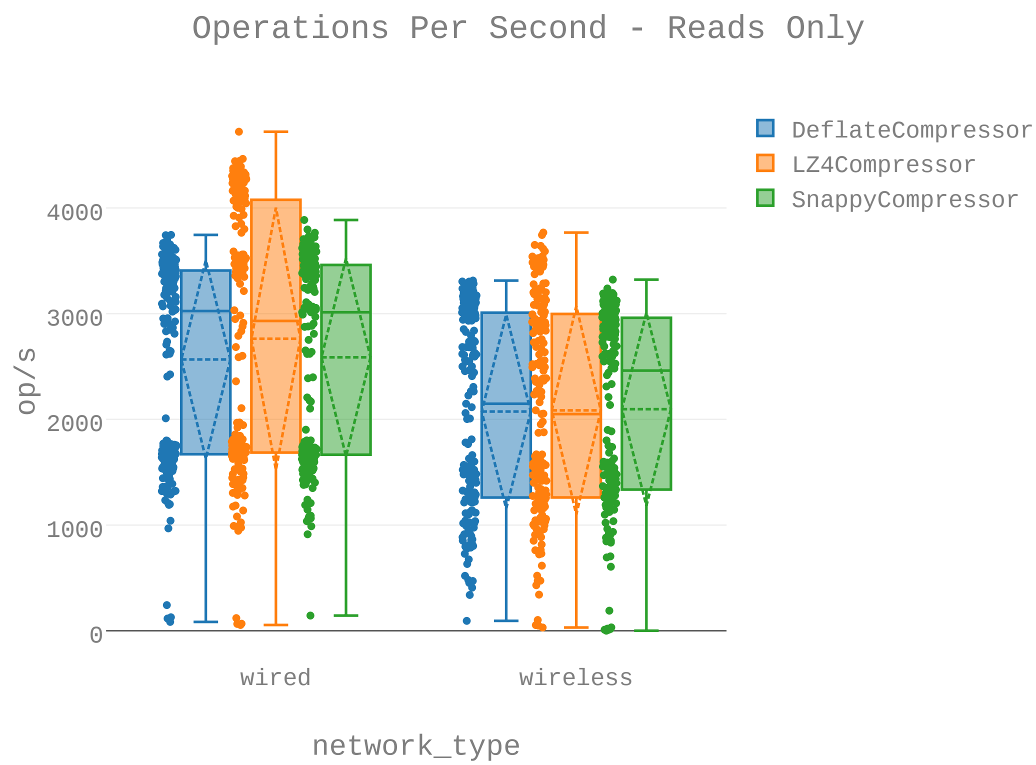

Figure 6. Varying Compression Methods: Reads ... Results are grouped by wired and wireless LAN configurations. From top to bottom, the solid horizontal lines of each box represent the maximum, 75 percentile, median, 25 percentile, and minimum. The dashed horizontal line represents the mean. The oblique dashed lines represent the standard deviation. The plotted points to the left of each box represent the actual data points.

its results by replicating part of Abramovas study. The extent of the details of the network in [1] is detailed below:

"The characteristics of nodes used are, as follows: Node 1 Dual Core (3.4 GHz), 2GB RAM and disk with 7200 rpm; Node 2 Dual Core (3.4 GHz), 2GB RAM and disk with 7200 rpm; Node 3 Dual Core (3.4 GHz), 2GB RAM and disk with 7200 rpm; Node 4 Dual Core (3.0 GHz), 2GB RAM and disk with 7200 rpm; Node 5 Dual Core (3.0 GHz), 2GB RAM and disk with 7200 rpm; Node 6 Virtual Machine with one Core (3.4 GHz), 2GB RAM and disk with 7200 rpm." [1]

The results of Workloads A, C, and E in [1] are depicted in Figures 7, 8, and 9 respectively, all depicting a positive correlation between execution time and the number of nodes. The actual values are depicted in Table **??**.

3.3 Objective of This Set of Experiments

The objective of this experiment is to characterize varying configurations for Cassandra. This characterization will be in service of assessing the utility of Cassandra on the Raspberry Pi 2, which will in turn be an indicator toward the greater population of both distributed databases, archetype Cassandra, and hardware of archetype Raspberry Pi.

We aim to recreate results in [1] and then extend these experiments for a better characterization for IoT. In doing so, we answer the following research questions:

- Research Question 1: How much is the execution time for a given number of operations extended in the system under test using workload A (reads/updates)?
- Research Question 2: How much is the execution time for a given number of operations extended in the system under test using workload C (reads)?
- Research Question 3: How much is the execution time for a given number of operations extended in the system under test using workload E (insert/scans)?

Table 3. Results from [1] for 1 million record size database

Nodes	Workload	[OVERALL] RunTime(ms)
1	A	58430
3	A	65650
6	A	87310
1	C	88000
3	C	90210
6	C	118090
1	E	223180
3	E	330820
6	E	404660

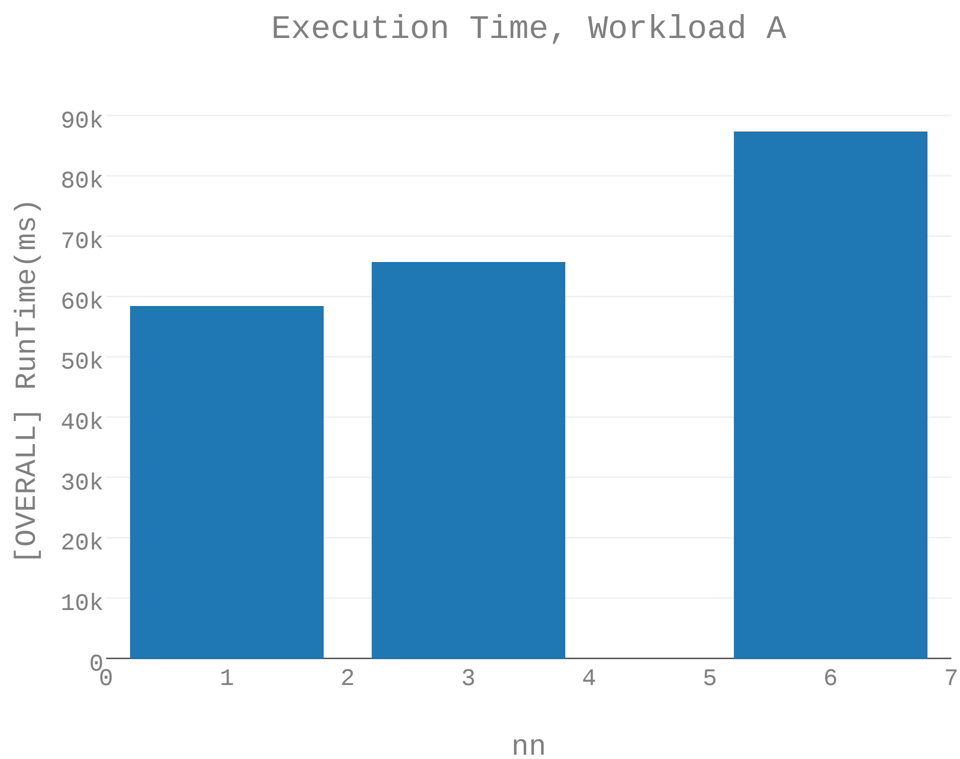

Figure 7. Cross Section of Results Imported from [1]. This figure shows the execution time reported for 10,000 operations of Workload A over three given configurations: a network with only one (1) node, a network with 3 nodes, and a network with 6 nodes.

- Research Question 4: And finally, how much is the execution time for a given number of operations extended in the system under test using custom workload I, the anticipated representative of in-situ distributed database IoT application?

To answer each of these questions, we pose the following questions:

- How do the results from virtual machine compare to the corresponding values from [1]?
- How do the results compare among the various RAM levels assigned to virtual machines?
- How do the results from the limited hardware on an Ethernet LAN scale as the cluster size increases?

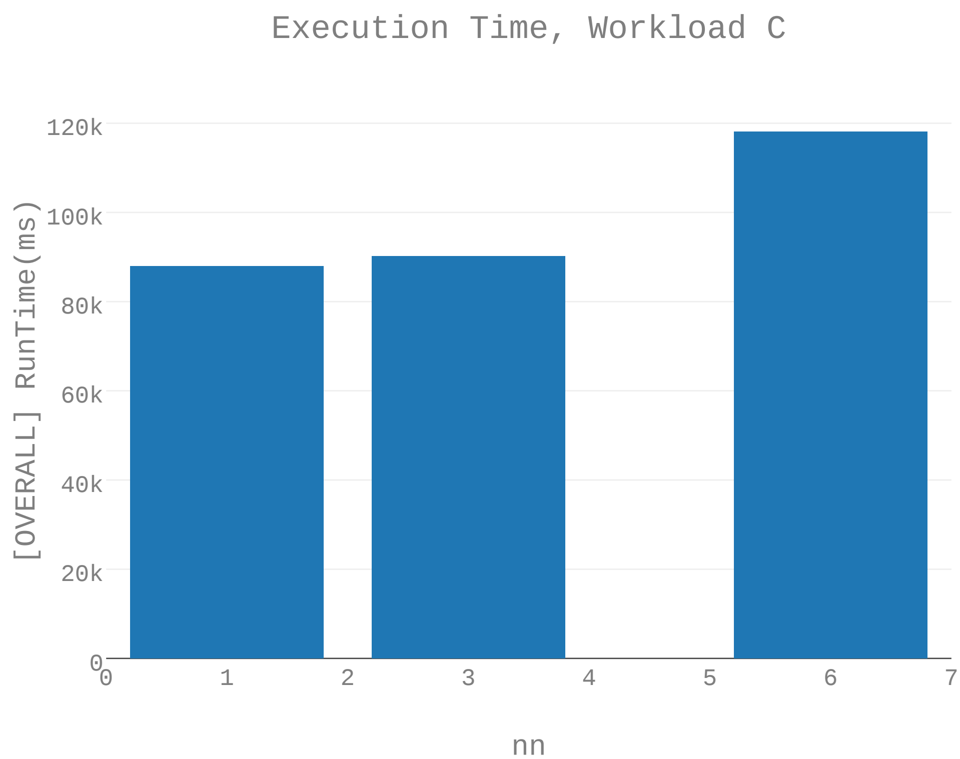

Figure 8. Cross Section of Results Imported from [1]. This figure shows the execution time reported for 10,000 operations of Workload C over three given configurations: a network with only one (1) node, a network with 3 nodes, and a network with 6 nodes.

- How do the results from the limited hardware on an Ethernet LAN compare to the corresponding values from [1]?
- How do the results from the limited hardware on an Ethernet LAN compare to the corresponding virtual machine results?
- How do the results from the limited hardware on a wireless LAN scale as the cluster size increases?
- How do the results from the limited hardware on an Ethernet LAN compare to that of the limited hardware on a wireless LAN?

The results of these experiments are anticipated to be within reasonable bounds of previous work [1]. Increasing the cluster size will result in an increase in execution

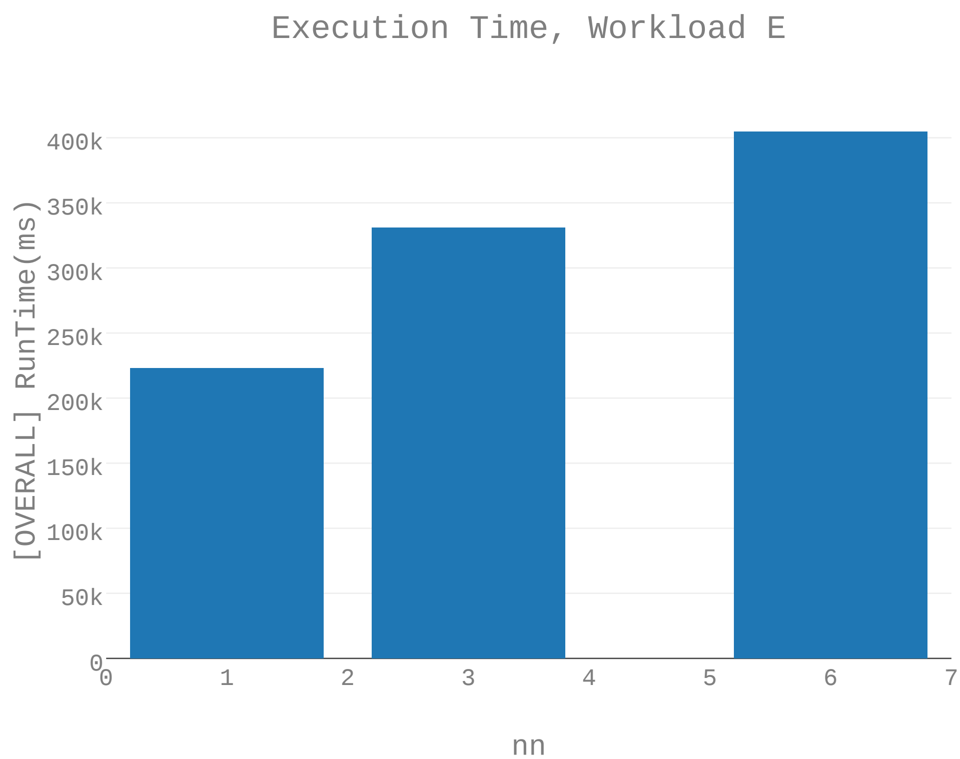

Figure 9. Cross Section of Results Imported from [1]. This figure shows the execution time reported for 10,000 operations of Workload E over three given configurations: a network with only one (1) node, a network with 3 nodes, and a network with 6 nodes.

time. Wireless links will cost significantly in performance compared to their wired counterparts.

3.4 Testbed Environment

3.4.1 System Boundaries.

Virtual machine nodes were contained in a single laptop. The wired LAN experiments were performed with all nodes and the associated router within about 3 meters of one another on a dedicated LAN. Likewise, the wireless LAN experiments were performed with all nodes and the router within a radius of 3 meters on a dedicated and secured LAN. The experiments were done in a residential, suburban area. The network was periodically monitored for unexpected hosts.

3.4.2 Experimental Limitations, Nuisance Factors, Known/Suspected Interactions.

The amount of interference on the industrial, scientific, and medical (ISM) band could not be controlled. It is left to the assumptions that any variation in interference was negligible between any two trials.

The laptop running the YCSB may have been running other minor programs or other processes to a limited extent. No experiments were done to determine if this would have a significant effect, and this was not strictly controlled.

3.4.3 Coordination and Networking.

On an Ethernet LAN, no coordination was needed. This network was physically isolated and no additional traffic was expected to interfere with it. On a wireless LAN, because the scale and timing of this experiment was limited, there was no formal coordination needed. However, because a larger experiment could possibly fill up one (or more) frequency channels, one must be courteous of the environment. An extended test would not be appropriate in an uncontrolled environment.

3.4.3.1 Ethernet LAN Setup.

The wiring of the Ethernet LAN is depicted in Figure 10. Note that for the Ethernet set up, the wireless capability for the home router was switched OFF.

All nodes, including the laptop used to execute the YCSB, were on the same network. Their IP addresses are listed in Table 4.

3.4.3.2 802.11 LAN Setup.

Table 6 describes the local area network as the wireless the. Note that for this setup, the Ethernet cables were physically unplugged.

Table 4. This table describes the IP addresses in order to paint a further detailed understanding of network set up. The netmask is 255.255.255.0

Node	Host Name	IP Address	Model
Node 1	raspberrypi0	192.168.1.100	2B
Node 2	raspberrypi1	192.168.1.101	2B
Node 3	raspberrypi2	192.168.1.102	2B
Node 4	raspberrypi3	192.168.1.103	2B
Node 5	raspberrypi4	192.168.1.104	2B
Node 6	raspberrypi9	192.168.1.109	3
Laptop	daniel-ThinkPad-W541	192.168.1.200	-

Table 5. This table describes the IP addresses in order to paint a further detailed understanding of network set up. The netmask is 255.255.255.0

Node	Host Name	IP Address
Node 1	c0	192.168.56.100
Node 2	c1	192.168.56.101
Node 3	c2	192.168.56.102
Node 4	c3	192.168.56.103
Node 5	c4	192.168.56.104
Node 6	c5	192.168.56.105
Laptop	daniel-ThinkPad-W541	192.168.56.200

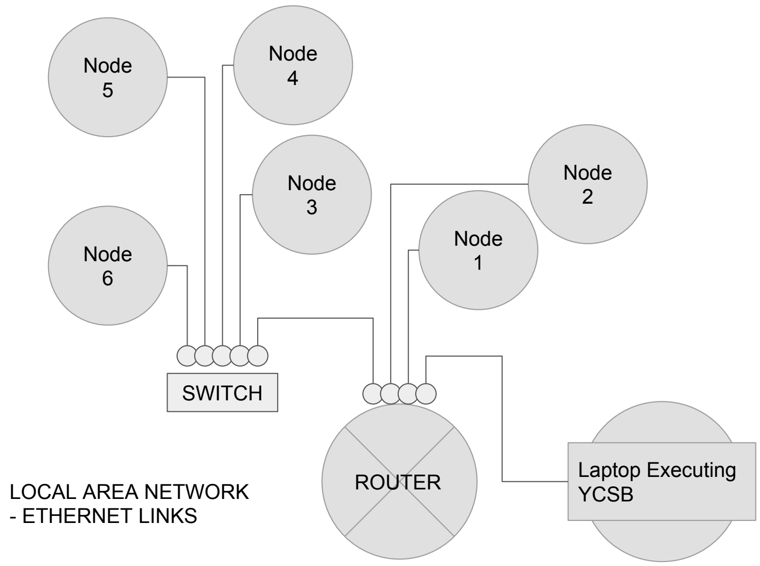

Figure 10. Topology and Wiring for Ethernet Setup

To enable wireless links on the Raspberry Pi 2's, the Wi-Pi Universal Serial Bus (USB) module was utilized, with transmission speed capabilities listed at "11b 1/2/5.5/11Mbps, 11g 6/9/12/18/24/36/48/54Mbps, 11n up to 150Mbps". The Raspberry Pi 3 model comes with its own built-in WiFi capabilities.

Table 6. This table describes the IP addresses in order to paint a further detailed understanding of network set up. The netmask is 255.255.255.0

Node	Host Name	IP Address
Node 1	raspberrypi0	192.168.1.130
Node 2	raspberrypi1	192.168.1.131
Node 3	raspberrypi2	192.168.1.132
Node 4	raspberrypi3	192.168.1.133
Node 5	raspberrypi4	192.168.1.134
Node 6	raspberrypi9	192.168.1.139
Laptop	daniel-ThinkPad-W541	192.168.1.200

3.4.4 Treatments, Independent Variables.

The independent variables of interest are the node type and link type. Node type is characterized by memory, or RAM, processor speed, and I/O rates, and are represented by virtual nodes and the Raspberry Pi.

The link types are the internal nodes on the virtual machine network, Ethernet links, and wireless 802.11 links.

There are many other factors at work, but other factors, like workload, are only varied to give appropriate context to the variance in node type and link type.

3.4.5 Configuration of Cassandra.

Unless otherwise specified, the configuration of Cassandra, the keyspace, and the table within the keyspace are all held constant to their default values. All three of these things can be configured hierarchically, and their configuration can affect the performance of Cassandra on a given load. A skillful adjustment of these parameters can result in an optimized performance of Cassandra.

This paper aims to highlight the differences in varying hardware, and thus the exact configuration, despite that it might raise the absolute level of performance, is not expected to elevate the relative level of performance one would see shifting between, say a virtual node on a laptop and a Raspberry Pi module.

There are a few settings that had to be taken into account. Denoted *commitlog_total_space_in_mb*, this will accumulate to an undesired level. This was set to 512 MB in the Cassandra configuration file, cassandra.yaml. (Changing this to 256 MB for the 512 RAM case did not correct the error mentioned above.) Also, to prevent the accumulation of space, the setting *auto_snapshot* to false in configuration file cassandra.yaml. Having *auto_snapshot* set to true automatically backs up, or saves a snapshot of, the data on Cassandra. For a limited-capacity node, this would quickly lead to a crash.

The configuration files can be found among the appendices.

3.5 Experimental Setup

In this experiment, we measure the total run time of a fixed number of operations of Workload A for various memory sizes. The choice of memory size is due to the expectations of IoT devices. The Raspberry Pi 2 and 3 have 1GB of memory and is our representative technology for IoT. The 4GB configuration can be considered be more representative of a low-end desktop, laptops, or virtual machine. The 2GB configuration is an intermediate stage that show an intermediate performance level and naturally, may represent the aim of future of IoT nodes. In addition, [1] used 2GB machines, which may help this work to be compared against existing work.

The experiments performed can be summarized in Table 7.

3.5.1 Virtual Node Setup.

For this work, virtual nodes were created to match these characteristics to a reasonable extent. Facing minimal propagation delay due to being connected on a nodal network, experiments with the virtual machines seek to place an upper limit on potential IoT performance expectations. Replicating the exact network in [1] is not absolutely necessarily, notwithstanding the details and materials available to do

Table 7. This table summarizes each network topology that was explored in each research question. The RAM was varied on the Virtual Machines.

Communication (nm)	Platform (nt)	Assigned RAM
Nodal	Virtual Machine	1 GB
Nodal	Virtual Machine	2 GB
Nodal	Virtual Machine	4 GB
Ethernet LAN	Raspberry Pi	1 GB
802.11 LAN	Raspberry Pi	1 GB

so are unavailable.

On a 64-bit 31.1GiB RAM laptop with Intel Core i7-4910MQ 2.90GHz 8-core central processing unit, six (6) identical virtual machines were created in software VirtualBox. Each machine consisted of Ubuntu 64 bit machine was was allocated 8 GB of hard drive disk space. The full details for these machines, reported by means of the lshw Linux command, are located among the appendices. Also in VirtualBox, a host-only network entitled vboxnet0 was instantiated, to which all six machines were connected.

The YCSB was installed and run on the host laptop. PyCharm drove a terminal process, which in turn drove the YCSB software.

To further paint the picture of the network setup, the IP addresses are included in Table 5.

3.5.2 Other Factors.

The manufacturer and model for each SD card was kept constant for each node. The details can be found in Table 8.

This work assumes that an in-situ storage application in the realm of IoT implies a small database, in this case represented by 1 million records, as opposed to 10 million or 100 million or more. Although Abramovas paper seems to imply there is feasibility for large database with many, many nodes given the right balance, this work focuses

Table 8. Specifications for SD Cards

Specification	Value
Capacity	16 GB
Read Speed	up to 90 MB/s
Write Speed	up to 40 MB/s
Video Speed	C10 U3

more on the initial impact to performance of introducing less-capable hardware in order to lighten costs or actual physical weight for an application that would see this as a benefit.

3.5.3 Number of Operations Per Trial.

The results in [1] report an execution time after 10,000 operations, and the decision was made to keep this constant rather than vary the representative number of operations. No explicit justification is given in [1] for the number 10,000, but it can be inferred that is a compromise between under-sampling and an undue burden in time spent sampling.

As a brief exploration into the possbility of under-sampling, one can take a look at what it might look like to vary the sample size in a given trial, and how the execution time would vary over the sequence of trials. The graph above shows this for two different configurations: 1GB and 4GB RAM on a virtual machine. As expected, the execution time does reflect initial cache warm-up time, but then, more or less, reflects some sort of steady state, albeit oscillating performance. It was also not made explicit whether or not the 10,000 operations represented one of many trials of 10,000 or represented a single and only trial. For the purposes of exploring the data, this author chose to run 30 trials of 10,000 operations each.

The oscillating behavior seen in trials 5 through 120 of Figure 11 is distracting and risks inaccurate comparisons among configurations. Although it is beyond the scope of this paper to determine the exact cause of this oscillation, one might consider that activities required for the operation of the database, such as compaction, the gossip protocol, and other operations compete with the reads and will contribute to continuous variation over time, and may affect performance measurements. The reason for choosing 10,000 operations is not explicitly reported in [1], but one may

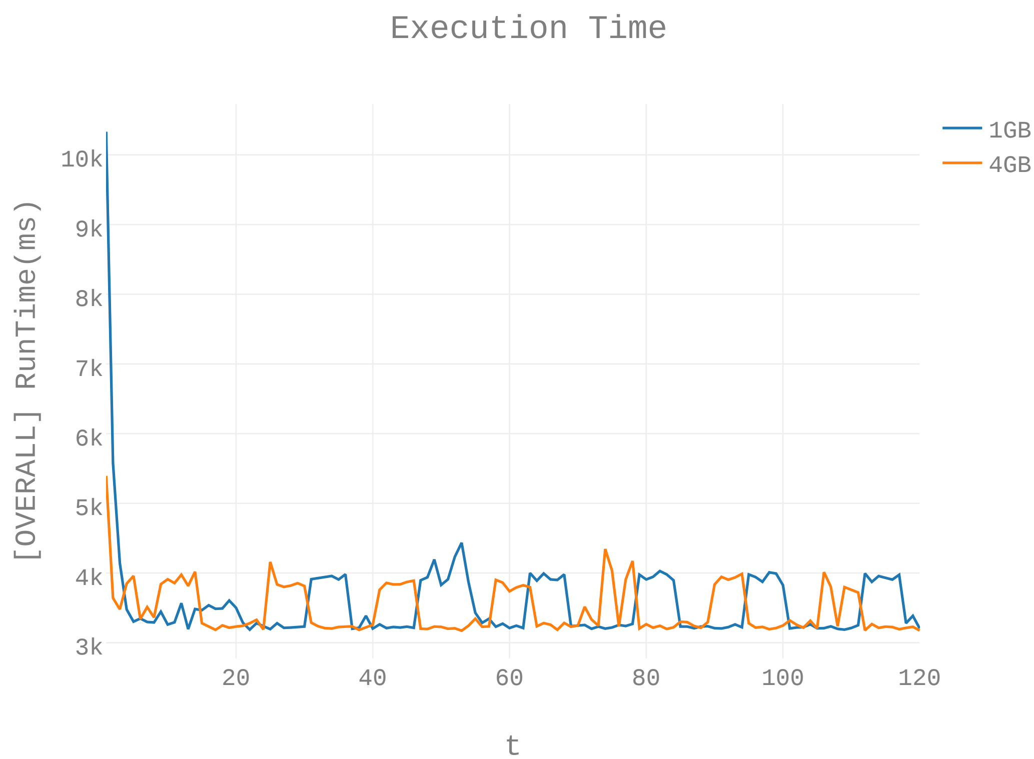

Figure 11. Two sample runs, adjusting the trials such that only 1000 operations of Workload A to a given trial. Once the cache is warmed up, it can be seen that for both series, there is a periodicity present with respect to the progress of trials. In an experiment concerned with steady state performance and overall trends, such oscillation may be an undue distraction. Label t represents the trials: Trial 1, Trial 2, and so on.

infer the reason is to integrate this variation to make better comparisons among configurations.

Contrast the 1k operation trials with the 10k operation trials depicted in Figure 12. Here, at least with the naked eye, there is no oscillation, and most certainly not to the extent seen in the 1k trial case. Whatever effected the oscillating behavior in the previous graph is integrated into the 10k trial case and is no longer a distraction to steady-state analysis, as each 10k operation trial soaks up, or rather integrates, the performance of quantity 10 1k operation trials. Of course, more operations take longer to run, and thus the cost in testing time starts to curtail the value of integrating more trials.

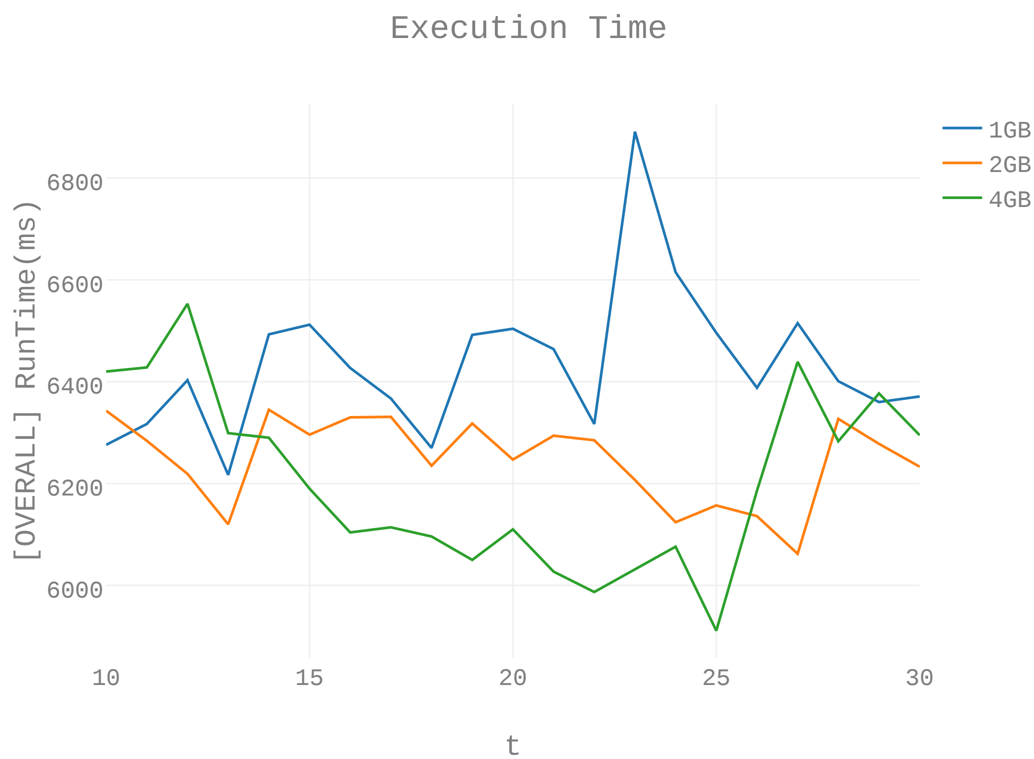

Figure 12. A series of trials for 1GB, 2GB, and 4GB RAM virtual machines. The inset demonstrates that any pattern that could significantly distinguish one trial from another, such as the oscillation seen in Figure 11 is integrated such that, after the cache warm-up period, the relative position of the trial does not predict the relative outcome. Label t represents the trials: Trial 1, Trial 2, and so on.

3.5.4 Workloads.

Just as in [1], standard workloads A, C, and E were put to the test, while B, D, and F are de-prioritized. In addition, this paper introduces a custom workload, denoted I, to represent an IoT application, many inserts and few reads, a 99 percent to 1 percent ratio.

Standard workloads A, C, and E, from the YCSB are summarized in Table 9.

3.5.5 Threads in the YCSB.

The number of threads was kept constant at 1, although by increasing the number of threads one could achieve greater throughput. However, since it was desired to

Table 9. Standard YCSB workloads used in this methodology. Workload A consists of 50 percent reads and 50 percent updates. Workload C consists of 100 percent reads. Workload E consists of 100 percent scans.

Workload	Read	Update	Scan
A	0.5	0.5	0.0
C	1.0	0.0	0.0
E	0.0	0.0	0.95

Table 10. Standard YCSB workloads used in this methodology. Workload A consists of 50 percent reads and 50 percent updates. Workload C consists of 100 percent reads. Workload E consists of 100 percent scans. In addition, a custom workload I is summarized, which consists of 99 percent writes and 1 percent reads to represent IoT.

Workload	Read	Update	Scan	Insert
A	0.50	0.50	0.00	0.00
C	1.00	0.00	0.00	0.00
E	0.00	0.00	0.95	0.05
I	0.01	0.00	0.00	0.99

compare different calculations, the default was retained for all configurations.

It may be worth noting that for loads, this number was increased for practical reasons. However, these preparatory loads were not measured in these experiments, only the defined workloads A, C, E, and custom workload I.

3.5.6 Assumptions.

Naturally, in order to perform the experiment and evaluate the results, some assumptions had to be made. Investigation into any of these assumptions may be an avenue for future work.

1. The benchmark represents the application, which assumes a simple schema. In other words, Cassandra's performance is not particularly sensitive to the schema.

2. There is no active attacker or intrusion into the local area networks. Both the local area networks are isolated.

3. There are no errors with the custom benchmark that would skew the results. Any error is due to the fact that the system's limits have been reached.

4. There are no bugs in the benchmark that would skew the results.

5. Effects on the network due to distance are negligible.

6. This experiment assumes that nodes are homogeneous. The basis for this assumption is that all nodes have been specified to the same model of Raspberry Pi 2. The same make and model for the SD Cards have been used. The image upon the SD Cards has been copied and only adjusted to account for specific, differentiated IP addresses.

7. This experiment assumes an uninterrupted power supply. Power is not measured nor accounted for in the model. As long as the power has been turned on, it stays on, and fluctuations in voltage or any kind of imperfections in the power supply are negligible with respect to Cassandra's performance.

8. Although the ISM band is unregulated, this experiment assumes invariant interference from other emitters. The experiment assumes an urban to suburban environment. In other words, congestion that overwhelms Cassandra's performance can be assumed to be rare with respect to the population, and is ignored for the purposes of the experiment.

3.6 Execution and Analysis

The YCSB, installed on the host laptop, is also run from the host laptop. The YCSB can be run from the terminal, but for convenience, a Python script was developed to drive a series terminal processes.

For each experiment, the trials for each configuration will be reported as a summary of execution times for 10,000 operations. All execution times will be reported in milliseconds.

The YCSB reports a number of measured values, including operations per second and latency distribution (minimum, mean, 50 percentile or median, 75 percentile, 99 percentile, maximum). The total execution time in milliseconds was chosen in order to keep the measured values true to the limits of the experiment. Although this paper aims to analyze results to reflect the steady state, this paper does not deny the possibility that variance in operations per second or variance in latency may result from variance in the number of operations per trial.

Also, one can note in both Figure 11 and Figure 12, that in the first five trials or so, one can observe the cache warm-up period. This steep decline is expected due to the effect of the key cache, which is at its default setting: Cassandra sets the key cache to the either 5% of the heap, or 100 MB, whichever is less. In [1], the key cache is reported to be at 100 MB. The steady state operation is dependent on the keys requested, so for a workload like the YCSB, one would not expect a lot of variation.

Truncating the head of the trials, trials 1 though 9, the cache effect is no longer depicted, rendering an expectation of steady-state performance after cache warm-up. This is necessary to meet the assumptions of any ANOVA test.

Taking the issues of oscillating behavior and cache warm-up period into account, we can remove them to find a stable viewing window into the behavior of the Cassandra database, as depicted in Figure 12. Figure 12 shows the desired observation of the behavior in question for 1GB, 2GB ad 4GB memory sizes. We use this data to recreate Abramovas work and extend it for other memory sizes.

This is also the logic behind using the median to summarize the execution times, which should not differ significantly whether the cache warm-up trials are included or not.

Another important observation here, is that once the cache effect is cropped out, there is no obvious correlation between trials and the performance measurement. This further supports that 10,000 operations, minus cache effect, does represent a steady state that is likely to extend beyond 10,000 operations.

IV. Results and Evaluation

This section discusses the empirical results of the experiments performed. First, the effect of RAM is examined, which was conducted using the virtual machine configurations. Then, we examined the

4.1 Variation in RAM

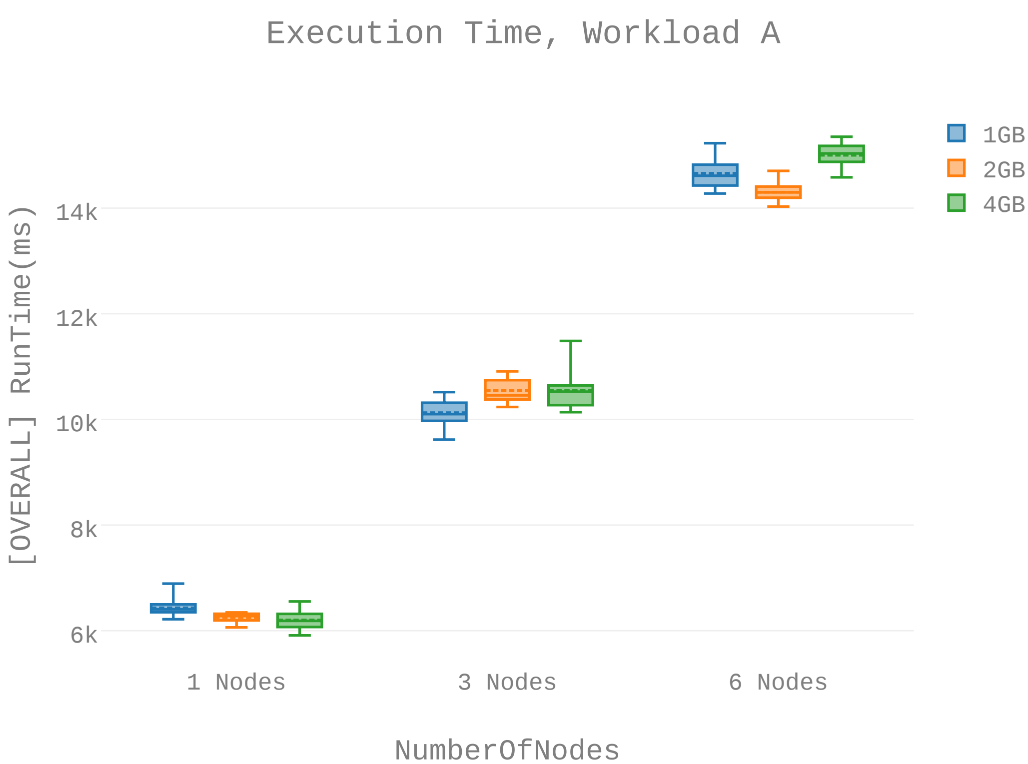

Figure 13. Variation in RAM over Workload A for the virtual machine clusters. The vertical axis shows the overall execution time in milliseconds. Each group along the x-axis runs the number of nodes in each cluster of interest. In each set, in order from left to right, the 1GB RAM is represented by blue, 2GB RAM by orange, and 4GB by green. Note that only trials 10 through 30 are represented. Each solid horizontal line depicts the maximum, 75 percent quartile, 50 percent quartile, 25 percent quartile, and minimum for the specified data set. Any dots present represent outliers, which for this graphing method, are any values exceeding three standard deviations from the mean. A dotted horizontal line represents the mean.

Shown here is a boxplot that represents the results of varying the RAM, grouped

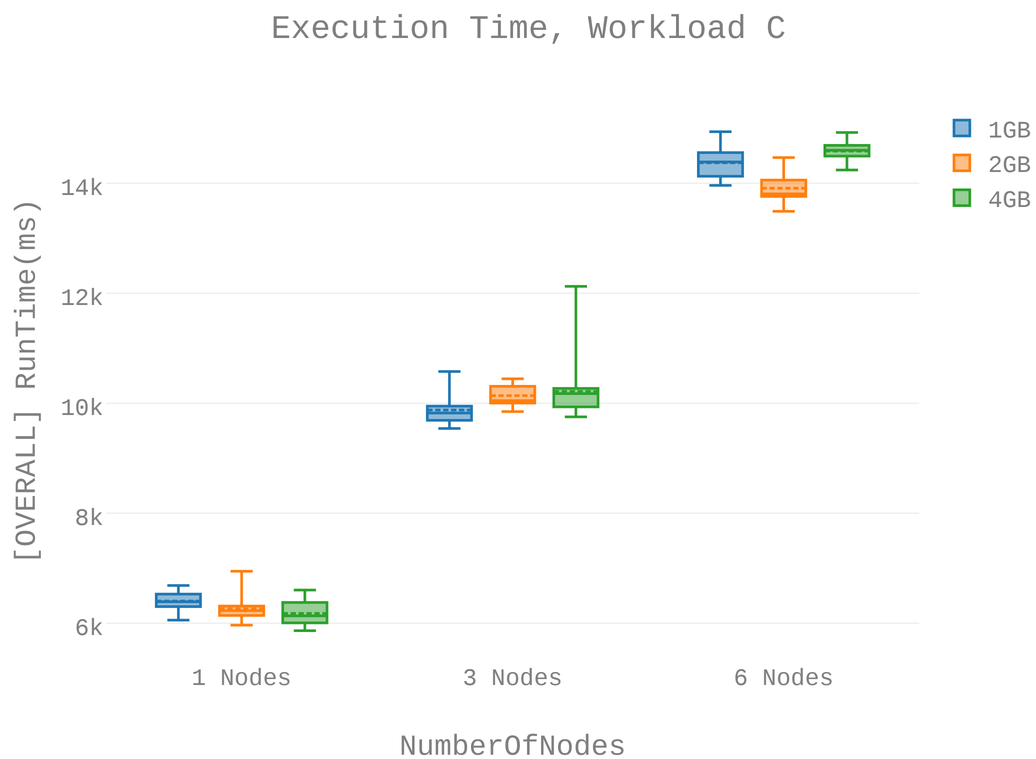

Figure 14. Variation in RAM over Workload C for the virtual machine clusters. The vertical axis shows the overall execution time in milliseconds. Each group along the x-axis runs the number of nodes in each cluster of interest. In each set, in order from left to right, the 1GB RAM is represented by blue, 2GB RAM by orange, and 4GB by green. Note that only trials 10 through 30 are represented.

by cluster size. We can accept that cluster size is expected to have sizeable results. What isn't clear is if RAM makes a predictable difference when it is varied. Taking a look, it would be hard to deny with full confidence that there is no pattern at all, even a complex pattern. While a complex pattern or formula is not undeniable, a linear model does not seem to be suggested by these results.

Also, it is important to note that the first 9 trials have been cropped out to represent the steady state represented by trials 10 through 30. If the cache effect were included, these points would skew the distribution higher. The maximum would be the most affected, while the median and lower ordinal measurements would be expected to be minimally affected.

Figure 15. Variation in RAM over Workload E for the virtual machine clusters. The vertical axis shows the overall execution time in milliseconds. Each group along the x-axis runs the number of nodes in each cluster of interest. In each set, in order from left to right, the 1GB RAM is represented by blue, 2GB RAM by orange, and 4GB by green. Note that only trials 10 through 30 are represented.

The results from Workload C echo the pattern painted by Workload A. This would seem to indicate the the read operation dominates the scalability of the workload.

The results from Workload E also seem to echo the pattern painted by Workloads A and C for all examined cases, except for the 1GB RAM, 1 node case. From Figure 16, where the zoomed in view crops out the 1GB RAM, 1 node case, it appears that the results from Workload E seem to follow the same pattern. In Figure 15, it is clear that the 1GB RAM, 1 node case stands out from the rest of the results.

Based on what is seen here, further investigation may add insight into what is going on here. In order to rule out interference on the CPU, another experiment was run, but this time with an increase in granularity. Naturally, one does not purchase

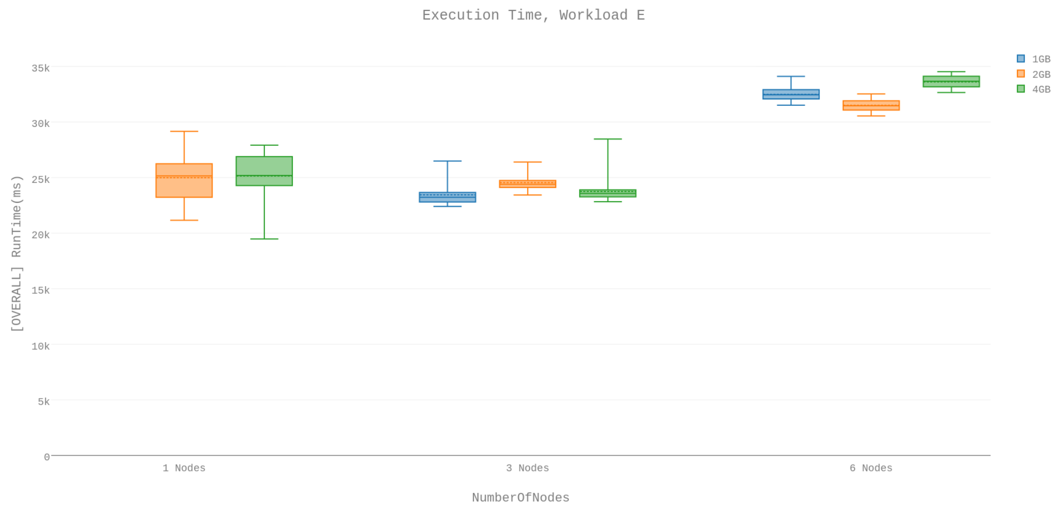

Figure 16. Variation in RAM over Workload E for the virtual machine clusters, zoomed in, cropping out the 1-node cluster with 1 GB of RAM. The vertical axis shows the overall execution time in milliseconds. Each group along the x-axis runs the number of nodes in each cluster of interest. In each set, in order from left to right, the 1GB RAM is represented by blue, 2GB RAM by orange, and 4GB by green. Note that only trials 10 through 30 are represented. Each solid horizontal line depicts the maximum, 75 percent quartile, 50 percent quartile, 25 percent quartile, and minimum for the specified data set. Any dots present represent outliers, which for this graphing method, are any values exceeding three standard deviations from the mean. A dotted horizontal line represents the mean.

RAM in such incremental amounts, but this gives insight into what it could be. The results are in Figure 17.

For the purposes of simulation, and in the absence of probable cause, the only conclusion that can be reached is that these results indicate a limit to which the simulated environment can serve as an apt analagous system.

Workload I shows distinction but stil lacks a satisfactory linear pattern. These results do seem to suggest that the transition from 2GB of RAM to 4GB of RAM implies a reliable benefit to performance. In contrast, the results seem to fail to suggest a reliable benefit increasing RAM from 1GB to 2GB for the conditions in these tests.

Determining the exact cause for the differentials between 2GB and 4GB is beyond the scope of this paper. However, if future work is merited, the next step would be to

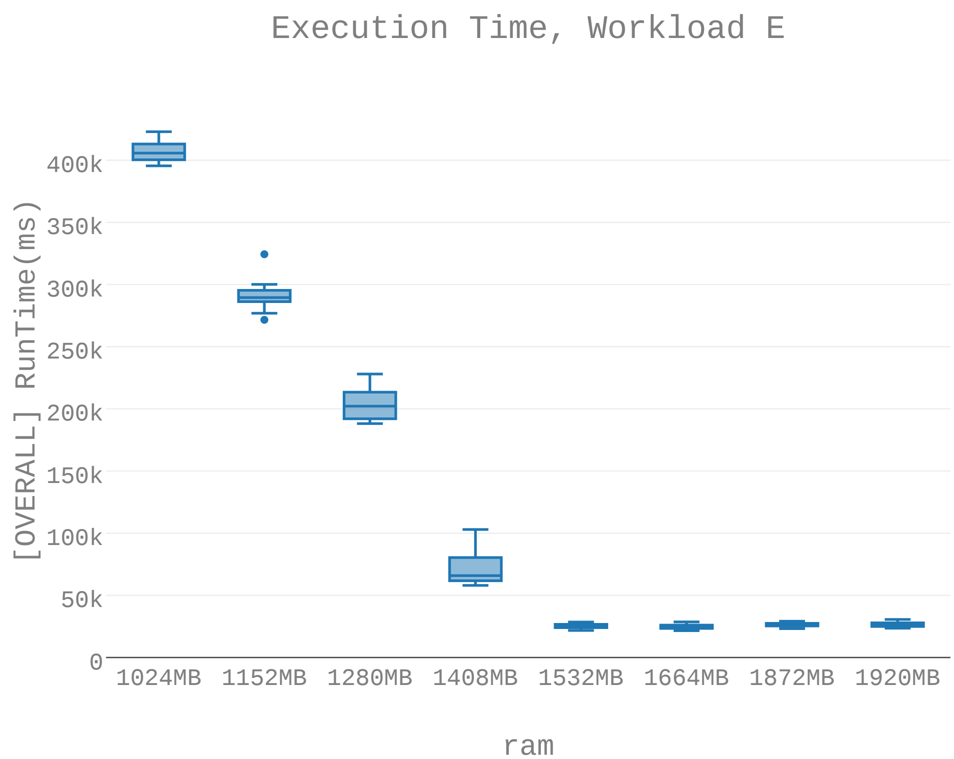

Figure 17. Variation in RAM over Workload E for the virtual machine 1-node cluster. This is a more granular approach to understand the transition between 1GB (1024MB) and 2GB (2048MB). Note there appears to be a linear pattern from 1024 MB to 1408 MB, and then the curve flattens between 1532 MB onward. This suggests a critical point that bounds the domain for which the virtual machine linearly models its hardware analogy, and in possible future work, would assist in a root cause analysis or may also be a basis to compare future iterations of Cassandra. Each solid horizontal line depicts the maximum, 75 percent quartile, 50 percent quartile, 25 percent quartile, and minimum for the specified data set. Any dots present represent outliers, which for this graphing method, are any values exceeding three standard deviations from the mean. The mean is omitted from this graph.

test compaction. If compaction is indeed a dominant factor, one could increase the operation count from 10,000 to a much higher amount, and expect the execution times for the 4GB cases for 2 node clusters up through 6 node clusters to increase in variation. Another way to test for compaction would be to keep compaction parameters absolutely constant in the Cassandra configuration file.

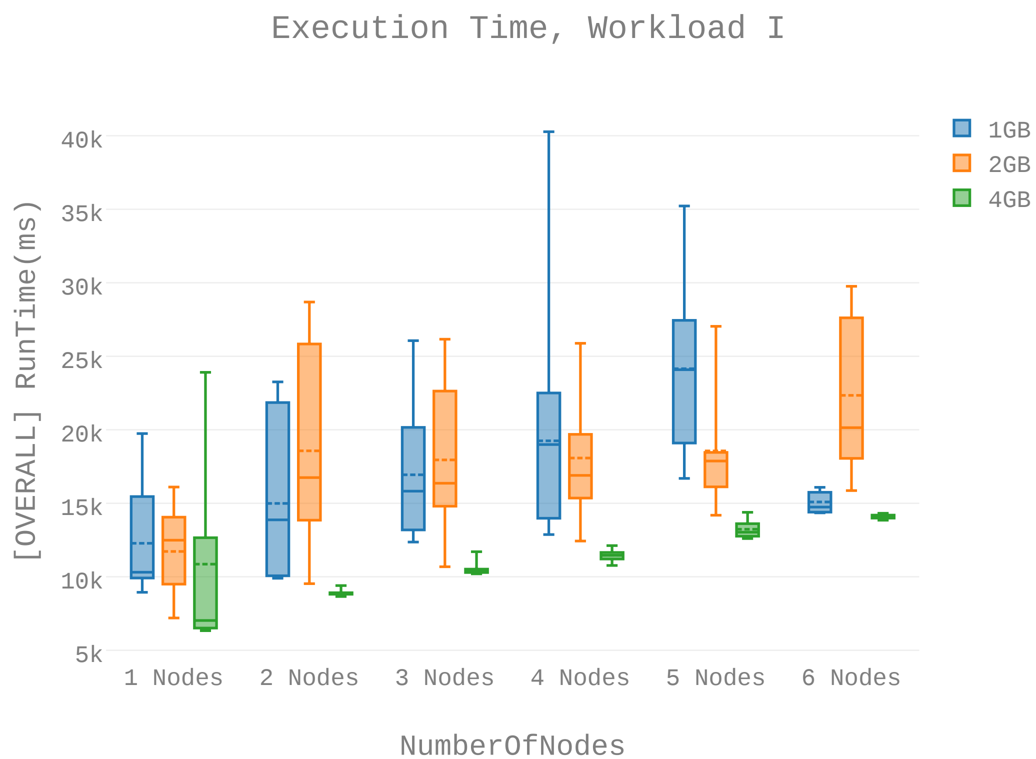

Figure 18. Variation in RAM over Workload I for the virtual machine clusters, zoomed in, cropping out the 1-node cluster with 1 GB of RAM. The vertical axis shows the overall execution time in milliseconds. Each group along the x-axis runs the number of nodes in each cluster of interest. In each set, in order from left to right, the 1GB RAM is represented by blue, 2GB RAM by orange, and 4GB by green. Note that only trials 10 through 30 are represented. Each solid horizontal line depicts the maximum, 75 percent quartile, 50 percent quartile, 25 percent quartile, and minimum for the specified data set. Any dots present represent outliers, which for this graphing method, are any values exceeding three standard deviations from the mean. A dotted horizontal line represents the mean.

4.2 Wired v. Wireless

In all four workloads: A in Figure 19, C in Figure 20, E in Figure 21 and I in Figure 22, the results seem to suggest that the scalability suffers upon the wireless 802.11 protocol as opposed to the wired Ethernet protocol.

For workloads A, C, and E, an oscillation is clear on the wireless protocol, where 1, 3, and 5 node clusters seem to outperform 2, 4, and 6 node clusters. However, an underlying factor is not clear here. This oscillation is neither advertised in doc-

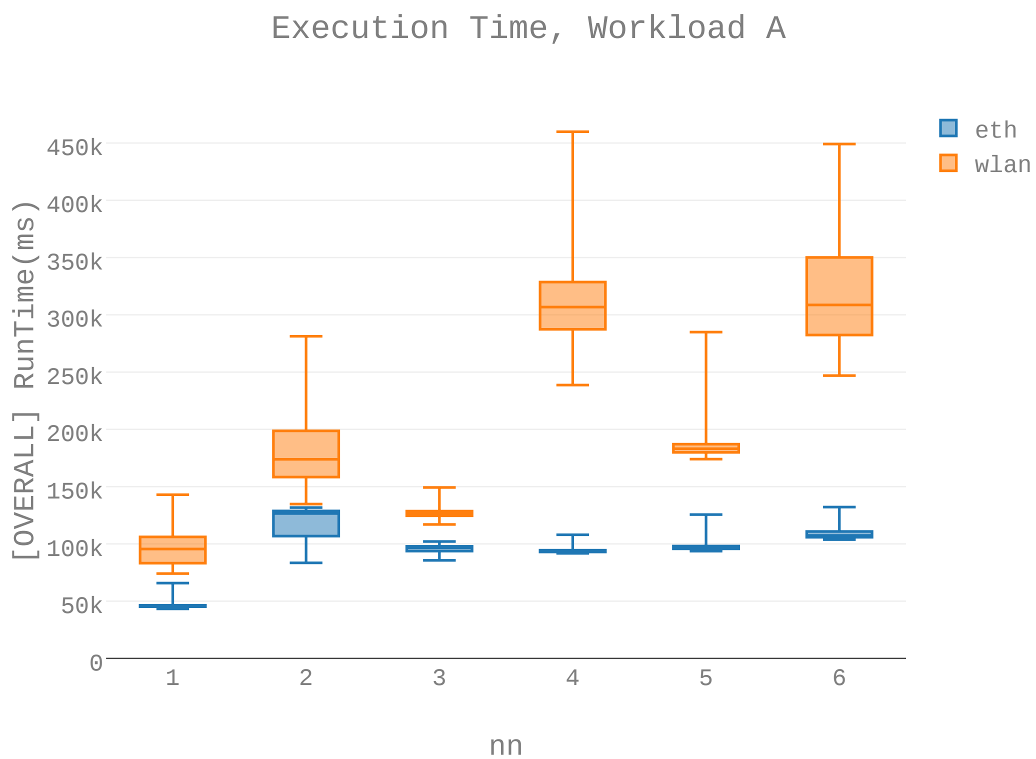

Figure 19. For 21 trials of 10,000 operations each of Workload A, this graph represents the execution time of a wired Ethernet cluster (eth) alongside the execution time of a wireless 802.11 cluster (wlan). On the vertical axis is execution time plotted over the number of nodes (nn) in the cluster under test on the horizontal axis. The wireless LAN, depicted in orange, ends up being the upper box for each cluster size. Each solid horizontal line depicts the maximum, 75 percent quartile, 50 percent quartile, 25 percent quartile, and minimum for the specified data set. Any dots present represent outliers, which for this graphing method, are any values exceeding three standard deviations from the mean. The mean is not depicted in this graph.

umentation nor known to be by design. Given that this oscillation does not extend to the results in the Ethernet analogy, this seems to suggest that this oscillation is characteristic of the wireless 802.11 protocol somehow. In turn, this brings attention to the inherent difference in handling collisions, as wireless uses collision avoidance while Ethernet uses collision detection.

Furthermore, as this oscillating pattern does not seem to present itself in Workload I, it further seems to suggest that three factors: workload, cluster size, and the 802.11 networking, interact to render the oscillating effect. If merited, future work

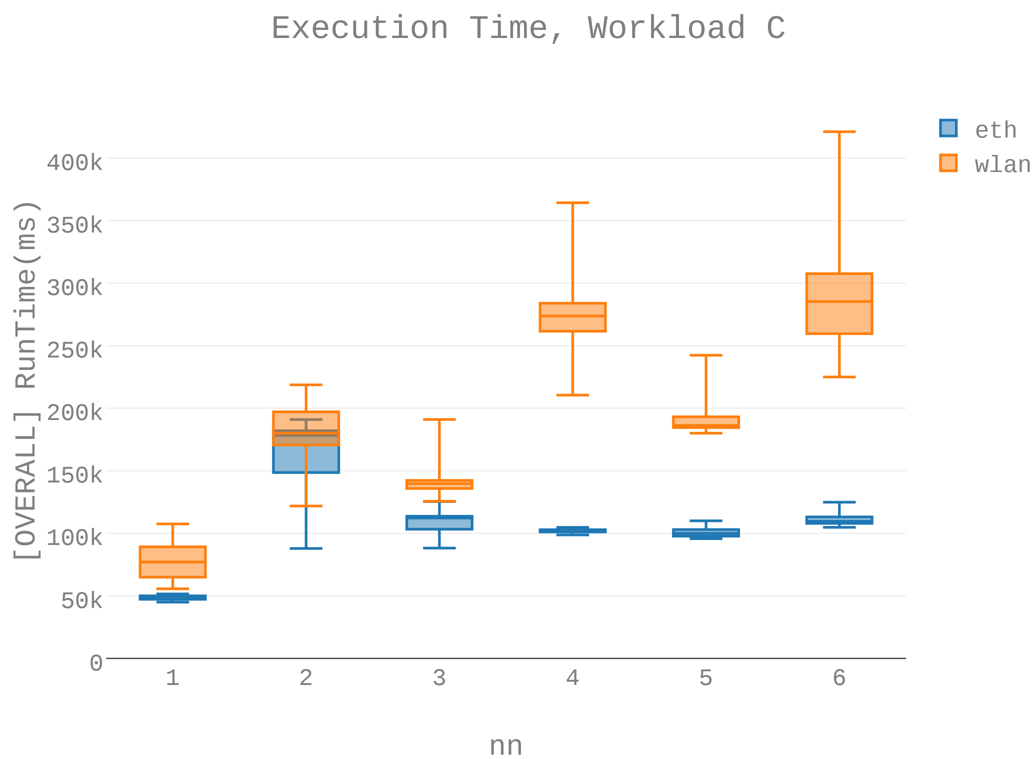

Figure 20. For 21 trials of 10,000 operations each of Workload C, this graph represents the execution time of a wired Ethernet cluster (eth) alongside the execution time of a wireless 802.11 cluster (wlan). On the vertical axis is execution time plotted over the number of nodes (nn) in the cluster under test on the horizontal axis. The wireless LAN, depicted in orange, ends up being the upper box for each cluster size. Each solid horizontal line depicts the maximum, 75 percent quartile, 50 percent quartile, 25 percent quartile, and minimum for the specified data set. Any dots present represent outliers, which for this graphing method, are any values exceeding three standard deviations from the mean. The mean is not depicted in this graph.

could include varying the read proportion of the workload to a more granular extent. However, it may be worth noting at this point that an interested application developer may also want to consider the value in trying another database other than Cassandra for a sufficiently read-heavy workload.

For workload E, some outliers, reporting extremely short execution times, have been eliminated from the sample. These are probably errors due to timeout in the six-node case. These errors may suggest that conditions are reaching a limit in feasibility, and if Workload E is desired to this extent, then additional measures may have to be

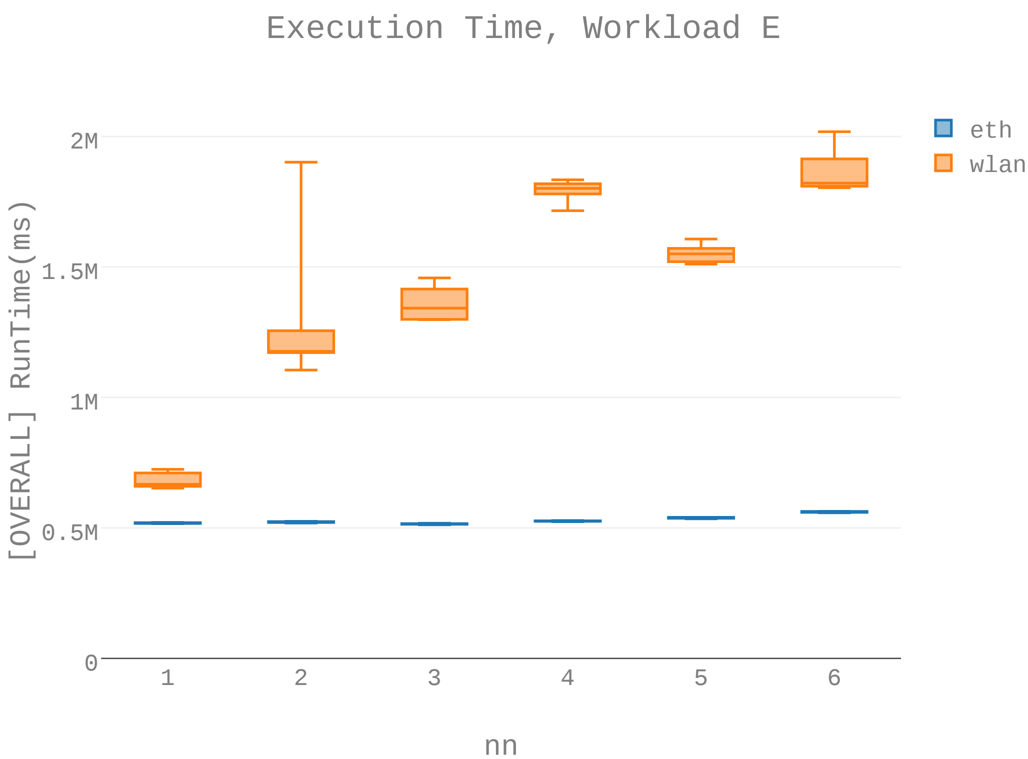

Figure 21. For 21 trials of 10,000 operations each of Workload E, this graph represents the execution time of a wired Ethernet cluster (eth) alongside the execution time of a wireless 802.11 cluster (wlan). On the vertical axis is execution time plotted over the number of nodes (nn) in the cluster under test on the horizontal axis. The wireless LAN, depicted in orange, ends up being the upper box for each cluster size. Each solid horizontal line depicts the maximum, 75 percent quartile, 50 percent quartile, 25 percent quartile, and minimum for the specified data set. Any dots present represent outliers, which for this graphing method, are any values exceeding three standard deviations from the mean. The mean is not depicted in this graph.

taken in order to render a characterization.

4.3 Hardware v. Virtual

The median results from the Raspberry Pi Ethernet cluster, the 1GB virtual machine, and the results from [1] can be observed in the following figures: Workload A in Figure 23, Workload C in Figure 24, Workload E in Figure 25, and Workload I in Figure 26.

Since parallel experiments were done on a realistic system in [1], it was of inter-

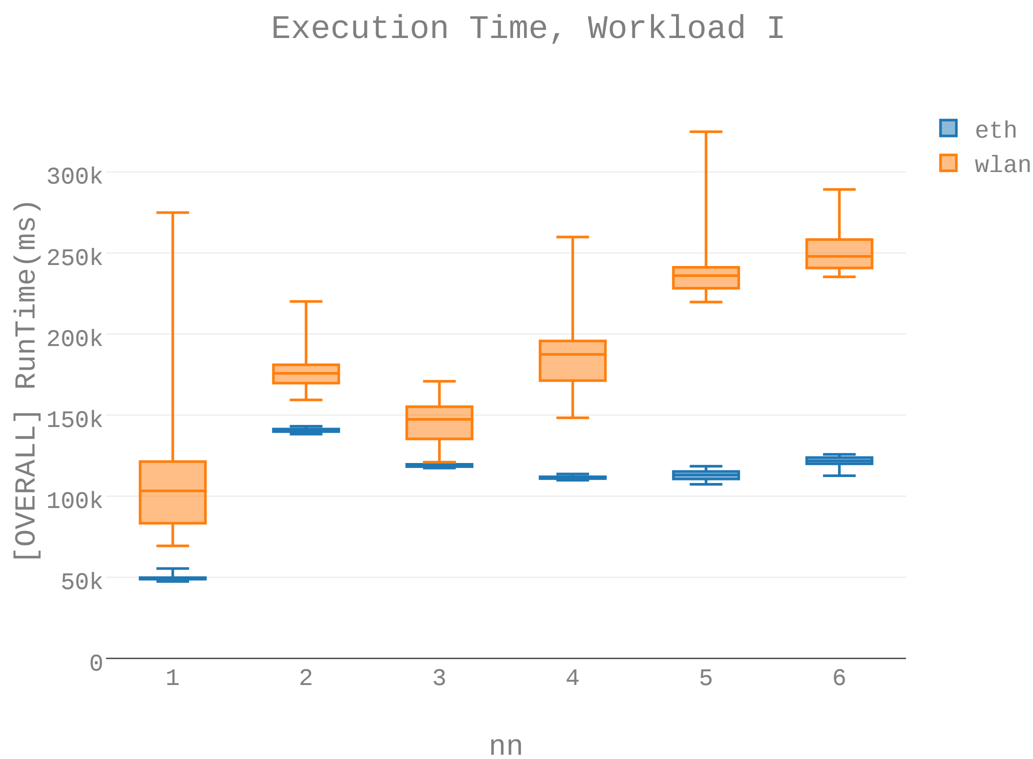

Figure 22. For 21 trials of 10,000 operations each of Workload I, this graph represents the execution time of a wired Ethernet cluster (eth) alongside the execution time of a wireless 802.11 cluster (wlan). On the vertical axis is execution time plotted over the number of nodes (nn) in the cluster under test on the horizontal axis. The wireless LAN, depicted in orange, ends up being the upper box for each cluster size. Each solid horizontal line depicts the maximum, 75 percent quartile, 50 percent quartile, 25 percent quartile, and minimum for the specified data set. Any dots present represent outliers, which for this graphing method, are any values exceeding three standard deviations from the mean. The mean is not depicted in this graph.

est as to whether the differential between the performance of [1]'s system and the Raspberry Pi cluster remained similar across workloads. By visually examining the graphs, one might surmise that it might be the case that different configurations may have different advantages over each other based on workload makeup.

There are a few observations here that do not map to a clear explanation. It is not clear why, in the 1-node case, the Raspberry Pi cluster seems to outperform the reference in [1], while in the 3-node cluster case, the Raspberry Pi cluster seems to underperform the reference in [1]. It would be expected that whatever configurations

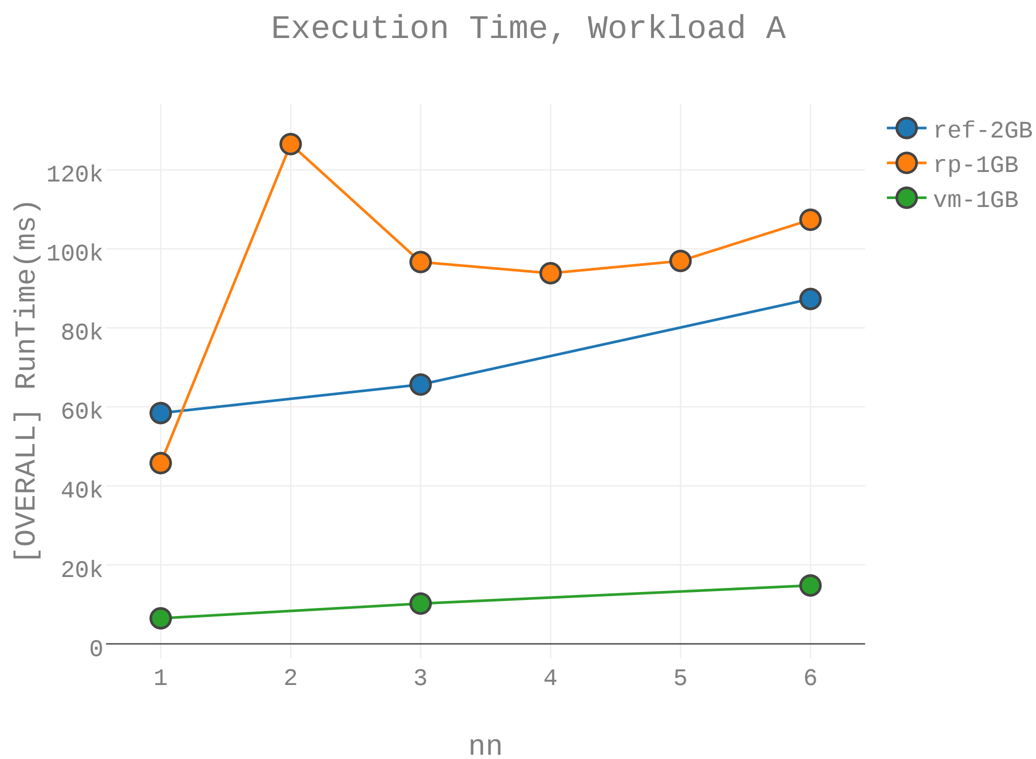

Figure 23. With trials of 10,000 operations each of Workload A, this graph represents the median execution time of the Raspberry Pi cluster (rp-1GB) for 21 trials, the median execution time for the virtual machine cluster set to 1GB of RAM (vm-1GB) for 21 trials, and, the imported value from past work (ref-2GB) in [1]. On the vertical axis is execution time plotted over the number of nodes (nn) in the cluster under test on the horizontal axis. The connecting lines are merely to assist in readability of the plotted points; only the dots represent values.

are there, one should outperform the other in all cases. Of course, [1] only reports one value, so it is difficult to do any formal analysis of the distributions with a sample size of one. Although, a difference in a one-node cluster versus multiple nodes does imply a transition to replication, and this may be a factor that may make the transition from one-node operation to multiple-node operation more difficult to predict. The second crossover for Workload C, over the 3-node cluster and the 6-node cluster, depicted in Figure 24, may represent an artifact of the limited sample size. Thus, the crossover seen in Figures 23 and 24 do not represent any significant finding.

These graphs, with the exception of the anomoly in Workload E discussed earlier,

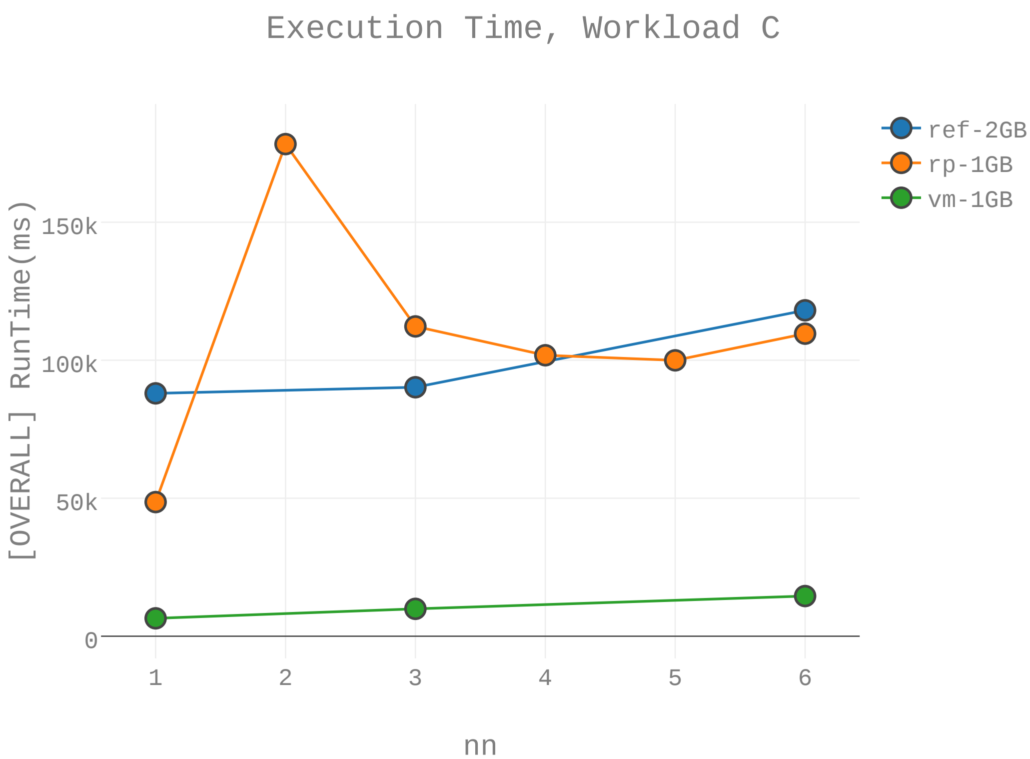

Figure 24. With trials of 10,000 operations each of Workload C, this graph represents the median execution time of the Raspberry Pi cluster (rp-1GB) for 21 trials, the median execution time for the virtual machine cluster set to 1GB of RAM (vm-1GB) for 21 trials, and, the imported value from past work (ref-2GB) in [1]. On the vertical axis is execution time plotted over the number of nodes (nn) in the cluster under test on the horizontal axis. The connecting lines are merely to assist in readability of the plotted points; only the dots represent values.

seem to suggest that that there may indeed be a predicatable relationship, some linear factor that could be applied to the virtual machine cluster to predict the Raspberry Pi. Workload A, C, and E give the strongest suggestion, while Workload I has a slightly divergent pattern and may have to tolerate more error.

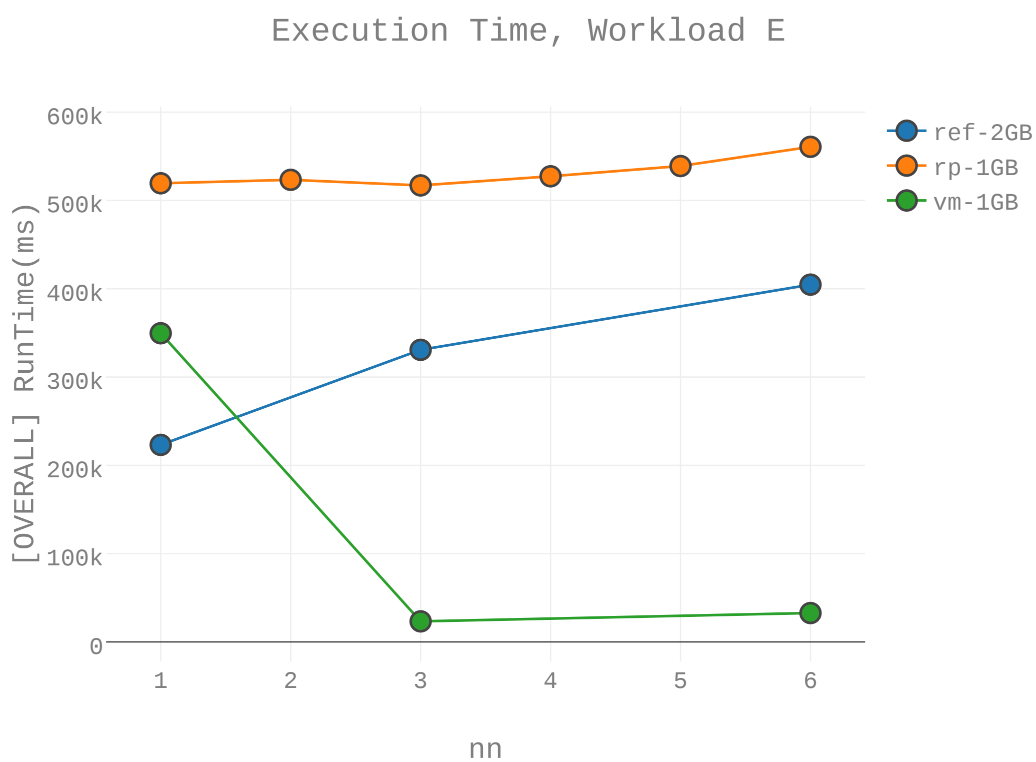

Figure 25. With trials of 10,000 operations each of Workload E, this graph represents the median execution time of the Raspberry Pi cluster (rp-1GB) for 21 trials, the median execution time for the virtual machine cluster set to 1GB of RAM (vm-1GB) for 21 trials, and, the imported value from past work (ref-2GB) in [1]. On the vertical axis is execution time plotted over the number of nodes (nn) in the cluster under test on the horizontal axis. The connecting lines are merely to assist in readability of the plotted points; only the dots represent values.

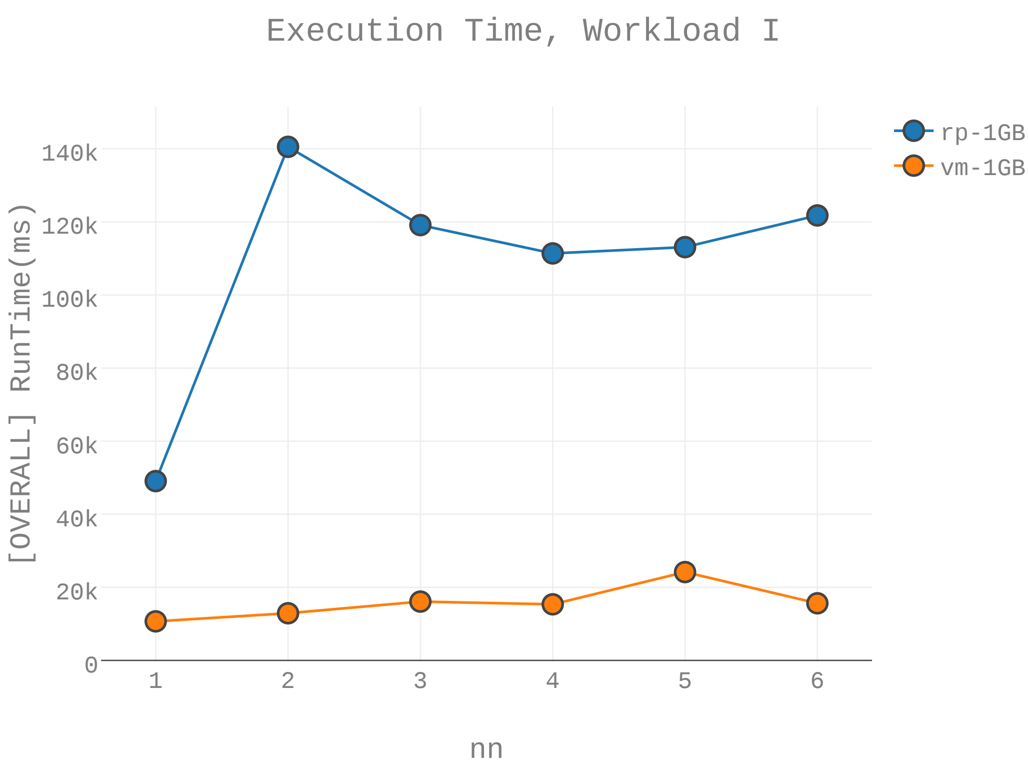

Figure 26. With trials of 10,000 operations each of Workload I, this graph represents the median execution time of the Raspberry Pi cluster (rp-1GB) for 21 trials, the median execution time for the virtual machine cluster set to 1GB of RAM (vm-1GB) for 21 trials, and, the imported value from past work (ref-2GB) in [1]. On the vertical axis is execution time plotted over the number of nodes (nn) in the cluster under test on the horizontal axis. The connecting lines are merely to assist in readability of the plotted points; only the dots represent values.

V. Conclusion and Future Work

5.1 Conclusion

The light weight of the Raspberry Pi and like devices represents a mobility and flexibility that, in certain contexts, gives these nodes potential competitive advantage over a heavy server-type node. An open question for an application considering the variance in potential nodes is how the less capable CPU may affect performance.

This work imported some results from [1] to form a set of reference points, reference points that for all practical purposes represent a likely candidate platform and network upon which one would nominally use Cassandra. The fact that the machines in [1] were restricted to 2GB was a bonus for this work, as these nodes hinted at getting slightly closer to the lucrative domain of interest: in-situ IoT storage. Using these references and some controlled variation within this work, the tests done here can start to paint a picture of what one can expect porting a distributed database like Cassandra to a platform like the Raspberry Pi 2 or 3. To summarize some of the data collected, this work was able to make the following determinations with respect to the research questions 1 through 4 posed earlier:

5.1.1 Finding 1.

For 21 trials, each consisting of 10,000 operations of workload A (50 percent reads and 50 percent updates) and performed over 1, 3, and 6-node configurations, the greatest absolute deviation from the reference to the Raspberry Pi configuration was found to be 34851.0 milliseconds, or about 35 seconds. Second, for 10,000 operations of this workload and over 1, 2, 3, 4, 5, and 6-node configurations, the greatest absolute deviation from any given trial in the wired configuration to an analogous trial using wireless configuration was found to be 367262.0 milliseconds, or about 6.1 minutes.

5.1.2 Finding 2.

For 21 trials, each consisting of 10,000 operations of workload C (100 percent reads) and performed over 1, 3, and 6-node configurations, the greatest absolute deviation from the reference to the Raspberry Pi configuration was found to be 43025.0 milliseconds, or about 43 seconds. Second, for 10,000 operations of this workload and over 1, 2, 3, 4, 5, and 6-node configurations, the greatest absolute deviation from any given trial in the wired configuration to an analogous trial using wireless configuration was found to be 211759.0 milliseconds, or about 3.5 minutes.

5.1.3 Finding 3.

For 21 trials, each consisting of 10,000 operations of workload E (close to 100 percent scans) and performed over 1, 3, and 6-node configurations, the greatest absolute deviation from the reference to the Raspberry Pi configuration was found to be 301019.0 milliseconds, or about 5 minutes. Second, for 10,000 operations of this workload and over 1, 2, 3, 4, 5, and 6-node configurations, the greatest absolute deviation from any given trial in the wired configuration to an analogous trial using wireless configuration was found to be 1463335.0 milliseconds, or about 24 minutes.

5.1.4 Finding 4.

For 21 trials, each consisting of 10,000 operations of workload I (99 percent inserts, 1 percent reads) no analogous reference was extracted from literature. However, for 10,000 operations of this workload and over 1, 2, 3, 4, 5, and 6-node configurations, the greatest absolute deviation from any given trial in the wired configuration to an analogous trial using wireless configuration was found to be 227459.0 milliseconds, or about 3.8 minutes.

One of the selling points of a distributed system, and thus a distributed database, is the option to increase the likelihood of survival of data by spreading nodes geographically. While servers can and are distributed geographically as well,

5.2 Future Work

As was alluded to in the introduction, this research seeks to support a number of possible future endeavors.

5.2.1 Generalized Model.

5.2.1.1 Overview.

Despite the complexities of hardware, a generalized model may assert that a node can be abstracted to RAM, rated I/O speeds, and the processor speed. The experiments done in this report do not vary these parameters sufficiently to characterize an empirical model, but they provide a data point as well as a framework for developing other experiments to further refine this model. This work has already suggested that the amount of memory may not be critical or useful in predicting hardware performance, leaving I/O speeds and processor speed. This work grouped them all as one.

There has been a lot of work in trying to characterize networks, but really for aOne of the selling points of a distributed system, and thus a distributed database, is the option to increase the likelihood of survival of data by spreading nodes geographically. While servers can and are distributed geographically as well, model like this, this proposition suggests the network can be characterized by the mean ping team between nodes.

5.2.1.2 Experiments that would Refine This Model.

Naturally, trying other databases, like MongoDB, in Cassandra's place is one way to achieve further confidence. In part, the motivation behind using the YCSB was its portability for testing a multitude of distributed database applications. Although the workloads for Cassandra were varied, and different databases are typically optimized and designed for particular workloads, a different database or databases would increase qualitative confidence that the model can be extended to a database not explicitly tested.

In addition, the absolute values in the results of this work do not necessarily represent Cassandra at its best. In the interest of replication, default values were used much more often than not. Cassandra boasts a multitude of parameters with which one can vary in attempt to optimize performance, almost to the point where you are almost guaranteed to be running it sub-optimally without employing a Cassandra expert.

5.2.2 Wifi Collection, Mapping and Crowd Detection.

Naturally, another step forward is developing or adapting an application that puts the system to the test. The application could be compared to the various workloads that were executed to verify if one of the workloads accurately represents an IoT application.

Bibliography

1. V. Abramova, J. Bernardino, and P. Furtado, "Testing Cloud Benchmark Scalability with Cassandra," *2014 IEEE World Congress on Services*, pp. 434–441, 2014. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6903301

2. "Raspberry Pi 2 Model B." [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-2-model-b/

3. "LENOVO THINKSERVER RD650." [Online]. Available: http://cc.cnetcontent.com/inlinecontent/production/13/13e03a308886/cnet{_}4d4c{_}doc.pdf

4. Lu Tan and Neng Wang, "Future internet: The Internet of Things," *2010 3rd International Conference on Advanced Computer Theory and Engineering(ICACTE)*, pp. 5–376, 2010. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5579543

5. "Yahoo Cloud Serving Benchmark." [Online]. Available: https://research.yahoo.com/news/yahoo-cloud-serving-benchmark

6. "SensePost — Snoopy: a distributed tracking and profiling framework." [Online]. Available: https://www.sensepost.com/blog/2012/snoopy-a-distributed-tracking-and-profiling-framework/

7. "Aircrack-ng." [Online]. Available: https://www.aircrack-ng.org/

8. "AirSnort Homepage." [Online]. Available: http://faculty.ccri.edu/jbernardini/JB-Website/ETEK1500/LinuxTools/AirSnort{%}20Homepage.htm

9. I. Rose and M. Welsh, "Mapping the urban wireless landscape with Argos," in *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*

- *SenSys '10*. New York, New York, USA: ACM Press, 11 2010, p. 323. [Online]. Available: http://dl.acm.org/citation.cfm?id=1869983.1870015

10. "WiGLE: Wireless Network Mapping." [Online]. Available: https://wigle.net/

11. "HeatMapper — Free Wi-Fi coverage mapping software for homes and small offices." [Online]. Available: http://www.ekahau.com/wifidesign/ekahau-heatmapper

12. G. Castignani, A. Lampropulos, A. Blanc, and N. Montavont, "Wi2Me: A mobile sensing platform for wireless heterogeneous networks," *Proceedings - 32nd IEEE International Conference on Distributed Computing Systems Workshops, ICDCSW 2012*, pp. 108–113, 2012.

13. E. Keeble, "Casual Encounters (2013)." [Online]. Available: http://edwardkeeble.com/portfolio/casual-encounters/

14. S. Haigh, "Tracking people via Wifi (even when not connected)." [Online]. Available: https://www.crc.id.au/tracking-people-via-wifi-even-when-not-connected/

15. B. Bonne, A. Barzan, P. Quax, and W. Lamotte, "WiFiPi: Involuntary tracking of visitors at mass events," *2013 IEEE 14th International Symposium on a World of Wireless, Mobile and Multimedia Networks, WoWMoM 2013*, 2013.

16. L. Schauer, M. Werner, and P. Marcus, "Estimating Crowd Densities and Pedestrian Flows Using Wi-Fi and Bluetooth," *Proceedings of the 11th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, pp. 171–177, 2014.

17. R. Schiphorst, "Blue Mark Innovations." [Online]. Available: https://bluemark.io/aboutus/

18. C. Chilipirea, A.-c. Petre, and C. Dobre, "Presumably simple : monitoring crowds using WiFi," no. 1.

19. J. Pang, B. Greenstein, R. Gummadi, S. Srinivasan, and D. Wetherall, "802. 11 User Fingerprinting," *Proceedings of the 13th Annual ACM International Conference on Mobile Computing and Networking*, vol. 9, pp. 99–110, 2007.

20. M. Chernyshev, C. Valli, and P. Hannay, "Service Set Identifier Geolocation for Forensic Purposes: Opportunities and Challenges," *International Conference on System Sciences*, 2016.

21. M. Cunche, M. A. Kaafar, and R. Boreli, "Linking wireless devices using information contained in Wi-Fi probe requests," *Pervasive and Mobile Computing*, vol. 11, pp. 56–69, 2014. [Online]. Available: http://dx.doi.org/10.1016/j.pmcj.2013.04.001

22. S. Du, J. Hua, Y. Gao, and S. Zhong, "EV-Linker: Mapping eavesdropped Wi-Fi packets to individuals via electronic and visual signal matching," *Journal of Computer and System Sciences*, vol. 82, no. 1, pp. 156–172, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0022000015000677

23. M. Cunche and R. Boreli, "I know who you will meet this evening! Linking wireless devices using Wi-Fi probe requests," in *2012 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. IEEE, 6 2012, pp. 1–9. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6263700

24. A. D. Luzio, A. Mei, and J. Stefa, "Mind Your Probes : De-Anonymization of Large Crowds Through Smartphone WiFi Probe Requests," in *IEEE INFOCOM 2016*, 2016.

25. A. B. M. Musa and J. Eriksson, "Tracking Unmodified Smartphones Using Wi-Fi Monitors," in *SynSys*, 2012.

26. "Raspberry Pi 2 on sale now at $35 - Raspberry Pi." [Online]. Available: https://www.raspberrypi.org/blog/raspberry-pi-2-on-sale/

27. "BeagleBoard.org - black." [Online]. Available: https://beagleboard.org/black

28. "Banana Pi M3." [Online]. Available: http://www.banana-pi.org/m3.html

29. "CHIP." [Online]. Available: https://nextthing.co/pages/chip

30. "Parallella." [Online]. Available: https://www.parallella.org/

31. D. G. Waddington and C. Lin, "A Fast Lightweight Time-Series Store for IoT Data," 2016.

32. C. Baun, "Mobile clusters of single board computers: an option for providing resources to student projects and researchers." *SpringerPlus*, vol. 5, no. 1, p. 360, 2016. [Online]. Available: http://www.scopus.com/inward/record.url?eid=2-s2.0-84961794768{&}partnerID=tZOtx3y1

33. 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," *Proceedings - International Conference on Distributed Computing Systems*, pp. 108–112, 2013.

34. J. Kiepert, "Creating a Raspberry Pi-Based Beowulf Cluster," p. 16, 2013.

35. P. J. E. Velthuis, "Small Data Center using Raspberry Pi 2 for Video Streaming," *23th Twente Student Conference on IT*, 2015.

36. B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, "Benchmarking cloud serving systems with YCSB," *Proceedings of the 1st ACM symposium on Cloud computing - SoCC '10*, p. 143154, 2010.

37. "CassandraHardware - Cassandra Wiki." [Online]. Available: https://wiki.apache.org/cassandra/CassandraHardware

38. A. Lakshman and P. Malik, "Cassandra - A Decentralized Structured Storage System," *ACM SIGOPS Operating Systems Review*, vol. 44, no. 2, p. 35, 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=1773912.1773922

39. "Apache Cassandra Use Cases." [Online]. Available: http://www.planetcassandra.org/apache-cassandra-use-cases/

40. V. Abramova and J. Bernardino, "NoSQL databases: MongoDB vs cassandra," *Proceedings of the International C* Conference on Computer Science and Software Engineering, ACM 2013*, pp. 14–22, 2013.

41. B. Van Ryswyk, "Multi-Datacenter Cassandra on 32 Raspberry Pi's." [Online]. Available: http://www.datastax.com/dev/blog/32-node-raspberry-pi-cassandra-cluster

42. J. Sercel, "Cassandra on RaspberryPi 2 Medium." [Online]. Available: https://medium.com/@johnsercel/cassandra-on-raspberrypi-2-a84602953b23{#}.2mywozbod

43. "Cassandra Parameters for Dummies." [Online]. Available: http://www.ecyrd.com/cassandracalculator/

44. Datastax, "The cassandra-stress tool." [Online]. Available: https://docs.datastax.com/en/cassandra/2.1/cassandra/tools/toolsCStress{_}t.html

45. J. F. Kurose and K. W. Ross, *COMPUTER NETWORKING A Top-Down Approach.*

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
3–24–2017	Master's Thesis	Aug 2015 — Mar 2017

4. TITLE AND SUBTITLE

AFIT/ENG THESIS:
CLOUD BENCHMARK TESTING OF CASSANDRA ON RASPBERRY PI FOR INTERNET OF THINGS CAPABILITY

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Daniel P. Richardson

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AFRL Information Directorate
POC: Hiren Patel,
Cyber ISR Program Manager
Mailstop: 525 Brooks Rd, Rome NY 13441
Email: hiren.patel@us.af.mil

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

—-

15. SUBJECT TERMS

LaTeX,Thesis

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Lt Col John Pecarina, PhD, AFIT/ENG
U	U	U	U	lxxxiii	19b. TELEPHONE NUMBER *(include area code)* (937) 255-3636, x4555; daniel.richardson@afit.edu

Standard Form 298 (Rev. 8–98)
Prescribed by ANSI Std. Z39.18