6-2010

Using Hadoop and Cassandra for taxi data analytics: A feasibility study

Alvin Jun Yong KOH
Singapore Management University, alvin.koh.2008@smu.edu.sg

Xuan Khoa NGUYEN
Singapore Management University, xknguyen@smu.edu.sg

C. Jason WOODARD
Singapore Management University, jason.woodard@olin.edu

6-2010

Using Hadoop and Cassandra for Taxi Data Analytics: A Feasibility Study

Alvin Jun Yong KOH
Singapore Management University, alvin.koh.2008@smu.edu.sg

Xuan Khoa NGUYEN
Singapore Management University, xknguyen@smu.edu.sg

C. Jason WOODARD
Singapore Management University, jason.woodard@olin.edu

Singapore Management University

School of Information Systems

Technical Paper Series

Using Hadoop and Cassandra for Taxi Data Analytics: A Feasibility Study

Alvin KOH Jun Yong
School of Information Systems
Singapore Management University
alvin.koh.2008@smu.edu.sg

NGUYEN Xuan Khoa
School of Information Systems
Singapore Management University
xknguyen@smu.edu.sg

C. Jason WOODARD
School of Information Systems
Singapore Management University
jwoodard@smu.edu.sg

15 June 2010

Using Hadoop and Cassandra for Taxi Data Analytics: A Feasibility Study

Abstract

This paper reports on a preliminary study to assess the feasibility of using the Open Cirrus Cloud Computing Research Testbed to provide offline and online analytical support for taxi fleet operations. In the study, we benchmarked the performance gains from distributing the offline analysis of GPS location traces over multiple virtual machines using the Apache Hadoop implementation of the MapReduce paradigm. We also explored the use of the Apache Cassandra distributed database system for online retrieval of vehicle trace data. While configuring the testbed infrastructure was straightforward, we encountered severe I/O bottlenecks in running the benchmarks due to the lack of local disk storage on the compute nodes. This design limitation severely impedes the analysis of large data sets using cloud computing technologies.

1. Introduction

Since 2007, researchers at the SMU School of Information Systems have collaborated with Singapore taxi companies to analyse GPS traces and trip data from their fleet of about 15,000 taxicabs. A data set of over 4 billion GPS observations from 150 million trips (about 300 GB in uncompressed form) has been amassed through this effort.

Several research projects now underway at SMU are seeking to use this data to improve the efficiency of Singapore's taxi fleet. A major bottleneck in these projects is the time required to run algorithms over the entire data set, which can take weeks or even months on a single machine. As a result, our published results have been limited to analysis on a day or week's worth of data at a time.

The main motivation for this study was to explore the feasibility of using the Open Cirrus Cloud Computing Research Testbed to break this bottleneck. Building on prior work by Shahfik Amasha (2009), we hoped to achieve near-linear scalability across 50–100 compute nodes, allowing us to analyse roughly an order of magnitude more data in an order of magnitude less time than possible on a single server in our lab.

Figure 1 illustrates what we call the "analytical frontier" defined by the scale of the data set and the speed of processing it. On a single server, it generally takes hours to perform a simple computational task (e.g., calculating the occupancy rate for each vehicle) on a few days' worth of data. In principle, the Open Cirrus testbed would allow us to perform the same task on a year's worth of data in the same amount of time, or the same amount of data in far less time — potentially even seconds, which would open up new possibilities for real-time interactive queries.

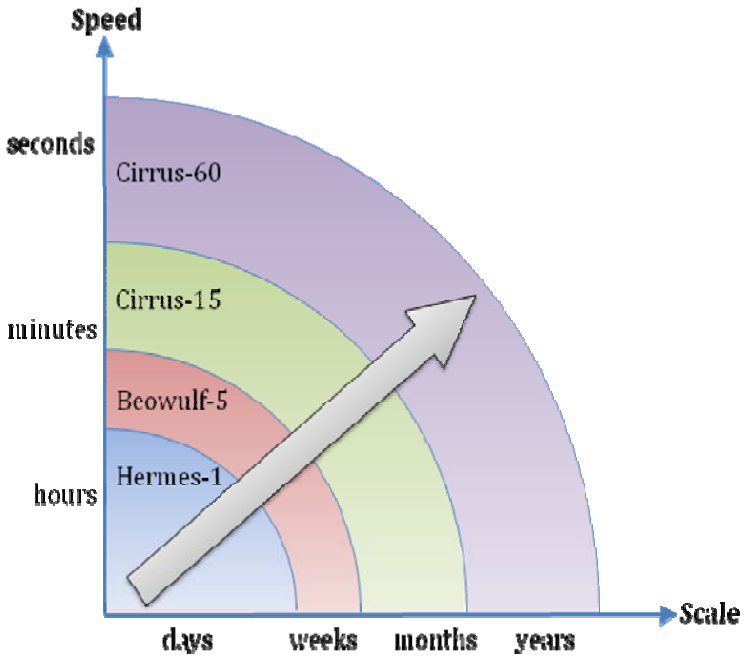

Figure 1: Using the Open Cirrus testbed to expand the analytical frontier

The elastic and scalable nature of cloud computing infrastructure offers great promise in realizing these ambitious goals. As such, the availability of the Open Cirrus testbed presented an ideal opportunity to assess the feasibility of using cloud-oriented technologies like Apache Hadoop and Cassandra for taxi data analytics.

2. Data and methods

The taxi fleet data set we used for the project includes GPS location traces and trip data. With about 15,000 company taxis generating several hundred million GPS observations per month, the raw data set amounts to over 300 GB for a full year. Table 1 provides a simplified representation of the location data used in this study.

Date Time	Vehicle Number	Driver ID	Longitude	Latitude	Speed	Status
01/03/2009 10:17:00	YYYYY	XXXXX	103.94063	1.32617	0	FREE

Table 1: A simplified representation of the taxi data set

In our lab, this data is stored in a PostgreSQL database on a single Linux server. While this setup has the advantage of being familiar to the researchers involved in the SMU Taxi Data Analytics project, it suffers from obvious scalability limitations. Moreover, most of our analysis to date has involved batch-mode access to the entire data set, which relational databases tend to do less efficiently than simple flat files. Furthermore, the

data set is effectively write-once (since new data is added at monthly intervals or less) and we do not require the transactional integrity afforded by modern RDBMS software. These factors suggested that the MapReduce paradigm may be more appropriate for our needs; see Table 2 for a comparison of MapReduce with the traditional RDBMS approach to data management.

	Traditional RDBMS	MapReduce
Data Size	Gigabytes	Petabytes
Access	Interactive & Batch	Batch
Updates	Read & write many times	Write once, read many times
Structure	Static schema	Dynamic schema
Integrity	High	Low
Scaling	Nonlinear	Linear

Table 2: Comparison of MapReduce and traditional RDBMS approaches (White, 2009)

Prior to our receiving a grant of compute and storage resources on the Open Cirrus testbed, Shahfik Amasha developed a benchmark for offline (batch-mode) analysis of the taxi data using the Apache Hadoop implementation of the MapReduce paradigm. He obtained results for a 5-node Hadoop cluster running on SMU's High Performance Computing infrastructure, which we use as a basis for comparison with the Open Cirrus results presented below. Apart from removing sensitive information to protect the privacy of taxi drivers, the benchmark code and data were identical to those used on the SMU HPC cluster.

Towards the end of the study we decided to explore a second cloud-oriented technology, the Apache Cassandra distributed database system, to assess the feasibility of using it for online (interactive) analysis of the taxi data. By this time our grant had expired, so we report results obtained on the SMU Beowulf cluster instead.

3. Offline benchmark using Hadoop

The offline benchmark consisted of Hadoop map and reduce tasks that sequentially processed the taxi location traces to identify state transitions (e.g., from idle to occupied or vice versa) and grouped the data into "chunks" of time in a given state. Several taxi-related projects at SMU have implemented similar routines independently — resulting in duplicated effort and redundant data storage — so it is a good example of a common use case that could benefit from a cloud computing approach.

3.1 Setup

For our Hadoop configuration on the Open Cirrus testbed, we used 1 head node with 4 CPUs and 8 GB RAM, and 6 slave nodes with 1 CPU and 2 GB RAM each. In our

experiments, the head node was always online as the Distributed File System namenode, with a varying number of slave nodes active.

We ran a series of experiments with the same MapReduce program, testing 1, 2, 4, 8, 16, 32, 256, 512 and 1024 MB data sets against a varying number of slave nodes (1–6). Three consecutive tests were run for each configuration, with their durations averaged for the final result.

3.2 Results

There are several notable patterns in our experiment results:

- As illustrated in Figure 2, performance was poor for small input sizes due to the fixed overhead of the Hadoop framework. This was expected, as Hadoop is designed for much larger data sets. As we scaled up the input size above 64 MB (the size of one Hadoop DFS block), we started to see reasonable throughput rates and speedups.

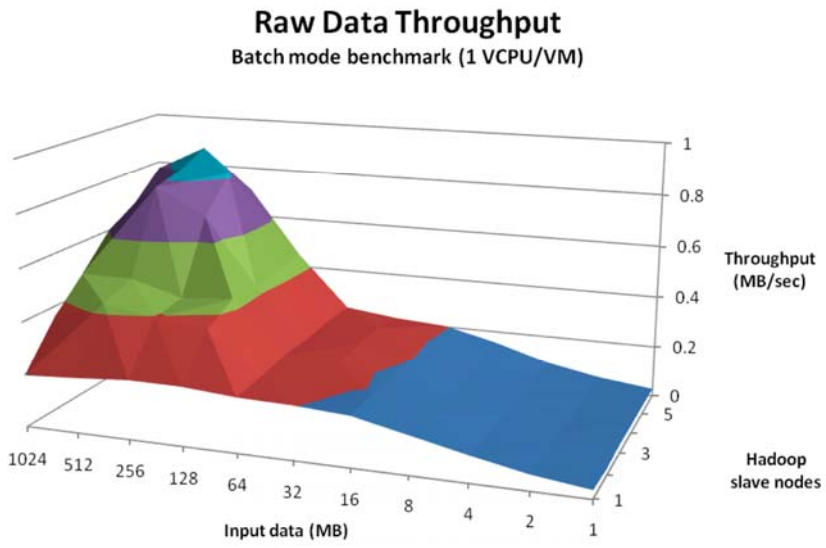

Figure 2: Raw data throughput on the Open Cirrus testbed

- However, these gains peaked at our 512 MB data set. At 1 GB, we observed diminishing performance returns. We were unable to obtain reliable data for larger input sizes due to network errors; this issue is discussed further below. This was much earlier than we had expected based on our experience with the SMU Beowulf cluster, which yielded nearly linear scaling through 20 GB, the maximum input size tested.

- As illustrated in Figure 3, we also observed substantially sub-linear performance gains from adding Hadoop slave nodes. These gains diminished as nodes were added. In the 1 GB case, for example, adding a second node increased throughput from about 0.20 to 0.36 MB/sec, i.e., 90% of the 0.40 MB/sec that would be

obtained in the ideal case of linear scaling. Adding a third node yielded only about 80% of linear scaling, and so on down to 66% for 6 nodes.

On an absolute basis, the maximum throughput on the Open Cirrus testbed was about 0.77 MB/sec. In our experiments at SMU, we observed throughput rates of well above 10 MB/sec on a 5-node cluster for a wide range of input sizes. Even a student's laptop computer yielded almost twice the throughput of the single-node configuration on the Open Cirrus testbed (0.41 vs. 0.21 MB/sec for 512 MB, 0.38 vs. 0.20 MB/sec for 1 GB).

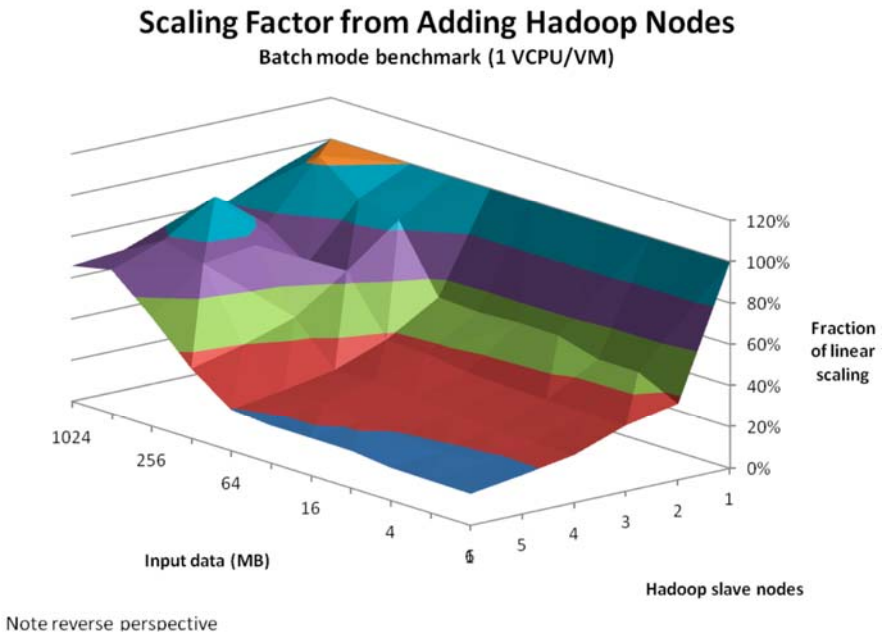

Figure 3: Performance gains from additional nodes on the Open Cirrus testbed

3.3 Analysis

Through additional analysis and discussions with the service provider, we obtained a straightforward explanation for the performance we observed on the Open Cirrus testbed: the fact that the compute nodes were not provisioned with physically local disk storage, but instead accessed a common storage resource via iSCSI. While we were informed of this fact during the testbed configuration phase of the project, we failed to anticipate how severely it would impact performance on our benchmark tests.

In hindsight, the logic is clear. The MapReduce paradigm in general (and Hadoop in particular) is designed to exploit data locality by bringing the processing to the data rather than vice versa. The Hadoop file system replicates data selectively to certain nodes, and Hadoop allocates MapReduce tasks to minimize additional data transfer. This is ideal for the type of computing infrastructure for which these technologies were first developed, namely large data centres full of commodity PCs.

When storage is centralized, however, the gains from parallel computation are diminished — or even totally negated — as all nodes try to access the central resource. Depending on the hardware configuration, bottlenecks may occur in the network itself,

the network interface on a storage device or compute node (especially if multiple virtual machines share a single physical interface), or a physical disk drive storing data needed by multiple MapReduce tasks. While this type of architecture may be appropriate for CPU-intensive applications (typical of the "grid computing" paradigm), it is ill-suited for the kind of data-intensive applications commonly associated with cloud computing.

4. Online benchmark using Cassandra

Since we were unable to complete our offline benchmark experiments on the Open Cirrus testbed due to the local storage issue, we decided to develop an online benchmark which would be less sensitive to the testbed's I/O performance. We focused on the task of obtaining a set of location traces for a given vehicle, which could be used in an interactive application for plotting a taxi's route history or identifying "hot spots" where demand for taxis exceeds supply.

This is typically a cumbersome task, since the entire data set must be searched to retrieve the traces for a single vehicle. We reasoned that it would be much faster if the data set could be cached in RAM in a distributed way. Since the testbed nodes are equipped with 32 MB of RAM each, the entire data set could in principle be cached on 10–12 nodes (less than 5% of the total nodes available on the testbed).

After exploring several "NoSQL" technologies, we decided to use the Apache Cassandra distributed database system (Lakshman and Malik, 2009) to test this idea. We developed a simple benchmark to measure the speed of retrieving all of the location trace records from 1000 randomly selected vehicles. As with the Hadoop benchmark, we evaluated the system's performance as a function of the size of the data set and the number of nodes on which the data is stored. In addition, we ran the benchmark twice consecutively to measure the performance gains from caching the vehicle traces. Although we focused on data reads, this strategy could easily be extended to benchmark concurrent reads and writes.

4.1 Setup

Unfortunately we were not able to complete the development of the Cassandra benchmark in time to run it on the Open Cirrus testbed, so we used 5 nodes on the SMU Beowulf cluster instead, each equipped with 2 CPUs and 4 GB RAM.

We configured Cassandra to use only one SuperColumn / ColumnFamily with default key caching (200,000 keys), no row caching and a replication factor of 1. The original data was stored on an NFS-mounted filesystem, and was imported into Cassandra in sizes of 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384 MB. Data importing and querying always took place on the same node, while the number of data storage nodes was varied from 1–5.

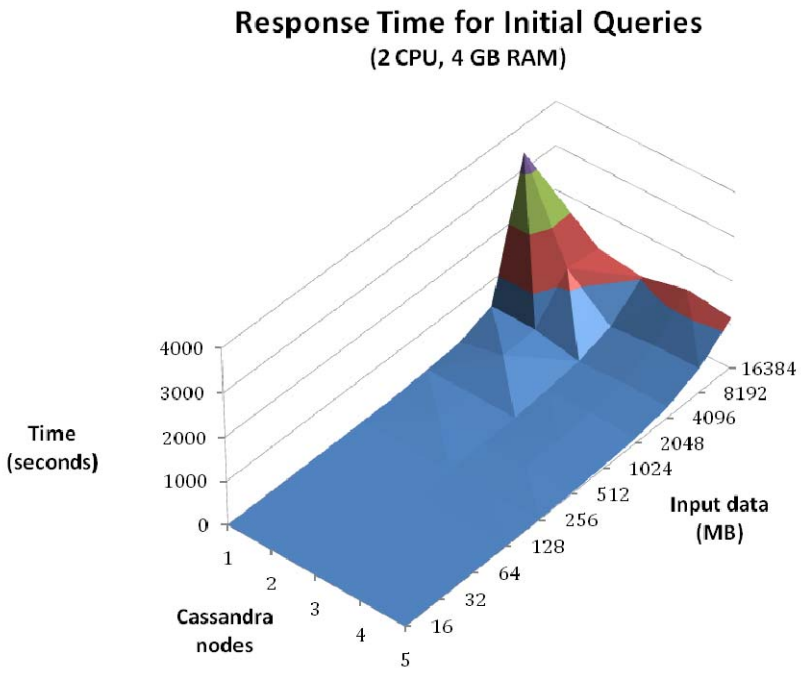

Figure 4: Performance gains from distributing data storage over several nodes

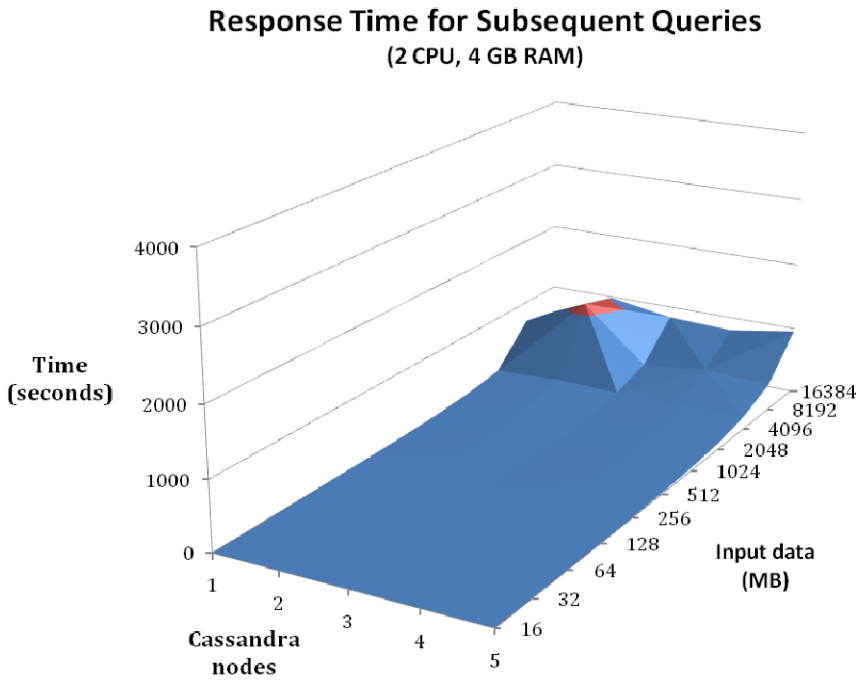

Figure 5: Additional performance gains from caching Cassandra keys

4.2 Results

Our main results for the Cassandra benchmark are shown in Figures 4 and 5 above.

- As expected, the initial query response time tends to increase for larger data sets (Figure 4). Adding nodes to the system tends to improve performance, especially for data sets larger than 1 GB. In particular, there seems to be a severe bottleneck when querying data sets of 8 and 16 GB in the 1- and 2-node configurations (i.e., where the amount of data exceeds the total RAM available), which is greatly improved by adding nodes.

- We also observed significant improvements in the performance of subsequent queries (Figure 5) due to Cassandra's key caching. Performance gains from adding nodes were diminished, however.

Figure 6 shows that the speed of data retrieval fluctuates substantially, with peaks around 20,000 records per second. We believe this result can be improved upon because we used a single thread to query the data, while Cassandra is designed to scale well with multiple concurrent reads and writes.

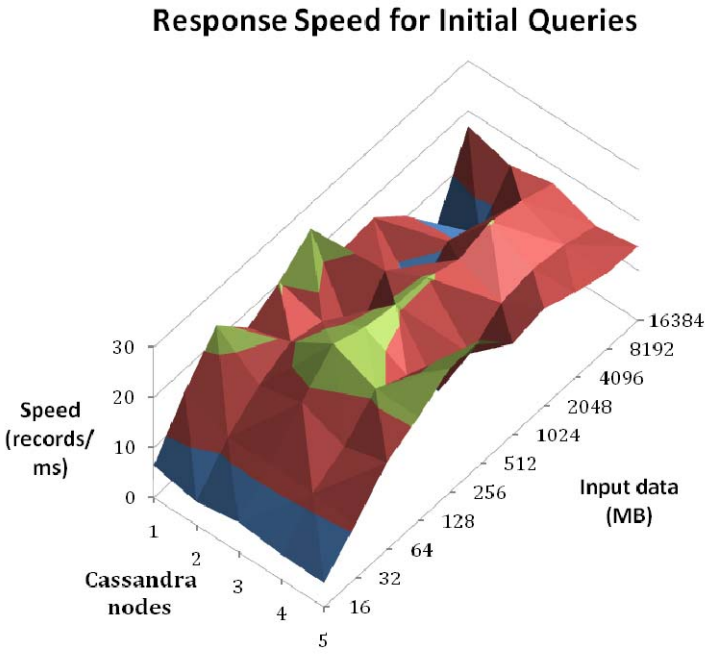

Figure 6: Fluctuating response speed as a function of input data size and number of nodes

4.3 Analysis

Both the positive and negative aspects of these results are roughly as we expected.

With Cassandra, reads are usually slower than writes. Moreover, we kept the replication factor at 1 due to the limited RAM and disk storage on the data, so we couldn't take

advantage of the ability to read concurrently from multiple replicas. This explains why we saw only modest performance improvements when adding nodes.

In addition, we only began to explore the various types of caching that are supported by Cassandra, including key caching and row caching — as well as simply delegating the caching problem to the operating system. We only use the default key caching scheme, which caches taxi IDs in memory. However, because we have only 15,000 different taxis in our system whereas Cassandra can cache up to 200,000 keys by default in memory, we don't benefit much from this feature.

Although we did not achieve the level of performance needed for interactive queries over weeks or months of taxi data, we believe that there are several ways the performance can be improved further. In particular, the combination of more RAM and more effective caching strategies could yield dramatic benefits. For interactive applications, we expect that Cassandra's support for multi-threaded queries could also help deliver speed and scalability.

5. Discussion and conclusions

5.1 Lessons learned

While the issues we encountered with the Open Cirrus testbed yielded disappointing benchmark results and prevented us from demonstrating the level of scalability we had hoped, they provided valuable lessons about the impact of cloud infrastructure design on the performance of data-intensive applications.

These lessons must be applied carefully, as the characteristics of distributed computing frameworks vary widely (as do the demands of particular applications). For example, Cassandra tends to be more sensitive to network performance than Hadoop — even with physically local storage — since Cassandra replicas do not have the ability to execute computing tasks locally as with Hadoop, meaning that tasks requiring a large amount of data may need to transfer this data over the network in order to operate on it. We believe that a commercially successful cloud computing service must be robust and flexible enough to deliver high performance under a variety of provisioning scenarios and application loads.

5.2 Potential economic impact

In the taxi domain, we see substantial economic benefits to developing analytical capabilities that are enabled by cloud computing technologies and services. There are many ways to assess the value of these benefits, but suppose for concreteness that 1% of the hours an average Singapore taxi driver spends cruising for passengers could be converted either to idle time or time with paying passengers on board.

- According to the taxi companies' data, drivers collectively spend approximately 1.8 million hours per month cruising for passengers. Saving 1% of that would yield 18,000 hours per month in which taxis are offline instead of burning fuel while driving. At an average cruising speed of 40 km/hr, a fuel efficiency of 12

km/litre, and a price of $1.20/litre, the companies could generate aggregate savings of up to $864,000 per year on fuel.

- The savings indicated above are effectively a lower bound, since we would expect drivers to convert some fraction of this idle time into time with paying passengers on board. To obtain an upper bound, let us assume that 100% of the efficiency gains from reduced cruising time are converted into fare revenue. Suppose an average revenue of $200 per day per taxi. A 1% revenue gain yields an extra $2 per day. Over a fleet of 15,000 taxis, this translates into about $10 million per year.

With these (admittedly rough) calculations as bounds, the net economic impact of a 1% efficiency gain could be on the order of several million dollars per year.

5.3 Future work

Despite our benchmark results, we remain optimistic about the use of cloud computing technologies for taxi data analytics. We are similarly optimistic that if appropriately reconfigured, the Open Cirrus testbed can yield the level of performance and scalability needed for data-intensive applications. We hope that others may benefit from our experiences in any case.

References

Amasha, Shahfik (2009). "Distributed Data Analysis Using Map-Reduce," term paper, IS 470: Guided Research in Information Systems, SMU School of Information Systems.

Lakshman, Avinash and Prashant Malik (2009). "Cassandra – A Decentralized Structured Storage System," paper presented at the 3rd ACM SIGOPS International Workshop on Large Scale Distributed Systems and Middleware (LADIS 2009), available at http://www.cs.cornell.edu/projects/ladis2009/papers/lakshman-ladis2009.pdf.

White, Tom. (2009). *Hadoop: The Definitive Guide*. O'Reilly Media.

Acknowledgements

This project was generously supported by the SMU School of Information Systems Research Centre, the SMU Centre for Academic Computing (now part of Integrated Information Technology Services), and the National Grid Office of the Infocomm Development Authority of Singapore. We especially thank Tan Wee Chuan and Ng Beng Wuan (SMU), Napat Chalakornkosol (IDA), and Wong Yeow Mun (Alatum / SingTel) for their assistance and encouragement.

We also wish to acknowledge the generous support of the collaborating taxi companies in sharing their data with the SMU Taxi Data Analytics project, and our colleagues at SMU SIS (especially Rajesh Balan) for their joint efforts on related work.