Scaling to Build the Consolidated Audit Trail: A Financial Services Application of Google Cloud Bigtable

Neil Palmer, Michael Sherman, Yingxia Wang, Sebastian Just
(neil.palmer;michael.sherman;yingxia.wang;sebastian.just)@fisglobal.com

FIS

Abstract

Google Cloud Bigtable is a fully managed, high-performance, extremely scalable NoSQL database service offered through the industry-standard, open-source Apache HBase API, powered by Bigtable. The Consolidated Audit Trail (CAT) is a massive, government-mandated database that will track every equities and options market event in the US financial industry over a six-year period. We consider Google Cloud Bigtable in the context of CAT, and perform a series of experiments measuring the speed and scalability of Cloud Bigtable. We find that Google Cloud Bigtable scales well and will be able to meet the requirements of CAT. Additionally, we discuss how Google Cloud Bigtable fits into the larger full technology stack of the CAT platform.

1 Background

1.1 Introduction

Google Cloud Bigtable[1] is the latest formerly-internal Google technology to be opened to the public through Google Cloud Platform [1] [2]. We undertook an analysis of the capabilities of Google Cloud Bigtable, specifically to see if Bigtable would fit into a proof-of-concept system for the Consolidated Audit Trail (CAT) [3]. In this whitepaper, we provide some context, explain the experiments we ran on Bigtable, and share the results of our experiments.

1.2 The Consolidated Audit Trail (CAT)

FIS (that completed its purchase of SunGard on November 30, 2015) is a bidder to build CAT [5], an SEC-mandated data repository that will hold every event that happens in the US equities and options markets. This means that every order, every trade, every route of an order between different organizations, every cancellation, every change to an order — essentially everything that happens after someone enters a trade on a computer screen or communicates an order to a broker on a phone to the fulfillment or termination of that order [6].

There are many requirements that the SEC and the stock exchanges have identified for CAT [3]. Most crucially, CAT must be able to ingest, process, and store as many as 100 billion market events every trading day (about 30 petabytes of data over the next six years [7]), and this data must be available to regulators for analysis, monitoring and surveillance [3].

But the step in between is the most challenging aspect of building CAT: taking a huge amount of raw data from broker-dealers and stock exchanges and turning it into something useful for market regulators.

The core part of converting this mass of raw market events into something that regulators can leverage is presenting it in a logical, consistent, and analysis-friendly way. Furthermore, a key aspect of making stock market activity analysis-friendly is the notion of linking individual market events together into order lifecycles. An order lifecycle is the flow of a stock market order starting from when a customer places an order with a broker-dealer to whenever that order is fulfilled (or, in some cases, cancelled or changed) [8]. Figure 1 shows an example order lifecycle.

Order lifecycles can involve thousands of market events, and there are millions of order lifecycles per trading day, all which have to be assembled by the CAT system. In a graph theory context, order lifecycle assembly can be thought of as finding the edges between billions of vertices across millions of different graphs, where only one configuration is correct. Throughout the remainder of this whitepaper, we refer to the building of order lifecycle graphs as "linkage."

1 Throughout this whitepaper we use the terms "Google Cloud Bigtable", "Cloud Bigtable", and "Bigtable" to refer to Google Cloud Bigtable. If we are referring to Google's internal Bigtable or the Bigtable technology as described in the original Bigtable paper [4], it is mentioned specifically.

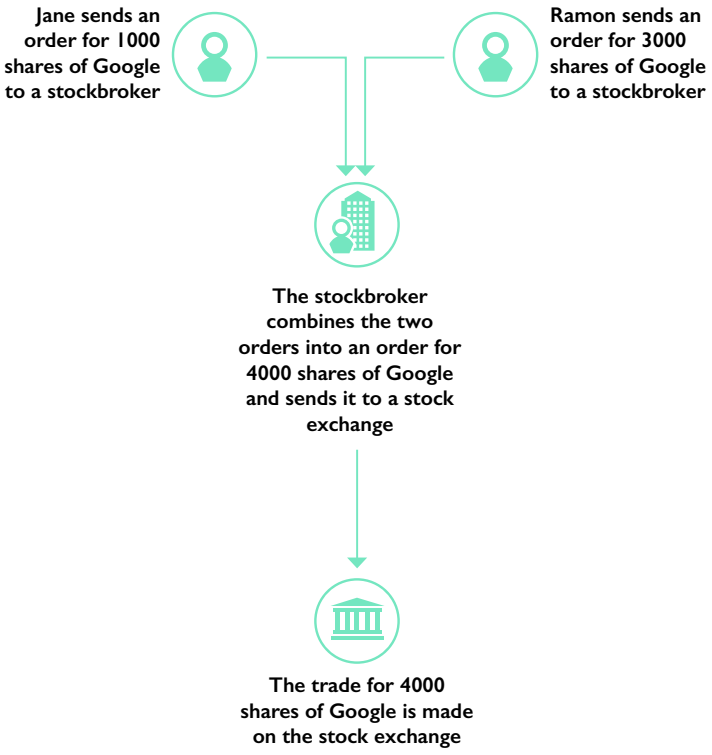

Figure 1: An example of an order lifecycle. Two stock market orders from Jane and Ramon are combined by a stockbroker into a new order, which the stockbroker sends to a stock exchange. The stock exchange then completes the order. For CAT, the stockbroker would submit three events (the orders from Jane and Ramon, and the order the stockbroker sends to the stock market) and the stock exchange would submit two events (one event for the order received from the stockbroker, and another event for filling the order).

1.3 Google Cloud Bigtable as a Solution to the CAT Problem

Google Cloud Bigtable is a fully managed, high-performance, extremely scalable NoSQL database service offered through the industry-standard, open-source Apache HBase API [1]. Under the hood, this new service is powered by Bigtable [1]. Cloud Bigtable offers features that meet the key technological requirements for CAT. These requirements and features are detailed in Table 1.

Table 1: A list of CAT requirements with the applicable Google Cloud Bigtable feature fulfilling the requirement.

CAT Requirement	Google Cloud Bigtable Feature [4]
Validate, process, store, and create linkages for 100 billion market events in 4 hours	Highly scalable, atomicity of rows, and distribution of tablets allow many simultaneous table operations
Identify errors on reported events within 4 hours	
Support heterogeneous types of data (e.g. options vs. regular stocks; completed orders vs. cancelled orders)	Flexible schema, support for billions of columns, column data only written to rows that have values for that column for optimal performance
Handle new types of linkages and new kinds of orders not currently supported by the industry	
Identify linkage errors, but be resilient enough to continue with the linkage when some events in the order lifecycle contain certain errors	Column families allow for error data to be written to a separate logical location stored within the same table.
Maintain a history and status of changes to an event and order lifecycle as error corrections are submitted	Built-in timestamping of all insertions means change history is always kept and available for audit
Begin processing events and creating linkages before all data is submitted	Auto-balancing and compactions ensure optimal speeds as more data arrives, but reasonable costs while data volumes are low
Scale up quickly when demand is above expectations	New tablet servers can be added to a Bigtable cluster in minutes, making Bigtable highly scalable
Tolerate volume increases up to 25% annually without losing speed	
Run queries during lifecycle linkage process	Nothing prevents querying while data is being written, and distribution of data across tablets mean different computers can serve multiple requests simultaneously
Backups during linkage process for quick recovery	Backups are co-written with production data, with a quick, invisible switchover in case of failure
Affordable, with no need to stockpile computing resources used only on "worst-case" days	Cloud computing model means no charges for unused computing resources

2 Evaluating the Scalability of Data Insertion into Google Cloud Bigtable

To determine if Cloud Bigtable can handle the volumes of data CAT will process and to obtain a benchmark of performance, we undertook a series of experiments. All the experiments involved the validation, parsing, and insertion of large volumes of Financial Information eXchange (FIX) messages[2] [9] into Bigtable. Our technology stack and architecture are described in Figure 2.

2 FIX is an electronic communications protocol widely used in financial services. Specifically, for the tests in this whitepaper FIX messages were text strings of field-data pairs. Many organizations in the financial services industry hope CAT will support FIX messages [12], minimizing compliance overhead.

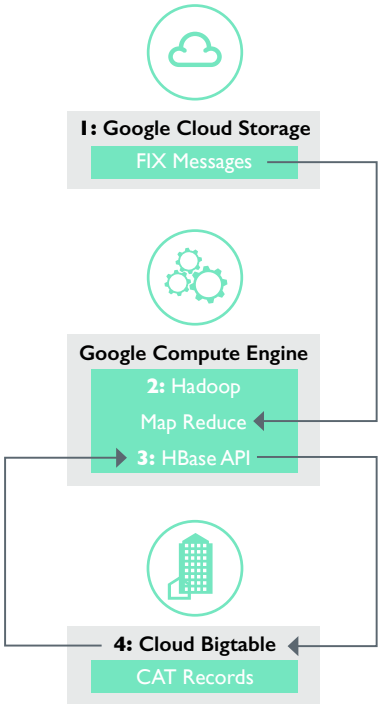

Figure 2: The technology stack for our Bigtable experiments. **1:** FIX messages were stored in Google Cloud Storage [10] buckets. Google Cloud Storage runs on Google File System (GFS)[3] [11], which is similar to HDFS[4]. **2:** Using Google Cloud's bdutil [14] tool, Hadoop clusters were created using virtual machines running on Google Compute Engine [15]. No Hadoop settings were changed from the default settings provided by bdutil. The Hadoop clusters used Google Cloud Storage buckets as a shared file system instead of HDFS [14]. **3:** MapReduce jobs were run on the Hadoop clusters. The MapReduce jobs communicated with Bigtable through the open-source HBase API. No default MapReduce settings were changed from the default settings provided by bdutil **4:** Processed data was stored on Bigtable, in a custom format we developed specifically to CAT's requirements. Note that Bigtable resides in a separate, Bigtable-only cluster from the cluster running the MapReduce jobs. This is a characteristic of the Cloud Bigtable service.

2.1 Experiment 1: Performance of Different Virtual Machine Configurations

This first experiment compared the performance of different virtual machine (VM) configurations available through Google Compute Engine (GCE).

Four different groups of VM configurations are available on GCE (standard, highcpu, shared core, and highmem). Initial tests showed that the highcpu and shared core VM groups did not have enough memory to handle our MapReduce job, and the highmem VM group performed only a bit better than the standard VM group despite costing more money to run. Therefore, we chose to formally test only the standard VM group. There are five different VM configurations available in GCE's standard VM group, outlined in Table 2 [16].

Table 2: Different virtual machine configurations (virtual CPUs and memory) available in Google Compute Engine's standard virtual machine group [16].

Virtual Machine Configuration	Virtual CPUs	Memory (GB)
n1-standard-1	1	3.75
n1-standard-2	2	7.50
n1-standard-4	4	15
n1-standard-8	8	30
n1-standard-16	16	60

The costs per core are the same across the standard VM group, therefore we wanted the VM that gave us the best performance per core. Our experimental procedure:

1. Four different clusters were created using bdutil, each having a total of 80 cores and a variable number of VMs. Each cluster was composed entirely of one configuration of standard VM, excepting n1-standard-1 which was not powerful enough to run our MapReduce job.
2. Each cluster ran the same MapReduce job: validate 100 million FIX messages, parse the FIX messages into a CAT-compliant schema, then insert the parsed messages into Cloud Bigtable.
3. The time to complete the MapReduce job was recorded for each cluster, and this time was transformed into a throughput measurement of FIX messages validated, parsed, and inserted per second per core.

Results are in Figure 3.

n1-standard-8 gave the best results. n1-standard-4 also performed well but we opted to do further experiments with n1-standard-8 because smaller clusters are easier to manage.

The difference in performance is likely due to how many JVMs are created, which is based on the CPU and RAM configuration of the worker node. We observed CPU underutilization for n1-standard-2 (2 core) and n1-standard-4 (4 core) VM configurations, and CPU overutilization for the n1-standard 16 (16 core) VM configuration. By tuning the Hadoop and MapReduce settings and other system parameters, we could potentially get better relative performance from the 2, 4 and 16-core VM configurations, but to keep consistent with the idea of using managed services we did not tune anything.

Lastly, relative VM performance can change as cluster sizes change, but this test represented a starting baseline for the remaining experiments. Research into the behavior of different VM configurations with different cluster sizes and on the more advanced linkage engine is ongoing.

3 File storage at Google also involves a technology called Colossus [13] but limited public information about Colossus is available.
4 HDFS is the Hadoop Distributed File System, and was inspired by the original GFS whitepaper [17].

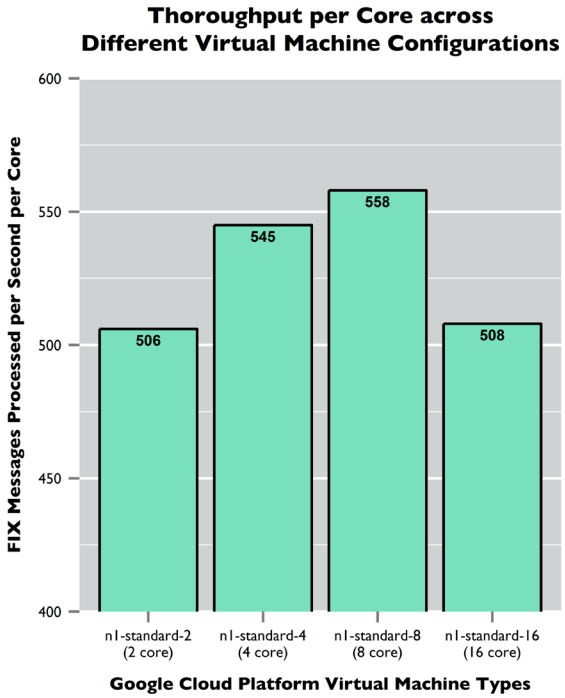

Figure 3: This figure shows the number of FIX messages validated, parsed, and inserted into Cloud Bigtable per second per core across four different Hadoop clusters, each composed entirely of workers of a single VM configuration. Each cluster consisted of the same number of virtual CPU cores (80), but a different number of worker nodes. The results were standardized to allow for a per-core comparison, since the cost per core is consistent across all VM configurations in the figure. n1-standard-4 and n1-standard-8 performed similarly, but since fewer VMs are easier to manage we opted to rely on n1-standard-8 as the VM configuration for all other experiments.

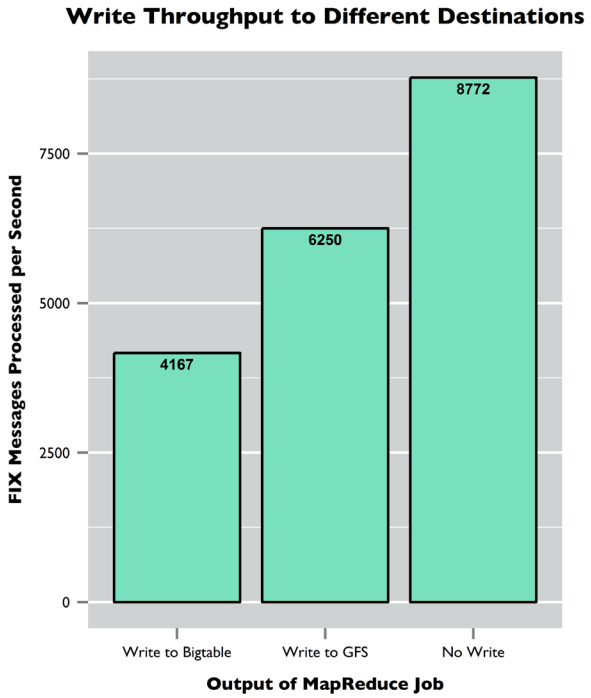

Figure 4: This figure shows the number of FIX messages validated, parsed, and then written (or not) per second across three MapReduce jobs with different output destinations (but otherwise identical). All jobs were run on a clusters consisting only of a single n1-standard-8 (8 core) worker. "Write to Bigtable" inserted all parsed FIX messages into Bigtable, "Write to GFS" wrote all parsed FIX messages to a textfile in a Google Cloud Storage bucket, and "No Write" parsed FIX messages but then did not write anything.

2.2 Experiment 2: Relative Performance of Google Cloud Bigtable Insertion

We then measured the speed of Bigtable insertions relative to baselines. Three MapReduce jobs were run, each on a single Google Compute Engine n1-standard-8 (8 core) VM instance. The input to all three jobs was 1 million FIX messages, but the outputs were different:

- MapReduce job "GFS" processed and parsed each FIX message then appended the output to a text file in a Google Cloud Storage bucket.
- MapReduce job "Bigtable" processed and parsed each FIX message, then inserted the parsed output as a row into Bigtable.
- MapReduce job "No Write" processed and parsed each FIX message then did nothing — no write to anywhere.

The relative performance of the three MapReduce jobs can be seen in Figure 4. The "Bigtable" job performs about half as fast as the "No Write" job (47.5% as fast), and about two-thirds as fast as the "GFS" job (66.7%).

2.3 Experiment 3: Performance of Google Cloud Bigtable with Increasing Job Size

The next experiment evaluated Bigtable's performance with increasing MapReduce job size. This experiment primarily functioned as a basic check of Bigtable insertion, as embarrassingly parallel jobs should not require more or less time per input unit as the number of input units increases[5] [18].

Clusters of 10 n1-standard-8 (8 core) VMs were used. Each cluster validated, parsed, and inserted a different number of FIX messages into Bigtable. The amount of time to finish each job was tracked. Figure 5a and 5b show the results of this experiment.

Unsurprisingly, as seen in Figure 5b, the cluster throughput was roughly constant regardless of the amount of data processed [18].

2.4 Experiment 4: Performance of Google Cloud Bigtable with Increasing Cluster Size

As a final evaluation of small-scale[6] performance, we analyzed the performance of Bigtable as the number of worker nodes in a cluster increased.

5 It is worth noting that MapReduce jobs have a fixed amount of overhead that does not change based on job size, and we do not explicitly take this fixed overhead into account. Our MapReduce jobs were all long enough (a half hour) that the impact of fixed overhead is negligible, and less than the time variation we would often see between different runs of the same job. Variation in times between identical jobs is expected with large parallel systems.

6 Small being a relative term. Many large broker-dealers could process a day's worth of FIX messages in a reasonable amount of time on the cluster sizes we tested in this section.

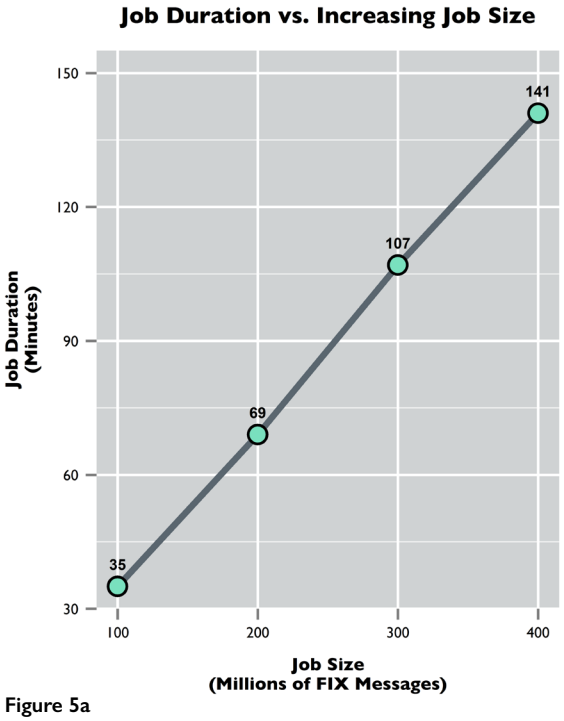

Figure 5a

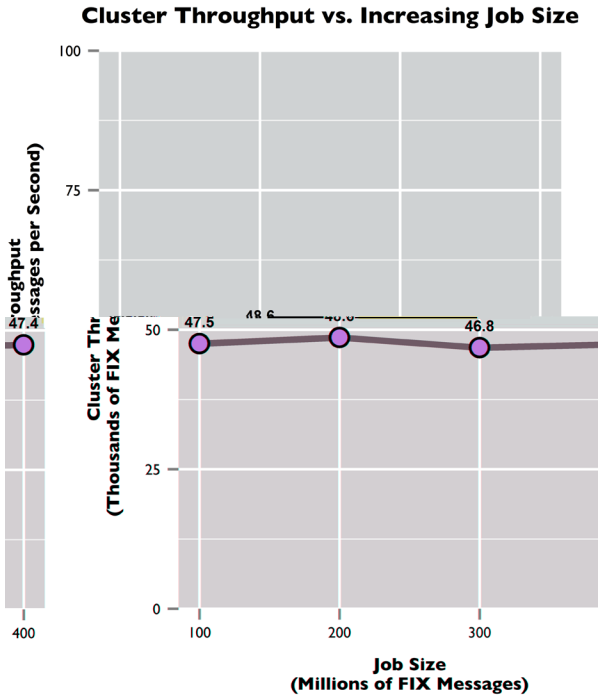

Figure 5b

Figures 5a and 5b: These figures show the effect on job duration and cluster throughput as the size of the MapReduce job is increased. All data is from clusters of 10 n1-standard-8 (8 core) virtual machines. Figure 5a shows the amount of time required to validate, parse, and insert into Bigtable an increasing number of FIX messages. Figure 5b is a transformation of the same data in Figure 5a, created by dividing the total number of FIX messages in a job by the number of seconds the job took to complete. It shows the whole-cluster throughput of the different jobs. The near-linearity of the lines connecting the data points in both figures show linear scaling with linearly-increasing job size.

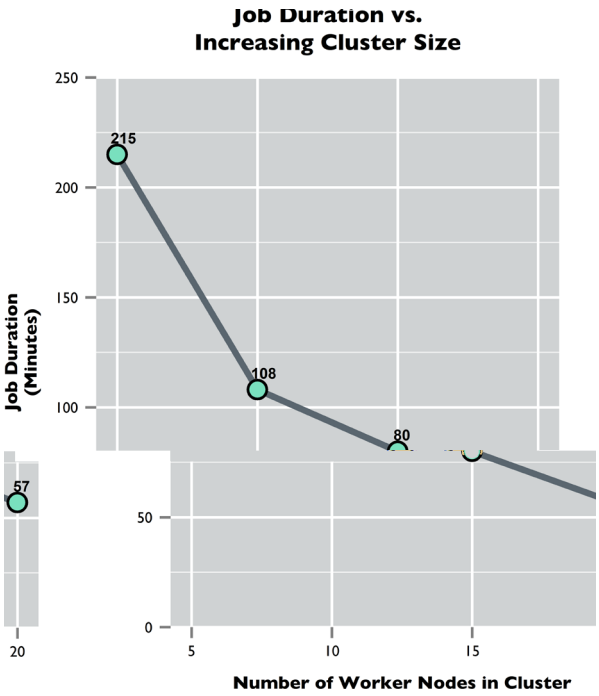

Figure 6a

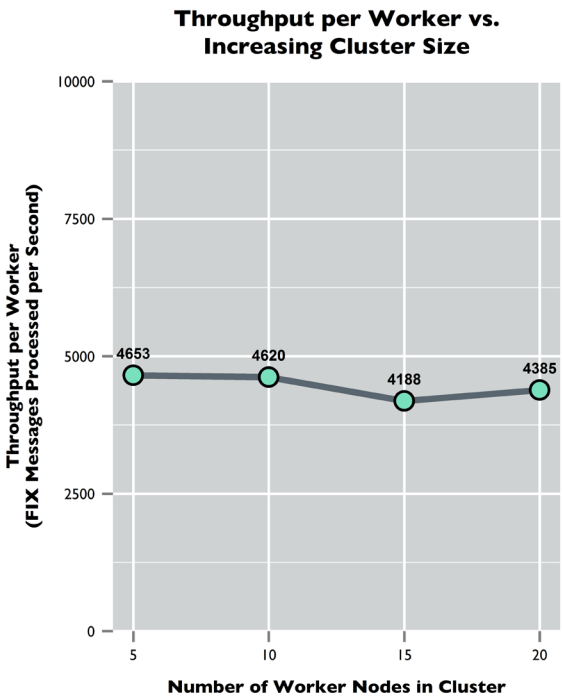

Figure 6b

Figures 6a and b: These figures show the effect of increasing cluster size on MapReduce job duration and worker throughput. All data is from clusters of n1-standard-8 (8 core) VMs. Figure 6a shows the amount of time required to validate, parse, and insert into Bigtable 300 million FIX messages on increasingly large clusters. Figure 6b is a transformation of the same data in Figure 6a, created by dividing 300 million by the job duration in minutes and the number of worker nodes in the cluster. It shows the average throughput of each workers in each cluster. The near-linearity of the lines connecting the data points in Figure 6b shows near-linear scaling with linearly-increasing cluster size.

Hadoop clusters of n1-standard-8 (8 core) VMs were created in sizes from 5 to 20 worker nodes. Each cluster ran the same MapReduce job: the validation, parsing, and insertion into Bigtable of 300 million FIX messages. The amount of time to finish each job was tracked. Figures 6a and 6b show the results of this experiment.

2.5 Experiment 5: Performance of Google Cloud Bigtable with Extremely Large Clusters

After our initial observations, it qualitatively appeared we did not use large enough clusters to get a full understanding of Bigtable insertion scalability. We were also far below the scale required for CAT. Thus, we undertook another experiment, similar in design to experiment 4.

First, per the instructions of the Cloud Bigtable team, we "primed" Cloud Bigtable by creating a small table with a similar structure to the data we were about to insert. This "priming" allows Cloud Bigtable to balance the table out amongst many tablets prior to large-scale insertion, which increases performance. Then, three Hadoop clusters (in sizes of 100, 200, and 300 worker nodes) of n1-standard-8 (8 core) VMs were created, data was inserted from each cluster, and the throughput of the clusters measured[7]. This data was combined with the data from experiment 4 and the data from the single-worker "Write to Bigtable" insertion test in experiment 2, and then the universal scalability model was applied to this full dataset [19]. The results are shown in Figure 7, with the predicted scalability model and a 95% confidence band overlaid.

The universal scalability model interprets the scaling as linear, and predicts infinite scalability. While linear scalability is impossible, at the scale we measured directly Cloud Bigtable insertion still effectively scales linearly.

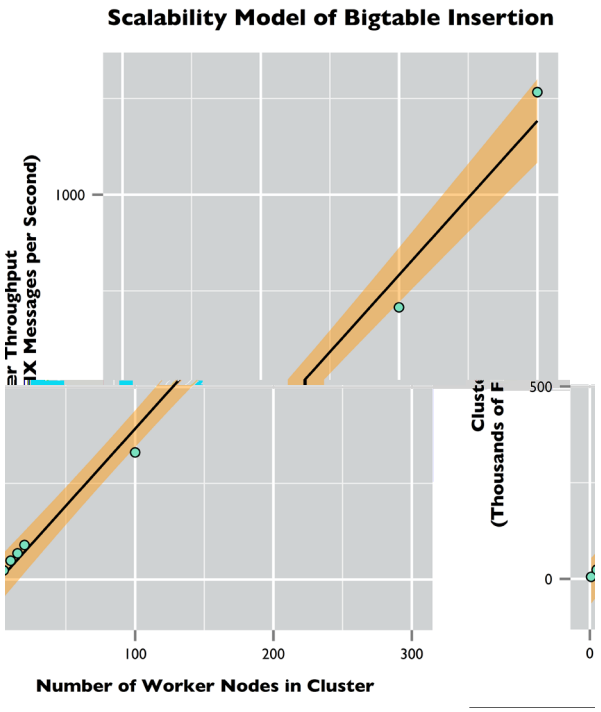

Figure 7

Figure 7: This figure shows the effect of increasing cluster size on cluster throughput and the induced universal scalability model. All data is from clusters of n1-standard-8 (8 core) VMs. Data in this figure is from MapReduce jobs of both varying cluster size and varying job size, which may have caused inconsistencies but was necessary given the large spread of cluster sizes covered. The data points are the throughput measurements, the line is the predicted scalability as obtained from the universal scalability model, and the colored band is the 95% confidence band on both sides of the prediction. The model is linear, and infinite scalability is predicted.

3 Discussion

3.1 Key Findings

- Bigtable demonstrated the ability to handle data at CAT scale, with peak ingestion rates of 10 billion FIX messages per hour and sustained ingestion of 6.25 billion FIX messages in one hour.
- With clusters up to 300 worker nodes, Bigtable insertion scaling does not begin to deviate from linear.
- Tuning of virtual machines, Hadoop clusters, or MapReduce jobs is not necessary to get good performance from Cloud Bigtable.
- Given no tuning or optimization, n1-standard-8 (8 core) virtual machines provide the best performance-per-dollar.
- Bigtable insertion (via MapReduce) job duration scales linearly with increasing job size.

3.2 Performance Metrics

Something not explicitly stated with our experimental results above is exactly how well Bigtable performed both in relation to CAT requirements as well as more generally. We have collected some key performance metrics in Table 3.

Table 3: Various measurements of Cloud Bigtable performance.

Metric	Approximate Performance
Maximum peak throughput, general	2.7 Gigabytes written per second
Data writable with 60 minutes of peak throughput, general	10 Terabytes
Maximum peak throughput, task-specific	2.7 Million FIX messages processed and inserted per second
Data writable with one hour of peak throughput, task-specific	10 Billion FIX messages processed and inserted
Maximum sustained throughput for one hour, general	1.7 Gigabytes written per second
Data written in an hour of sustained throughput, general	6 Terabytes
Maximum sustained throughput for one hour, task-specific	1.7 Million FIX messages processed and inserted per second
Data written in an hour of sustained throughput, task-specific	6.25 Billion FIX messages processed and inserted

7 We expect the instability of the large cluster results is due to the rebalancing of the tablets as our insertion throughput increased. See the "Scalability of Google Cloud Bigtable" subsection in the Discussion for more on tablet rebalancing.

It is important to note that these metrics were constrained by the number of worker nodes and the reserved bandwidth we provisioned within Google Cloud Platform, and not by Bigtable itself. Future testing will involve many more workers, and will reserve greater internal bandwidth between Google Compute Engine, Google Cloud Storage, and Google Bigtable.

3.3 Scalability of Google Cloud Bigtable

Bigtable scaled extremely well — at the volumes we tested, we did not experience any measurable contention or coherency issues. We believe that Bigtable can easily handle the scale of CAT: linear scaling was seen up to a level of performance that can process and insert 1/16 of a "worst-case" trading day in one hour.

However, we do want to clarify an important point: as Cloud Bigtable is a managed service, we could not measure the scalability of Bigtable directly. Additionally, our data was automatically moved between tablets to increase access speeds, a process described in the Bigtable paper [4]. This likely had some impact on our results, but we could only monitor the resources communicating with Bigtable — meaning we could only calculate the scalability of our Hadoop worker cluster, but not the Bigtable cluster directly. One way we mitigated this impact was by "priming" Bigtable with data similar to the data it was about to ingest, which would cause some of the rebalancing work to happen prior to the start of the timed jobs. This"priming" would not be an issue in production, it is merely necessary for proper performance tests on newly created tables.

3.4 Optimal VM Configurations and Tunings

Our experiment to determine the optimal VM configuration was very basic, and more testing is being done to optimize the worker nodes. For the purposes of testing Google Cloud Platform and the Cloud Bigtable service we decided to keep as many default settings as possible, in the spirit of managed services. However, defaults are not always optimal, especially with systems as complex as Hadoop, MapReduce, and Bigtable.

For example, we believe the reason the n1-standard-16 (16 core) VM drops in per-core throughput from the n1-standard-8 (8 core) is due to a sub-optimal balance between the number of MapReduce tasks spawned on the node vs. the amount of free memory on the node. These settings can be adjusted manually.

Additionally, the pattern of performance seen with single-worker clusters does not hold exactly at different cluster sizes, and further formal experiments are ongoing.

3.5 MapReduce vs. Google Dataflow

For these experiments, we used MapReduce with the HBase API to communicate with our Bigtable cluster. Our MapReduce job was relatively naive, and was never subjected to refactoring or tuning. We are certain the MapReduce job can be streamlined for a performance improvement, but we do not feel that optimization is a priority because we plan to switch from MapReduce to Google Dataflow.

Google Dataflow (which is derived from Google's internal "FlumeJava") is a much more manageable framework than MapReduce with comparable or superior performance [20]. It is the "big data" ETL framework of choice internally at Google [21], and we expect performance gains when we complete this conversion [20]. Additionally, a Dataflow codebase will be much easier to manage than a MapReduce codebase, due to the far greater simplicity and flexibility of the framework.

3.6 Bigtable Schema Design

Bigtable is a "NoSQL" database, and represents a very different way of thinking about data and databases from traditional relational models and even other popular NoSQL databases like MongoDB and Redis. We made only minimal Bigtable-specific optimizations for our tests, and because of this we missed out on some advantages of Bigtable.

Of particular interest to us are the notions of column families and column sparsity in Bigtable. Column families speed up data access while still allowing operations by row [4]. In addition, Bigtable's unique handling of columns allows for billions of columns without performance degradation [4]. Both of these features have factored into our designs for CAT, and have played a key role in our order linkage engine.

3.7 Querying Google Cloud Bigtable

Post-linkage, when the power of Bigtable is not as crucial, CAT data can quickly and easily be exported to BigQuery or an HDFS-based querying tool where common SQL commands can be used to query it (e.g. Hive)[8]. We are currently using Google's BigQuery as an analytical store because of its high-speed data access (terabytes in seconds) via common SQL-like commands and its ability to be queried by existing sophisticated business intelligence (BI) tools such as Tableau and Qlikview. Experiments with Google's BigQuery Connector for Hadoop [22] have been very successful, with speeds comparable to the insertion into Bigtable, thus allowing for balanced performance. However, as HBase [23] and associated tooling such as Apache Phoenix [24] mature, this may negate the need to use a separate analytical store with access becoming more user friendly than just via the HBase API.

3.8 Limitations of Our Experiments

There are a number of limitations we have identified, both in the general design of the experiments and the application of the experiments and results to evaluate the suitability of Bigtable for CAT. These limitations, as well as some comments and future work to address the limitations, are listed in Table 4.

3.9 Limitations of Google Cloud Bigtable

During our experiments we came across a few limitations of the current version of Cloud Bigtable that might require accommodations in selected use cases. These are listed in Table 5.

8 Note that many HDFS-compatible data processing and querying tools will work on data stored in Google Cloud Storage.

3.10 Next Steps

We expect to achieve in excess of 25 billion FIX messages processed per hour, in order to ensure Bigtable can handle the "worst-case" CAT scenario. Additionally, we have an event linkage engine in development that handles reconstruction of full order lifecycles. We are working on scaling the linkage engine through both volume and increased complexity of the order scenarios. Finally, we are developing specific analytical stores and tools to allow regulators to quickly learn what they need from the collected CAT data. These analytical stores and tools will be fed from Bigtable, but will not require technical skills beyond SQL commands and API calls and will allow for rapid access to all six years of market event data stored by CAT.

4 Conclusions

We have described CAT, the challenge of order lifecycle linkage, and shared results from a series of experiments assessing Cloud Bigtable's abilities to fulfill the needs of CAT. Although experimentation is ongoing, thus far Cloud Bigtable has met or exceeded all our expectations. We have achieved throughput as high as 2.7 GB per second, which let us process about 2.7 million FIX messages per second in a partially-linked, CAT-compliant way. Based on our initial results, our next goal is to exceed a rate of 25 billion validated and linked CAT events per hour, which would fully satisfy initial CAT requirements.

We were able to conduct these experiments with both a relatively small team and budget, which gives us another data point in understanding how usage of Google Cloud Platform in general, and Bigtable in particular, will help keep the costs of building and running CAT as low as possible.

While Cloud Bigtable is a recently developed public version of Google's internal system and very unlike other databases (with the exception of HBase), we found it was an easy transition and have come to realize that Bigtable's unique way of storing data opens more doors than it closes. We are conducting further research at this time, and look forward to sharing more information with CAT stakeholders about how we are using Bigtable's characteristics to elegantly handle the whole of order linkage and other CAT requirements. We believe that Google Cloud Bigtable is clearly suitable for CAT and the order linkage problem. It scales well, can handle the complex nature of the data, and interfaces well with other tools necessary to meet all CAT requirements. We are also analyzing other financial services use cases for the architecture, and encourage curious readers to contact the authors of the paper. Any use case involving large volumes of data that need to be accessed quickly is a good use case for Bigtable, particularly with varied data that does not fit cleanly into a relational-style schema.

Table 4: Some limitations of our experiments, with comments about the limitations and descriptions of future work to address the limitations.

Limitation	Comments and Future Work
Not enough trials given the variation of time endemic to large, complex MapReduce jobs	Further tests have been run and continue to be run, and results appear similar. Most data points in this whitepaper are already the average of multiple identical runs.
Formalized testing of reads not completed	This is the next phase of our testing, and is already underway.
No access to real CAT data, cannot possibly account for all possible order scenarios	Real CAT data does not exist yet. We have used a combination of real FIX messages and simulated data. Testing of a larger variety and higher complexity of orders is underway.
No evaluation of durability	Both workers and tablet servers died during some of our tests, but did not stop MapReduce jobs and would not have been noticed except for the logging messages generated. Formal durability evaluation is planned, possibly with a Chaos Monkey-like tool [25].
Relative performance of virtual machine configurations only measured on small clusters	Testing of virtual machine configurations at higher scales has already commenced.
Failure to account for rebalancing and scaling of Bigtable cluster	While this may have affected some of our experiments, a better understanding of the Bigtable cluster could only raise our performance metrics. Making rebalancing and scaling of the Bigtable cluster more transparent and manageable would be helpful in increasing our understanding in future experiments.

Table 5: Some limitations of the current version of Cloud Bigtable, with comments about the limitations and possible accommodations to address the limitations.

Limitation	Comments and Accommodations
Cannot directly manage Bigtable cluster	During our experiments, Bigtable managed itself quite well. We have the option to manually change the cluster size, and intelligent use of Bigtable aids the Bigtable service's smart management.
Must use HBase API to access Bigtable	Easy, fast transfer of data to BigQuery with BigQuery connector for SQL-like querying.
No SQL query support for Bigtable	

Acknowledgements

We would like to thank the following people for providing their help, support, and technical and domain expertise in getting this proof of concept built: Sal Sferrazza, Sam Sinha, Benjamin Tortorelli, Meghadri Ghosh, Srikanth Narayana, Keith Gould, Peter Giesin, Chris Tancsa, Patricia Motta, Adriana Villasenor, Petra Kass, Gemma Wipfler, Steven Silberstein, and Marianne Brown at FIS, and especially the Google Cloud Bigtable team, who have been awesome.

More Information

FIS (formerly SunGard) Tests CAT Prototype Using Google Cloud Bigtable, with Quarter Hour Ingestion Rates Equivalent to 10 Billion Financial Trade Records per Hour: http://bit.ly/1JOn4do
SunGard Homepage: http://www.sungard.com
Announcing Google Cloud Bigtable: Google's world-famous database is now available for everyone: http://goo.gl/lVWXuy

References

[1] Google. *"Announcing Google Cloud Bigtable: Google's world-famous database is now available for everyone,"* googlecloudplatform.blogspot.com. [Online]. Available: http://goo.gl/lVWXuy [Accessed May 4, 2015].

[2] Google. *"Google Cloud Platform,"* cloud.google.com. [Online]. Available: https://cloud.google.com/ [Accessed: May 4, 2015].

[3] CAT NMS Plan Participants, *National Market System Plan Governing the Consolidated Audit Trail Pursuant to Rule 613 of Regulation NMS under the Securities Exchange Act of 1934*. catnmsplan.com [Online]. Available: http://catnmsplan.com/web/groups/catnms/@catnms/documents/appsupportdocs/p602500.pdf [Accessed: May 4, 2015].

[4] F. Chang, J. Dean, S. Ghemawat, W. Hsieh, D. Wallach, M. Burrows, T. Chandra, A. Fikes and R. Gruber, "Bigtable: A Distributed Storage System for Structured Data," in *Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation,* May 30-Jun. 3, 2006, Boston, MA, Berkeley, CA: USENIX Association, 2006.

[5] CAT NMS Plan Participants, *"SROs Select Short List Bids for the Consolidated Audit Trail"*, catnmsplan.com [Online]. Available: http://catnmsplan.com/web/groups/catnms/@catnms/documents/appsupportdocs/p542077.pdf [Accessed: May 4, 2015].

[6] A. Tabb and S. Bali, "The Consolidated Audit Trail (Part I): Reconstructing Humpty Dumpty," Tabb Group, Westborough, MA, Mar. 2 2015.

[7] A. Tabb and S. Bali, "The Consolidated Audit Trail (Part II): Problems and Pitfalls," Tabb Group, Westborough, MA, Mar. 10 2015.

[8] United States of America. Securities and Exchange Commission, Consolidated Audit Trail. 17 CFR § 242 (2012).

[9] *Financial Information Exchange Protocol Version 4.2*, Fix Protocol Limited, 2001.

[10] Google. *"Cloud Storage"* cloud.google.com. [Online]. Available: https://cloud.google.com/storage/ [Accessed: May 4, 2015].

[11] S. Gallagher (2012, Jan 26). "*The Great Disk Drive in the Sky: How Web giants store big—and we mean big—data"*. Ars Technica [Online]. Available: http://arstechnica.com/business/2012/01/26/the-big-disk-drive-in-the-sky-how-the-giants-of-the-web-store-big-data/ [Accessed: May 4, 2015].

[12] Financial Information Forum, *FIF Consolidated Audit Trail (CAT) Working Group Response to Proposed RFP Concepts Document*, Financial Information Forum, Jan. 18 2013. [Online]. Available: http://catnmsplan.com/web/groups/catnms/@catnms/documents/appsupportdocs/p602500.pdf [Accessed: May 4, 2015].

[13] C. Metz (2012, Jul. 10). "Google Remakes Online Empire with 'Colossus'", Wired [Online]. Available: http://www.wired.com/2012/07/google-colossus/ [Accessed: May 4, 2014].

[14] Google. *"Command-Line Deployment,"* cloud.google.com [Online]. Available: https://cloud.google.com/hadoop/setting-up-a-hadoop-cluster [Accessed: May 4, 2015].

[15] Google. *"Google Compute Engine,"* cloud.google.com. [Online]. Available: https://cloud.google.com/compute/ [Accessed May 4, 2015].

[16] Google. *"Instances,"* cloud.google.com. [Online]. Available: https://cloud.google.com/compute/docs/instances [Accessed: May 4, 2015].

[17] R. Vijayakumari, R. Kirankumar and K. Gangadhara Rao, "Comparative analysis of Google File System and Hadoop Distributed File System", *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 3, no. 1, pp. 553-558, Feb. 2014. [Online]. Available: http://www.warse.org/pdfs/2014/icetetssp106.pdf [Accessed: May 4, 2015].

[18] S. Kaisler, *Software Paradigms*. Hoboken, N.J.: Wiley-Interscience, 2005.

[19] N. Gunther, *Guerrilla Capacity Planning*. Berlin: Springer, 2007.

[20] C. Chambers, A. Raniwala, F. Perry, S. Adams, R. Henry, R. Bradshaw and N. Weizenbaum, *"FlumeJava,"* in *Proceedings of the 2010 ACM SIGPLAN conference on Programming language design and implementation, June 5-10, 2010, Toronto, ON*, New York: ACM, 2010. pp. 363-375.

[21] Y. Sverdlik (2014, Jun 25). *"Google Dumps MapReduce in Favor of New Hyper-Scale Cloud Analytics System"*. Data Center Knowledge [Online]. Available: http://www.datacenterknowledge.com/archives/2014/06/25/google-dumps-mapreduce-favor-new-hyper-scale-analytics-system/ [Accessed: May 4, 2015].

[22] Google. *"BigQuery Connector for Hadoop,"* cloud.google.com. [Online]. Available: https://cloud.google.com/hadoop/bigquery-connector [Accessed: May 4, 2015].

[23] L. George, Hbase: *The Definitive Guide*. Sebastopol, CA: O'Reilly, 2011.

[24] Apache. "*Apache Phoenix: High performance relational database layer over HBase for low latency applications"* phoenix.apache.org. [Online]. Available: http://phoenix.apache.org/ [Accessed: May 4, 2014].

[25] F. Faghri, S. Bazarbayev, M. Overholt, R. Farivar, R. Campbell and W. Sanders, "Failure Scenario as a Service (FSaaS) for Hadoop Clusters", in SDMCMM '12: *Proceedings of the Workshop on Secure and Dependable Middleware for Cloud Monitoring and Management, Dec. 4, 2012, Montreal*, QC, New York, NY: ACM, 2012.