QuantCloud: Big Data Infrastructure for Quantitative Finance on the Cloud

Article · January 2017

DOI: 10.1109/TBDATA.2017.2649544

4 authors, including:

Peng Zhang
Stony Brook University
62 PUBLICATIONS **134** CITATIONS
SEE PROFILE

Kai Yu
Dell
22 PUBLICATIONS **5** CITATIONS
SEE PROFILE

QuantCloud: Big Data Infrastructure for Quantitative Finance on the Cloud

Peng Zhang, *Member, IEEE*, Kai Yu, Jessica J. Yu and Samee U. Khan, *Senior Member, IEEE*

Abstract—In this paper, we present the QuantCloud infrastructure, designed for performing big data analytics in modern quantitative finance. Through analyzing market observations, quantitative finance (QF) utilizes mathematical models to search for subtle patterns and inefficiencies in financial markets to improve prospective profits. To discover profitable signals in anticipation of volatile trading patterns amid a global market, analytics are carried out on Exabyte-scale market metadata with a complex process in pursuit of a microsecond or even a nanosecond of data processing advantage. This objective motivates the development of innovative tools to address challenges for handling high volume, velocity, and variety investment instruments. Inspired by this need, we developed QuantCloud by employing large-scale SSD-backed datastore, various parallel processing algorithms, and portability in Cloud computing. QuantCloud bridges the gap between model computing techniques and financial data-driven research. The large volume of market data is structured in an SSD-backed datastore, and a daemon reacts to provide the Data-on-Demand services. Multiple client services process user requests in a parallel mode and query on-demand datasets from the datastore through Internet connections. We benchmark QuantCloud performance on a 40-core, 1TB-memory computer and a 5-TB SSD-backed datastore. We use NYSE TAQ data from the fourth quarter of 2014 as our market data. The results indicate data-access application latency as low as 3.6 nanoseconds per message, sustained throughput for parallel data processing as high as 74 million messages per second, and completion of 11 petabyte-level data analytics within 53 minutes. Our results demonstrate that the aggregated contributions of our infrastructure, parallel algorithms, and sophisticated implementations offer the algorithmic trading and financial engineering community new hope and numeric insights for their research and development.

Index Terms—Big Data, Cloud computing, Quantitative Finance, Parallel Processing

1 Introduction

QUANTITATIVE finance relies on data to provide vital and actionable information for all aspects of the industry [1-4]. Financial organizations gain numeric insights from ever-growing structured and unstructured data provided by a variety of sources including global markets and media [5, 6]. The financial industry encompasses diverse asset-managing businesses, including commercial and investment banks, stock brokers, hedge funds, and government-run institutions. Businesses are operated by organizations ranging from large corporations such as JPMorgan Chase to proprietary trading shops consisting of a few individuals. However, the essential goals remain to: (i) maximize profits, and (ii) minimize risks by gaining subtle insights into market opportunities. Understanding market data is vital for understanding the financial market in the big data era [3, 4, 7].

Mathematical modeling plays a key role in quantitatively understanding markets [4, 8, 9]. Over time, increasingly sophisticated models such as stochastic calculus have arisen to obtain corresponding values and relations between complex events. Greater volumes of analyzed data and increased model sophistication have led to a more accurate understanding of market patterns. As capital markets step into the big data era, processing constraints for large amounts of data has challenged model development. Petabyte or even exabyte-level data, such as NYSE (New York Stock Exchange) TAQ (Trade and Quota), need to be processed efficiently. To some extent this process requires extensive repetition under different model parameters in response to volatile global markets [10, 11].

Algorithmic trading (algo-trading) is the process of programming computers to place electronic trades according to predefined strategies [12, 13]. Algo-trading handles not only high-volume big data, but also high-velocity data processing. In today's markets, a stock can experience 500 quote changes and 150 trades in one microsecond [14]. Consequently, prices fluctuate by the milli- or even microsecond [15]. Placing a large number of orders at high speeds based on programmed strategies is crucial for profit generation. In particular, the rate of market data access and data-driven strategy processing closely determines algo-trading development [16]. In other words, more efficient big data infrastructure can create profitable opportunities.

Considering on the aforementioned issues, the data explosion is a significant challenge and opportunity for today's quantitative finance. The ever-growing volume, variety, and velocity of data from global markets and trade volatility constitute an unprecedented challenge for computing

- *P. Zhang is with the College of Engineering and Applied Sciences, Stony Brook University, NY 11794. E-mail: peng.zhang@stonybrook.edu.*
- *K. Yu is with the Global Solutions Engineerings Group, Dell Inc., Round Rock, TX 78682. E-mail: kai_yu@dell.com.*
- *J.J. Yu is with the Computer Science Department, Massachusetts Institute of Technology, MA 02139. E-mail: yujess@mit.edu.*
- *S.U. Khan is with the Electrical and Computer Engineering Department, North Dakota State University, Fargo, ND 58108. E-mail: samee.khan@ndsu.edu.*

efficiency. Once properly structured and analyzed, big data solutions can provide differentiating signals that translate into prospective profits. To achieve this end, the big data infrastructure in quantitative finance must combine scalable data storage with an ultrafast data processing system to support big data analytics [17-19].

The performance of data processing systems such as this big data analytics system is often limited by infrastructure components, such as the CPU, memory, network, and storage. While CPU, memory, and network performance has seen dramatic improvement over time, advancements in storage have lagged due to limitations presented by latency and throughput. Modern techniques, such as in-memory databases, which rely on main memory for a datastore medium, are faster than disk-optimized database systems [20], but are still limited by today's memory capability. In addition, in-memory database still lack a non-volatile storage medium to provide long-term persistent storage. To handle QF big data, SSD (solid state disks)-backed storage could be more efficient than HDD (hard disk drive)-backed storage [21, 22]. The SSD based storage system has been widely used for performance-critical business data processing applications such as commercial databases running on the Relational Database System (RDBMS). For instance, a study has shown that an SSD storage appliance based on 8 flash SSD cards can deliver 17 times the I/O throughput for OLTP relational database access and 2.7 times for OLAP relational database access at 10% the latency that 96 X 15k rpm conventional HDDs can provide [23]. By leveraging SSD storage, we can significantly reduce the I/O bottleneck, leading to improvement in overall database performance. Hence, we adopt this SSD storage as the datastore medium and explore how much this SSD storage can enhance big data analytics by comparing the performance of an SSD-backed system with that of an HDD-backed system.

Parallel processing is an elegant way to improve the data-processing speed, but the process for analyzing market data is complex. In particular, the market data need to be parsed and grouped by their symbols and sorted by time before they are ready for use by algorithmic traders. Often, the data in datastore are compressed and hashed for storage economy and data security, which in turn necessitate a time-consuming data decompression and dehashing procedure. However, these essential steps should be implemented and examined to provide a result repeatable by practitioners.

Lastly, cloud computing must be incorporated in the development of big data infrastructure [24-26]. Computing capability nearly doubles every 18 months [27-29]. Computer hardware upgrades take place almost every quarter, not to mention corresponding software updates. Thus, building local or personal computing systems is increasingly financially inefficient in the face of rapidly advancing state-of-the-art computing technologies. The emerging cloud computing paradigm allows the users to quickly deploy on-demand applications on converged infrastructure and shared services [25]. In this regard, it can provision scalable computing resources while lowering operational costs, drawing attention to financial computing service.

Contribution synopsis: The main contribution of this work is to integrate a data-on-demand database system, parallel processing, and Cloud computing in the QuantCloud toolchain. Through conducting application performance testing on commodity hardware, we characterize all essential aspects of performance such as latencies and sustained throughputs of typical applications, the speedup and parallel efficiency for infrastructure, and the quality of service under the saturation condition. In particular, the development of the QuantCloud tool consists of the following:

- A scalable database system is designed and implemented for high-volume data storage. The market data is stored in a compressed and encrypted format for space efficiency and data security purposes. At runtime, the frequently used data persist in memory, together with the indices for other disk-backed data.
- A parallel pipeline system is designed and implemented for a high-velocity data processor. Any user request is executed in a parallel mode. In particular, we overlap data transfer with the data decompression process, reducing network latency. Meanwhile, we perform data decompression, reorganization and reformation in a multi-threaded mode, exploiting the parallelism.
- Cloud computing is incorporated. We develop communication protocol upon TCP/IP and describe the user request in XML format. We show the potential for extending the infrastructure onto the cloud network.
- Most importantly, we developed and tested our tool on state-of-the-art commodity hardware including a TB-level SSD-backed datastore system and a 40-core, 1TB-memory computer. This benchmark showed the latency, sustained throughput, and quality of service for data-oriented research in quantitative finance. The results provide algorithmic traders and financial engineers with numeric insights.

The reminder of paper is organized as follows: Section 2 presents related works about Cloud infrastructures for Big Data finance services. Section 3 presents the QuantCloud infrastructure, organized into overviews followed by technical details for each key component. Section 4 presents the implementation details and hardware configuration. Performance targets are presented in Section 5. Results are summarized in Section 6 and discussed in Section 7. We draw our conclusions in Section 8.

2 Related Works

Big data is often closely connected with Cloud computing in the industry. Advantages of Cloud computing include high scalability, ubiquitous availability, and low maintenance cost. These qualities attract small finance firms that have limited funds and time for IT investment. Instead, the culture is changing [30]. For instance, Amazon Web Services (AWS) offer a platform for financial services in the Cloud[1]. On this platform, AWS works with third parties by

[1] url: https://aws.amazon.com/financial-services/

providing data source, storage, analytics and virtualization tools. However, it is difficult to build an integrated big data tool to cope with diversified components from different providers. In practice, offering an integrated big data infrastructure is more useful for the financial engineers than simply offering assorted high-performance building blocks. With an integrated infrastructure, financial engineers easily adopt well-developed solutions. Thus, with this work, we not only offer a scalable database using high-performance hardware SSD but also build a parallel pipeline system integrated with the database. Altogether, the integrated big data infrastructure supports high-speed analytics for big data.

In addition, Oracle offers Big Data Cloud usage for the financial services[2]. On this platform, Oracle Big Data Appliance presents Hadoop-oriented big data infrastructure integrated with Oracle database. Often, the MapReduce programming model is deployed, but MapReduce lacks support for these data-dependent, low-latency processes. However, in this work, we not only support data-on-demand services for fetching historical data but also experimentally demonstrate the latency is as low as 3.6 ns per tick message.

Unlike the general big data infrastructure, our financial big data infrastructure offers an integrated solution for quantitative finance. This solution not only provides a big data management approach but also incorporates efficient solutions to meet big data requirements unique to the area of quantitative finance. For instance, we optimize the database for the NYSE intraday transactions data, reorganize raw data by days and symbols, and store these reorganized data for symbols in the SSD-backed system. In the client, we develop a complete toolchain for data acquisition, decompression, dehashing and finally convert the streamlined transactions into a user-defined data structure. In practice, this is the best implementation for a financial engineer making big data tools with a big data infrastructure. However, traditional approaches did not incorporate so integrated a toolchain. Moreover, we develop our infrastructure alongside a portability solution embedded in the Cloud. To use our financial big data infrastructure, the Quants need only compose and send scripts to deploy the tasks in the Cloud – this requires very limited knowledge of programming. This approach helps to build high user retention and alleviate integration challenges for big data techniques in finance.

3 QuantCloud Infrastructure

3.1 Overview

QuantCloud includes a hybrid database system to support scalable data access and a parallel platform to support fast data processes. The *Server* refers to the database system, which manages the datastore and responds to data-on-demand queries. The *Client* refers to the parallel platform, which accepts and processes user requests in parallel and queries the metadata from *Server* based on each request's needs. The communication between Server, Client and User is entirely based on Internet connections. The message between Server and Client is coded in a compressed and hashed format for space efficiency and data security purposes. Messages between Client and User are coded in custom XML format for portability across diverse systems. Table 1 presents short terms and their explanations.

3.2 Design Components

Fig. 1 presents an overview of the QuantCloud infrastructure, in which Server and Client are two key components. Server and Client are responsible for data management and data processing for user requests, respectively. The key technical details are provided in following sections.

3.2.1 High-Volume Data Storage

The Server is built based on a high-volume SSD-backed datastore system. In the system, we store the intraday transactions data (TAQ or trades and quotes) listed on the NYSE which report the US equities trades (Trade) and best bid/offer (BBO). The TAQ data total 15~30 GB per day on average. Estimating that research and data analysis require a year of historical data, a single year's data total 4~5 TB of storage. If execution analysis for high frequency trading analysis requires level-2 data, the storage amount may easily quadruple. Options data are usually one order of magnitude greater than equities.

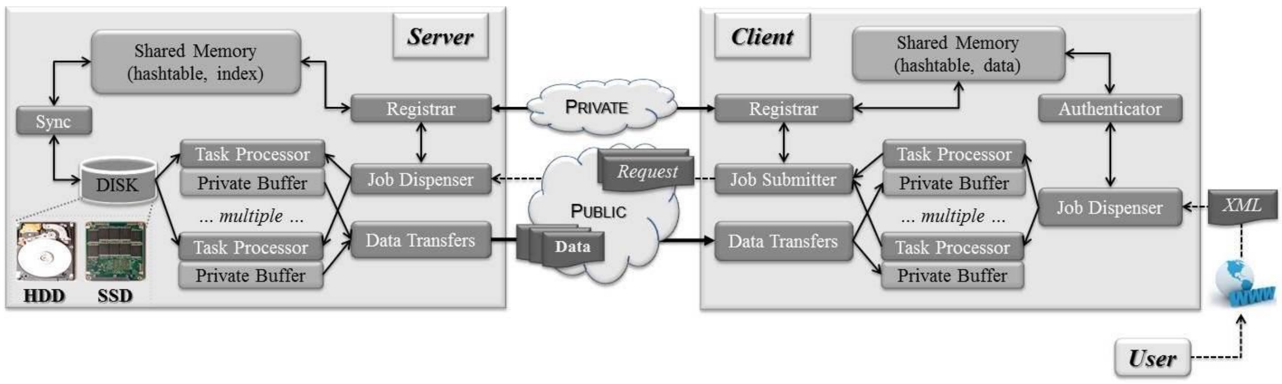

Fig. 1: Overview of QuantCloud Infrastructure

By learning user habits and routines, we reorganize the

[2] url: https://cloud.oracle.com/en_US/financial-services-cloud

metadata using days and symbols, and then index the time-series data for symbols. In particular, the market data are stored in separate data files labeled by weekdays. The intraday data are grouped by symbols and then ordered by message timestamps. One timeseries data per symbol is keyed by stock symbol for each single day. In total, there are two indexing systems: one big table for all per-day data files and one table for per-symbol timeseries data for each day.

In the datastore, we use one-way hashing to encrypt data for security purposes [31]. The hashed timeseries data are compressed by a customer-specified algorithm (such as zlib) and stored in a disk-backed file system. We store encrypted data and their hashtables, separately.

When providing services, Server daemonizes and responds to two message types: registration and data query. Registration allows Client to register at Server through authentication. After authentication, Server establishes a security channel to transfer hashtables to the Client. Data query allows a registered Client to send the data-on-demand queries to Server. Afterwards, the Server assigns a task processor to return a response for this query. Here, we use data-on-demand (DoD), a flexible and efficient format for providing a subset of large amounts of metadata. A DoD query specifies a set of stock symbols and a time period. The task processor only retrieves the data specified by this query. This query functions as a SQL SELECT statement to return data from a database.

To better utilize network bandwidth, the system consolidates multiple small results into a larger packet and splits very large messages into smaller packets of certain size. In this way, our system can effectively control the size of network packets within a certain range and help avoid network congestion. In our tests, we use 10MB as our threshold that is obtained through experimentation.

Multiple task processors in Server share disk-backed storage and handle multiple requests concurrently. This efficiently improves the throughput for market data access. Performance is presented in Section 6.1.

3.2.2 High-Velocity Data Processor

The Client is built for fast processing user requests. Upon starting, Client registers at given Server(s) and then waits for a user request. When a request arrives, Client first verifies the user identity and then queues his request. At Client, the next available task processor dequeues a request. For this request, the task processor communicates with a Server to fetch any necessary data and corresponding hashtables. As the network packets stream into the Client from a Server, the task processor starts to decompress and dehash the network packets into the market data. Finally, the system stores restored market data in a user-defined data structure for friendly use by the user function calls. The whole process is referred to as the *data decompression and process* (DDP).

This DDP process operates in a multithreading mode, in which the communication part (data transfers) and the computation part (task processor) can run in parallel. The "data transfers" module streams the metadata into the private buffer that is attached to a given "task processor". As soon as enough data arrive, the task processor starts to decompress and dehash the incoming metadata on a separate worker thread. We provide a detailed discussion on performance in Sections 6.2 and 6.3.

3.2.3 High-Variety Data Management

In our datastore, we consider two data types: Trade and BBO (Best Bid/Offer). In general, the amount of BBO data is approximately one order of magnitude greater than that of Trade data. Trade data are often used in research involving historical data. Therefore, in data management, we build the database for Trades purely in memory and the database for BBO on the disk-backed medium. However, we pre-load the index structures for BBO data and store them in shared memory. This provides easy access to frequently used data while accelerating access to disk files.

3.2.4 High-Throughput Data Analytics

The conventional technique for analyzing TAQ data strictly focuses on large amounts of data such as tick data from a month-long period. For example, single-year level-1 market data requires 4~5 TB of storage. Currently, loading all of the data at once always overwhelms the physical memory. Therefore, to perform such big-data analytics, we design a parallel-pipelined procedure. Fig. 2 shows a schematic flowchart for the parallel pipeline procedure. In this procedure, we take for example that a user requests 1-hour data. This user request for 1-hour market data is first split into a sequence of six requests each for 10-minute data and marked as: Req0~Req5. Correspondingly, we need to retrieve six returned data packets labeled Data0~Data5. Job Submitter of Client enqueues all of the requests and sends them to Job Dispenser of Server as long as there are no more than two data packets at Private Buffer of the Task Processor. Fig. 2 shows a scenario: as Task Processor of a Client processes Data1, Data2 transfers over the network to the Client and Data3 is assembled at Server. This pattern of overlapping communication and computation helps to reduce network latency. Memory for used data packets is recycled for incoming packets. This aims to reduce network latency and minimize the memory requirement for big data analytics development and supporting high-throughput targeted data processing.

3.2.5 Cloud Computing Considerations

Recent years have seen rapid growth of Cloud computing to provide adequate quality of computing services. Consider a small proprietary trading company consisting of a couple of individuals. Unlike giant finance enterprises, small companies can hardly afford investment in IT infrastructures. Instead, they resort to the 'pay-as-you-go' model for cloud services [3, 4].

A Cloud is used as a platform in which data, software, and hardware are shared. The benefits of using Cloud computing include ubiquitous availability and high scalability. A trader needs only a smartphone or a laptop to access big data and can launch data-intensive computing from any location with an Internet connection. Technically, cloud computing often uses a client-server architecture, in which a user client interacts with a cloud server. This diagram greatly improves the user experience by providing portable computing through the Cloud service.

This trend motivated us to employ a client-server model as our framework (Fig. 1) using a TCP/IP connection between modules. An XML format is employed between client and user for portability. Fig. 3 shows an example of such a request script in which a user requires data access service for the trade and BBO metadata for S&P500 symbols from Oct 1 2014 to Jan 1 2015. These symbols are stored in the file */home/user/spx500.txt*.

We assume two communication channels exist between a Server and its Client. One channel is established as the Private channel. This channel is used for client authentication, such as registration. Hashtable transfer also makes use of this channel. The other channel is established as the Public channel. Any registered Client can send its requests to Server and receive requested data from Server. Here, the returned data are hashed and compressed. Producing hashed data aids in data security and compressing hashed data allows for space efficiency.

In consideration of their functions, the Private channel requires security but not a high-speed network since it only transfers small amounts of data. On the other hand, the Public channels should be established over a high-speed network since they handle massive data transfers.

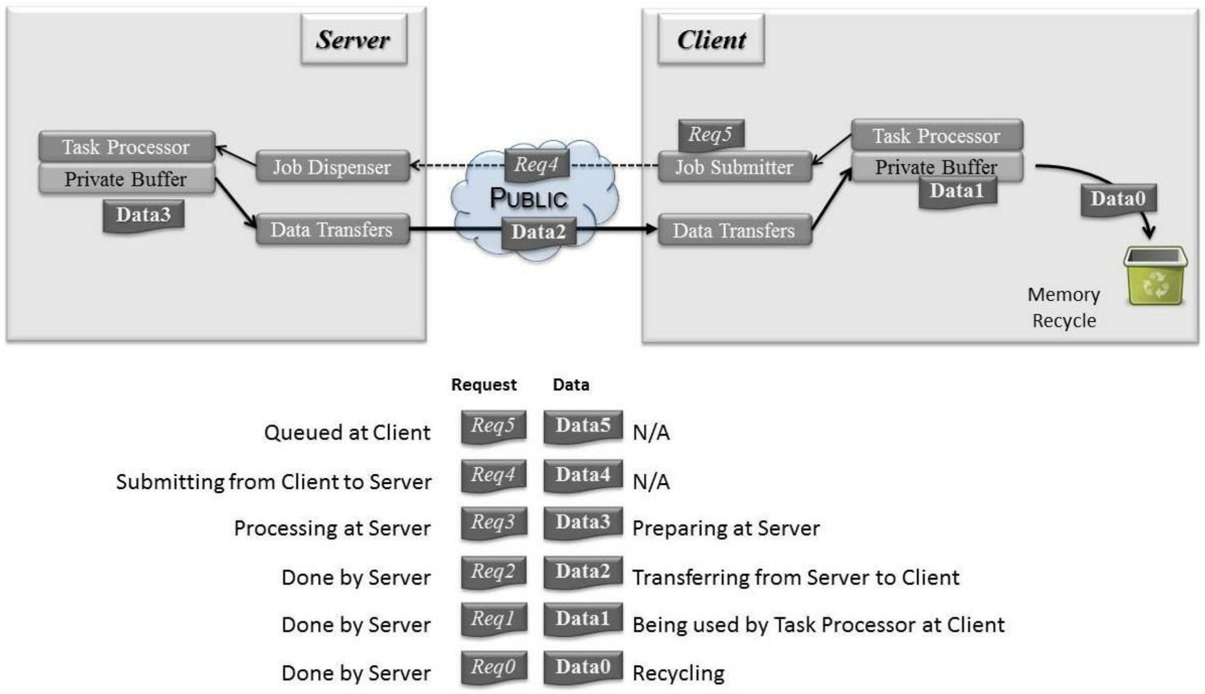

Fig. 2: Schematic flowchart for a parallel pipelined procedure

TABLE 1:
SHORT TERMS AND THEIR EXPLANATIONS

Short Terms	Explanations
BBO	Best bid/offer
DDP	data decompression and process
DoD	Data-on-Demand
FC	Fibre channel
HDD	Hard disk drive
QF	Quantitative Finance
QC	QuantCloud
QoS	Quality of Service
NYSE	New York Stock Exchange
S&P 500	Standard & Poor's 500
SSD	Solid state disk
TAQ	Trade and Quote
WCT	WallClock Time

```
<message type="3" priority="0">
  <service name="DataAccess" />
  <db_name value="TaqDaily" />
  <universe_file      name="spx500.txt"      loca-
tion="/home/user/" />
  <msg_types value="Trade,BBO" />
  <time_range  from="20141001  00:00:00.000  to="
20150101 00:00:00.000 />
</ message>
```

Fig. 3: Request script from User to Client.

4 QUANTCLOUD IMPLEMENTATIONS

4.1 Software Stack

The QuantCloud is implemented in C++. User APIs are provided in XML and help users submit their requests. Request and data transfer between Server and Client take place through TCP/IP protocol. "Shared Memory" is accessible among all of the participating threads at the Server or Client. All Task Processors at Server may share disks. The Job Dispenser employs a first-in, first-served strategy to process the incoming requests. A thread pool pattern is used to execute Task Processors at Server/Client. The main thread manages a thread pool for Task Processors. We implement thread pools using the Boost thread library. Each Task Processor uses the same thread pool pattern to run its subtasks. That is, a Task Processor initiates its own thread pool to decompress and dehash the market data packets. As for memory management, a Task Processor possesses its private heap through which it can recycle local memory without frequently using global storage resources. This characteristic effectively prevents the potential interference caused by other worker threads.

4.2 Hardware Components

The QuantCloud's hardware stack consists of the following components:

1. Two host machines: Server Host and Client Host, which host QuantCloud's server tier and client tier, respectively.
2. Two storage systems: SSD storage and HDD storage, between which we compare efficiency.
3. Two networks: 10GbE Ethernet network for communication between server and client tiers, and 16Gbps Fibre Channel (FC) network between the server tier and the two storage systems. Multiple links provide greater communication bandwidth and availability.

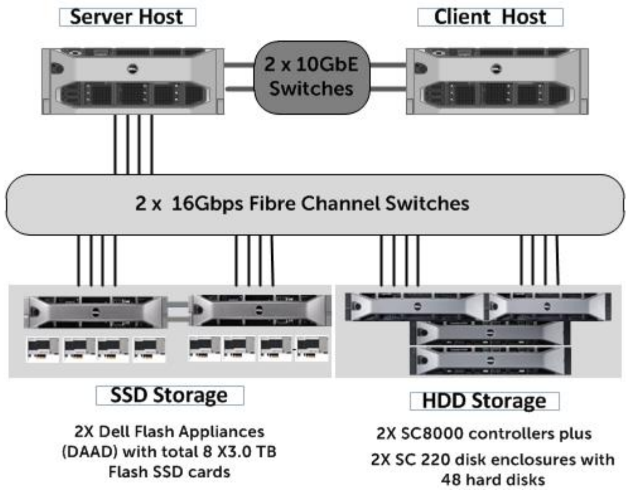
Server Host
Client Host
2 x 10GbE Switches
2 x 16Gbps Fibre Channel Switches
SSD Storage
2X Dell Flash Appliances (DAAD) with total 8 X3.0 TB Flash SSD cards
HDD Storage
2X SC8000 controllers plus
2X SC 220 disk enclosures with 48 hard disks

Fig. 4: Hardware Architecture and Components

The two host machines use Dell PowerEdge R910, a scalable four-socket server. Each host machine has four ten-core 3.20GHz Intel Xeon CPU E7-8891, totaling 40 processor cores, and it is equipped with 1TB DDR3 RAM. To promote client-server communication, each host machine has two dual-port 10GbE Ethernet network cards installed. To access FC-based storage, the server host has two dual port 16 Gb Host Bus Adapters (HBAs).

The two storage systems we compare are SSD based storage and HDD based storage. The SSD based storage is based on a flash storage appliance called Dell Acceleration Appliances for Databases (DAAD) [22, 23], which has two storage nodes, each containing 4×3 TB Flash SSD cards. In total, SSD based storage provides 24 TB of flash capacity. The HDD based storage system is hosted by Dell Compellent SC 8000 FC storage. This storage consists of two storage nodes (controllers) and two disk enclosures with total 48×15k rpm 250 GB hard disks. Across the two storage systems, each of the four storage nodes connects to the FC channel network via two dual ports 16 Gb HBAs.

A 2×10 GbE Ethernet switch network enables QuantCloud's client-server communication. Meanwhile, the FC storage network links the server host with both SSD based storage and the HDD based flash storage. Both of these dual switch networks provide necessary redundancy and double network bandwidth. The FC storage network connects the server host from its two dual ports 16 Gb to the Fibre channel (FC) switches. These links between the server host and FC switches provide a 64 Gb/s network bandwidth. On the storage end, each of the two storage systems connect to the FC switches with a total of four dual port 16 Gb HBAs, totaling 128 Gb/s in bandwidth.

This storage network configuration presents two storage volumes to the server:

/dev/ssd: based on 8 × 3.0 TB SSD flash storage.

/dev/hdd: based on 48 × 15k rpm 250 GB hard disks.

These two volumes were used to create the following Linux file systems backing the QuantCloud database, respectively: /SSD_ACFS and /HDD_ACFS.

With this design, a user sends specified XML formatted requests to a given client (Fig. 1), and the networking load between them is very small. Additionally, the user may send the requests from any location with an Internet connection. Thus, there is no hardware requirement for the networking between the user and the client.

5. Performance Targets

We aim to build a Cloud-based solution for providing high-performance DoD access and scalable task processes in the field of QF. To this end, we benchmark the system using the NYSE daily TAQ data (October to December 2014). Our tests specifically measure DoD access and process speeds, the scalability of our infrastructure, and the quality of service from a user perspective. Our results present first-hand numerical results for algorithmic traders and financial engineers.

5.1. Data Access and Process Speeds

WallClock time measures a test's completion time with a temporal resolution of microseconds. The results with regard to response time, latency and throughput are calculated at the client module. Data volume is measured in units of daily TAQ *message*. For a demand pattern, we study a user request for daily resolution data for the S&P 500 (i.e., Standard & Poor's 500), representing general analysis on historical marking data.

In results, we present two speed metrics: *throughput* (million messages per second) and *latency* (nanosecond per tick message). Throughput measures the specified infrastructure's capacity for data-transfer and data-process media. Latency measures the average process time on a message basis, indicating the capability of such critical big data analytics as high and medium frequency trading strategies.

5.2 Scalability

Speedup and parallel efficiency are calculated to show the scalability of our infrastructure. Our scalability tests vary the number of task processors at Server and Client. To this end, we use a single task processor's performance as a baseline. Then, we fix the application size and increase the number of task processors. We thus calculate the speedup and show the scaling efficiency for an application. After this assessment, we also calculate the parallel efficiency measures. The purpose of scalability testing is to identify bottlenecks that can impede the scalability of major applications and to provide numerical insights for building goal-driven infrastructure.

5.3 Quality of Service

Cloud computing is a model that allows sharing of a centralized datastore, data-process tasks, and access to computer resources. Though it is a more efficient solution for sharing competitive computing resources, utilizing personal machines to handle on-demand tasks, the Cloud also needs to treat concurrent user requests fairly [32]. To address this, we compute the *standard deviation* values of speeds for massive user requests in addition to the *average* values. Together, the average and standard deviation measures demonstrate the quality of service for a large number of big data analytics, characterizing a typical application in a Cloud environment.

6. Results

We present results ranging from straightforward data access to sophisticated data decomposition and process. In each test, we present the wallclock time and speeds, follow with scalability and quality of service, and finally analyze and discuss possible bottlenecks.

In the tests, we simulate a user request demand for NYSE TAQ data including Trade and BBO for the S&P 500 symbols. The NYSE TAQ (trade and quote) data[3] was presented in timesteps of seconds when it was introduced in 1997. Now the highest frequency of the TAQ data is at the millisecond level. Our database covers a volume of data from October 1, 2014 to December 31, 2014 and it has 64 weekdays. Data is stored in a compressed and hashed format in the datastore. Fig. 5 shows the data size distribution over weekdays and unveils an uneven distribution for the daily trading volume.

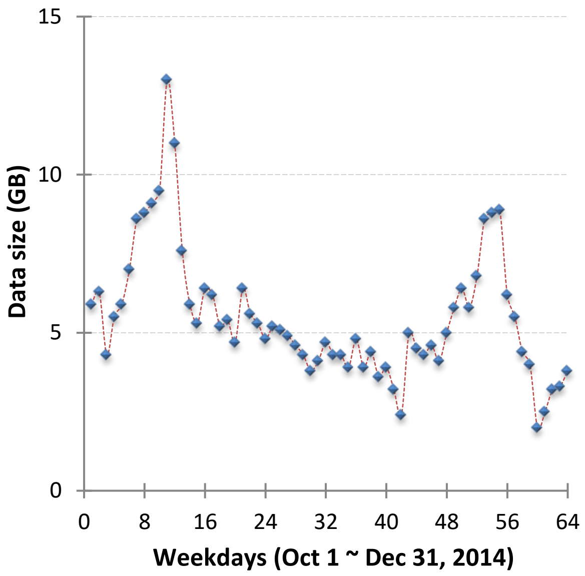

Fig. 5: Metadata size (in GB) distribution for 64 weekdays (Oct 1, 2014 ~ Dec 31, 2014)

6.1 Data-on-Demand Access

For this test, we use eight task processors at Server and vary the number of task processors at Client from one to ten. We create a total of 10 requests. Each request targets TAQ data for the 3 months and 100 symbols. Ten such requests require a total of 25.24 billion *tick* messages. The requests are concurrently submitted and enqueued at Client. Here, the *access* operation includes the request parsing and data packing at server, data transfer from datastore to a specified client, and data consolidation at the destination client. Comparative tests are performed between the SSD and HDD media.

Fig. 6 shows the wallclock time (in seconds) vs. different number of task processors at Client using different specified big data-storage media: (i) SSD and (ii) HDD. Fig. 7 and Fig. 8 display the throughput and latency measures, respectively.

These results show that: (i) The SSD is consistently superior to HDD as a datastore medium (Fig. 6); (ii) The parallelization of data access between server and client efficiently exploits the throughput and lowered latency, as shown in Fig. 7 and Fig. 8, respectively. In absolute speeds, our proposed infrastructure achieved a throughput of 276 million tick messages per second and a latency of 3.6 nanoseconds per tick message. This demonstrates a hallmark of accessing a large volume of on-demand data down to nanosecond-scale latency. To be reproducible, we achieved this record by designing parallel algorithms on commodity products.

Fig. 9 and Fig. 10 show speedup and parallel efficiency, respectively. This comparison result shows that SSD has a stronger speedup than HDD (Fig. 9). Multiple server-client connections help exploit the network bandwidth while lowering latency for data transfer over Internet connections. However, at the datastore server, moving data from a primary storage medium to the main memory is determined by the datastore medium. In this process, SSD outperforms HDD. SSD demonstrates stronger speedup and parallel efficiency as

[3] Daily TAQ data: *http://www.nyxdata.com/Data-Products/Daily-TAQ*

compared with HDD. This suggests that a fast datastore medium is requisite for supporting the parallelized infrastructure.

Second, the latency is calculated at the client module, which means the total time includes not only the data read but also the data packing at server and transfer over the network. When the number of task processors is small e.g. 1 or 2, the data packing and transfer strongly impact latency. Hence, although the SSD is faster than the HDD on the read, the total latency difference may be insignificant (Fig. 8). However, as the number of task processors increase, SSD demonstrates a clear advantage (Figs 7-8).

Fig. 11 shows the quality of service on a request basis, displaying a small standard deviation from the average. The smaller standard deviation indicates higher quality service. There is no prolonged request, and our infrastructure can easily predict the service.

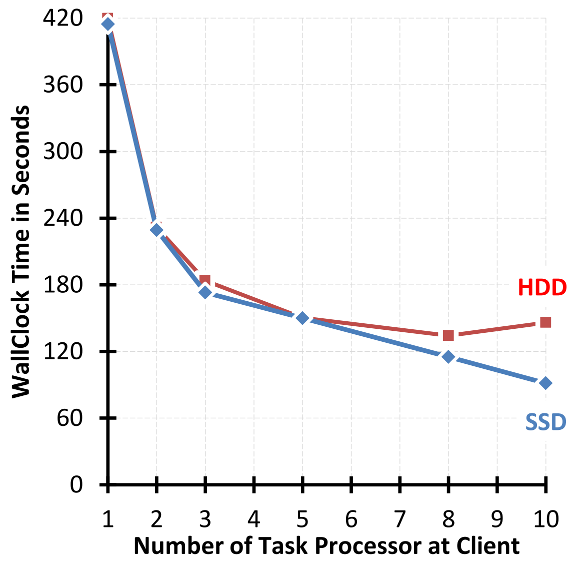

Fig. 6: Wallclock time (in seconds) vs. number of task processor at Client for the SSD and HDD media

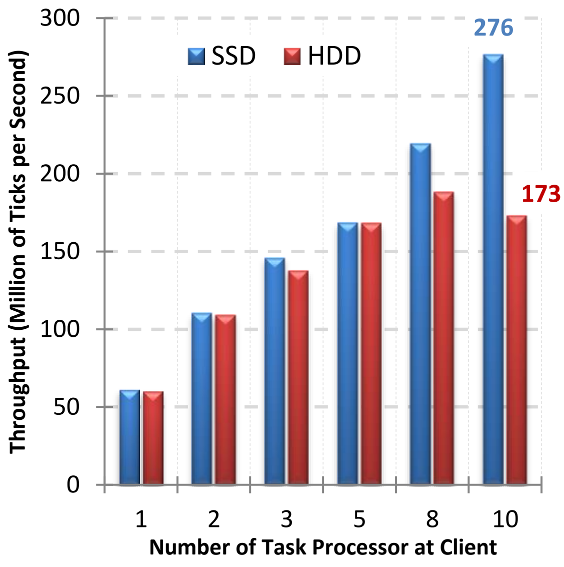

Fig. 7: *Throughput* (in million messages per second) vs. number of task processors at Client for SSD and HDD

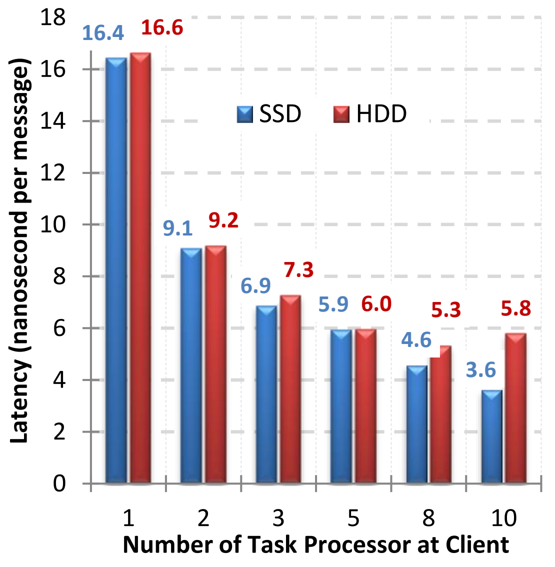

Fig. 8: *Latency* (nanosecond per message) vs. number of task processors at Client for the SSD and HDD media

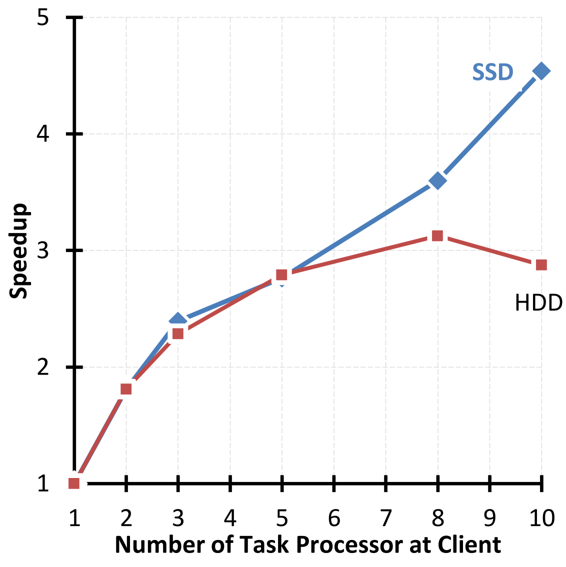

Fig. 9: Speedup for different number of task processors at client for the SSD and HDD media

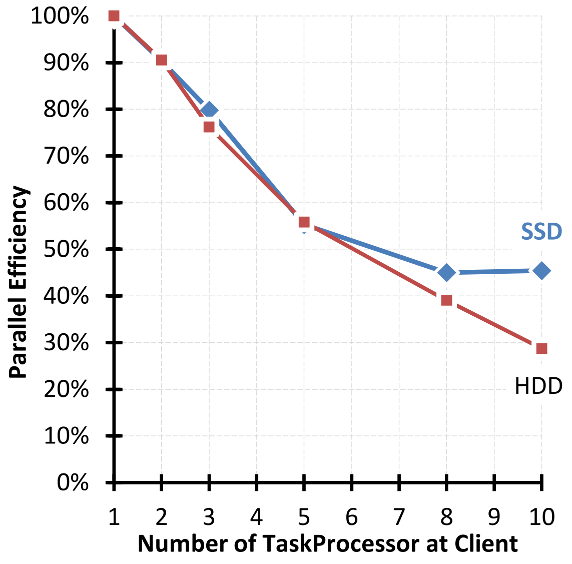

Fig. 10: Parallel efficiency for different number of task processors at client for the SSD and HDD media

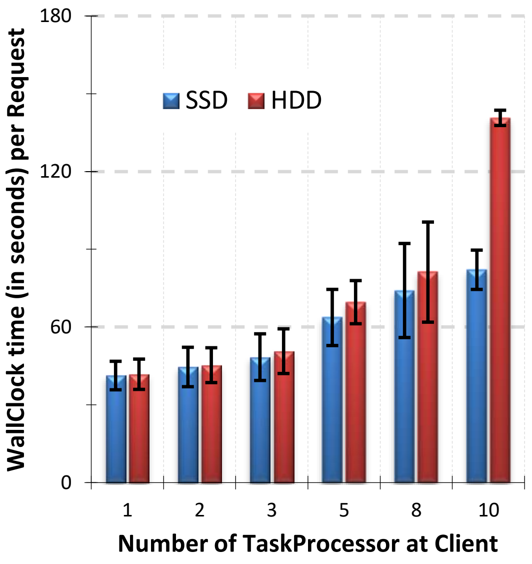

Fig. 11: Quality of Service: wallclock time (in seconds) per request vs. number of task processors at client for SSD and HDD

6.2 Data Decompression and Process

Extending previous efforts in the data *access* operation, we continue by testing the *data decompression and process* (DDP) operation. Here, data decompression refers to the process of decompressing zipped data at Client. The decompressed data are further processed: they are first restored to individual tick messages where certain fields need to be dehashed, and then they are categorized by symbols and sorted by time. After this process concludes, the original compressed and hashed metadata can be conveniently utilized and programmed by algorithmic traders for their strategies and research. This process involves many time-consuming and interdependent steps. The challenges include handling a variety of data structures for diversified fields in message, regrouping and sorting a big volume of data, and building user-friendly data structures that will be used by algorithmic traders.

Our test employs SSD as the datastore medium due to its superior performance. We start by varying the number of task processors at Client and Server, respectively, and studying their impacts and characteristics. Finally, we study the impact of multithreading on performance. The results can characterize performance from a real-world application standpoint.

6.2.1 Task Processors at Client

In this test, we use eight task processors at Server and vary the number of task processors at Client from one to eight. At Client, we use 32 threads per task processor. We create a total of 128 requests. Each request is for a single-day TAQ data of 100 symbols listed in the S&P 500. This represents an application pattern for querying and processing the daily market data. 128 requests require a total of 25.24 billion messages. These requests are concurrently submitted and enqueued at Client.

Figs 12 to 17 present results and analysis. Within legends in these figures, "Comm" refers to the *access* operation with no DDP. "Comm & Compt" means a combination of the access and the DDP. The horizontal axis represents the number of task processors at Client; and the vertical axes represent the measures and their units.

These results show that the DDP is a very time-consuming operation compared to the access operation (Fig. 12). The throughput drops to 74 million messages per second, which is only ¼ of the access-only throughput (Fig. 13). In addition, latency increases to 13.5 nanoseconds per message (Fig. 14). Multiple task processors help increase performance (Fig. 15). A superlinear speedup appears at 2 task processors (Fig. 16). Thus, parallelism is a favorable way of improving market data processing toward high-throughput analytics. Lastly, in terms of QoS, the average latency per request increases and so does the standard deviation as the number of task processors increases (Fig. 17). This is because more processes escalate the contention over limited computer resources. This contention, though improving overall performance, prolongs per-request processing time.

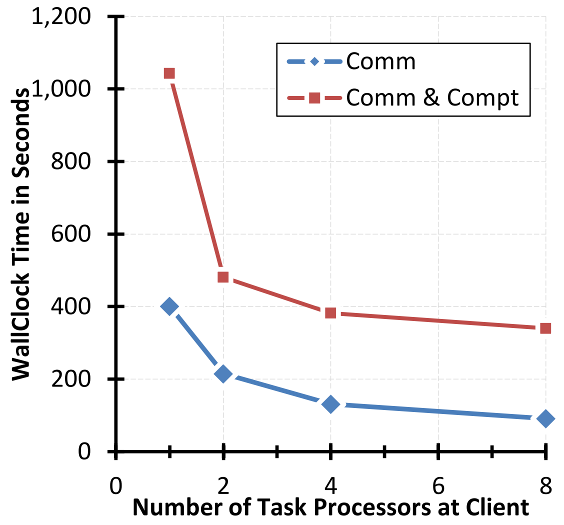

Fig. 12: Wallclock time (in seconds) vs. different number of Task Processor at Client

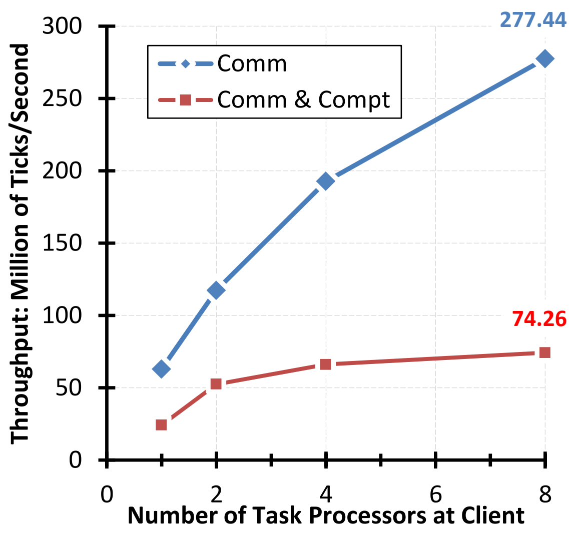

Fig. 13: *Throughput* (in million messages per second) vs. number of task processors at Client

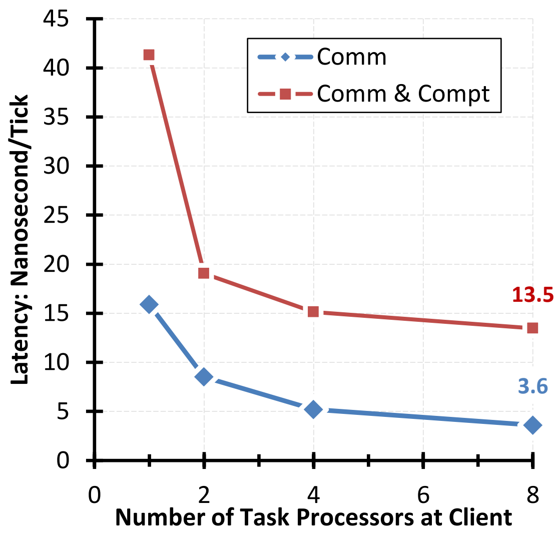

Fig. 14: *Latency* (nanosecond per message) vs. number of task processors at Client

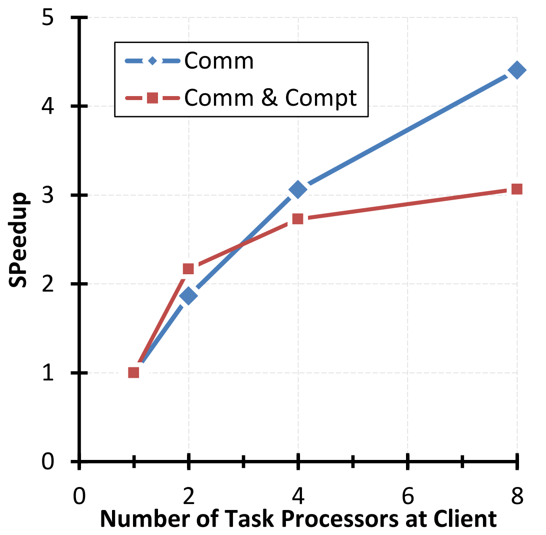

Fig. 15: Speedup for different number of task processors at client

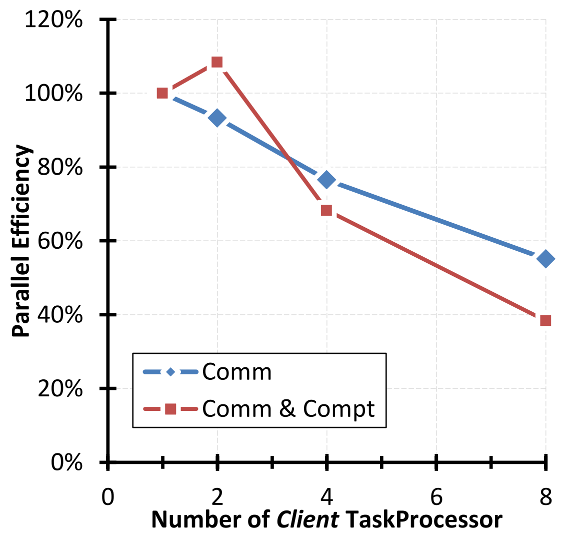

Fig. 16: Parallel efficiency for different number of task processors at client

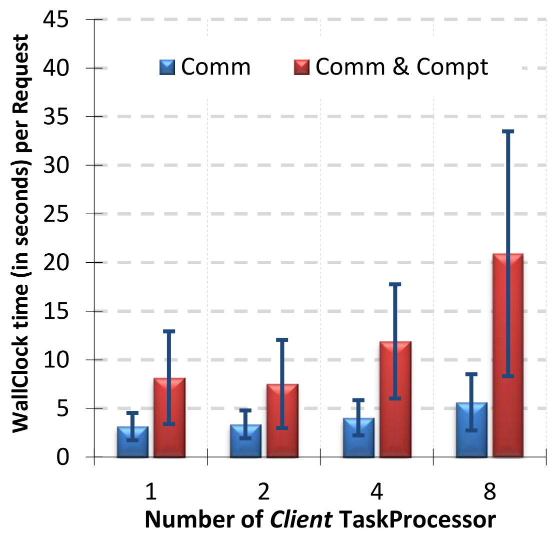

Fig. 17: Quality of Service: wallclock time (in seconds) per request vs. number of task processors at client

6.2.2 Task Processors at Server

In this test, we vary the number of task processors at server while keeping eight task processors at client. Other configuration details are identical to those in Section 6.2.1.

Fig. 18 shows the wallclock time for different numbers of task processors at server, and Fig. 19 shows the per-request quality of service. The communication is between the server and client, but computation mainly takes place on the client.

The results show that the number of task processors at server impacts the Access performance (Comm) greatly, since more network connections can improve the data transfer between server and client. On the other hand, the DDP performance (Comm & Compt) depends more on the computing capability at client. This also demonstrates that the SSD-based datastore is an efficient platform.

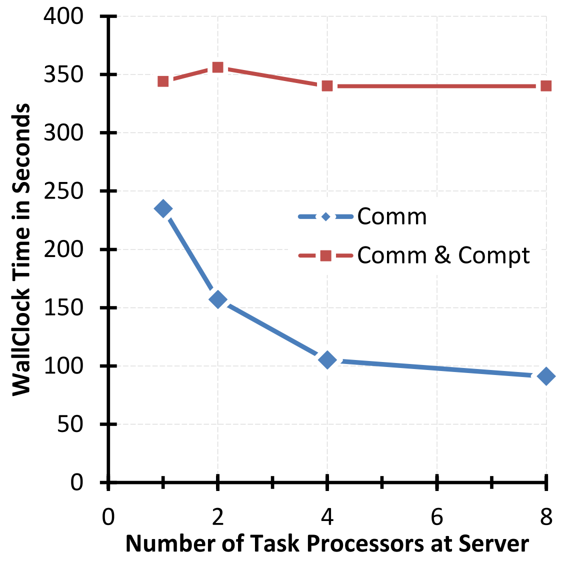

Fig. 18: Wallclock time (in seconds) vs. number of Task Processor at Server

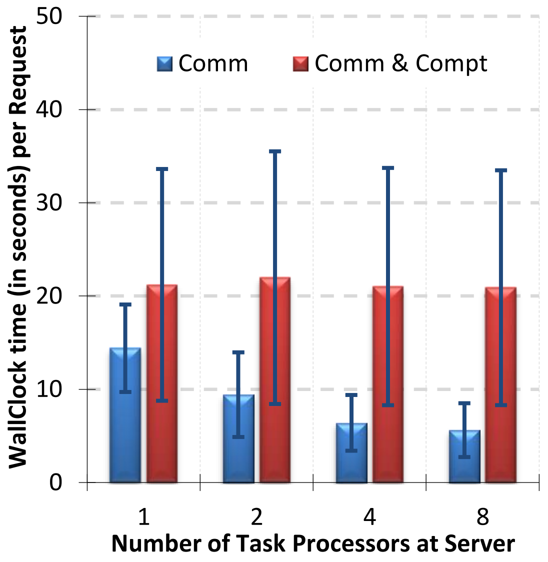

Fig. 19: Quality of Service: wallclock time (in seconds) per request vs. number of task processors at server

6.2.3 Multithreading at Client

In this test, we continue to study the impact of multithreading at client. We use eight task processors at server and client. Other configuration details are identical to those in Section 6.2.1.

Fig. 20 shows processing accelerates considerably between 1 and 4, but from 4 onwards differences are negligible. The optimum number of threads appears at 4. Meanwhile, the quality of per-request service demonstrates a similar trend (Fig. 21).

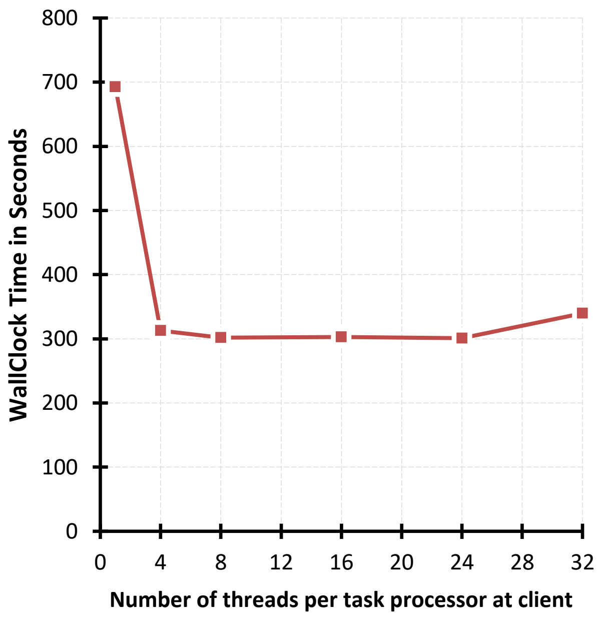

Fig. 20: Wallclock time (in seconds) vs. number of threads per task processor at Client

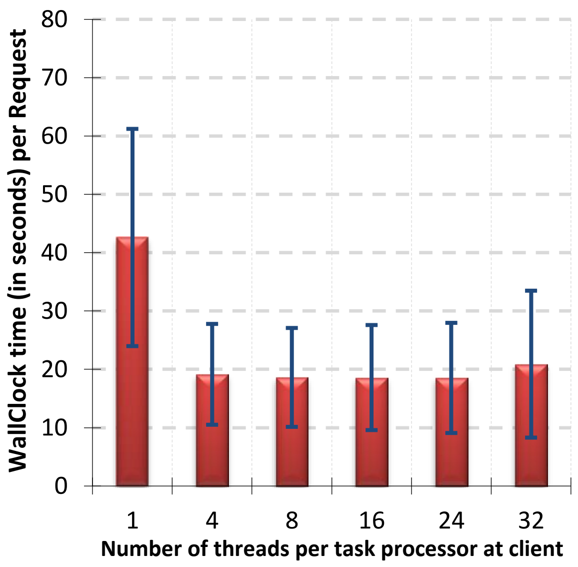

Fig. 21: Quality of Service: wallclock time (in seconds) per request vs. number of task processors at server

6.3 Petabyte-Scale Data Processing

In this test, we study QuantCloud performance in handling petabyte-level diversified requests. Request type includes those for daily, weekly, monthly and quarterly marketing data. Each request is for S&P 500 symbols. Table 2 shows the request configuration. In total, we handle over 200 billion messages, 4 different request types, and 328 requests. In general, BBO and Trade message structures have 59 and 38 bytes, respectively. The amount of BBO data is approximately one order of magnitude larger than that of Trade data, so the total size of data is over 11 petabytes. Thus, this test represents a petabyte-scale data processing challenge.

In Server, we initiate eight task processors. In Client, we initiate ten task processors, each using eight threads for data decompression and processing.

Fig. 22 shows the response time (in seconds) histogram for user requests. For this test, wallclock time is 3,192 seconds, the average latency is 15.8 nanoseconds per message, and the aggregate throughput is 63.3 million messages per second. That is, the 11-petabyte application consisting of widely differentiated requests completes in 53 minutes. This record implies that petabyte-level big data analytics can be completed in only a few minutes.

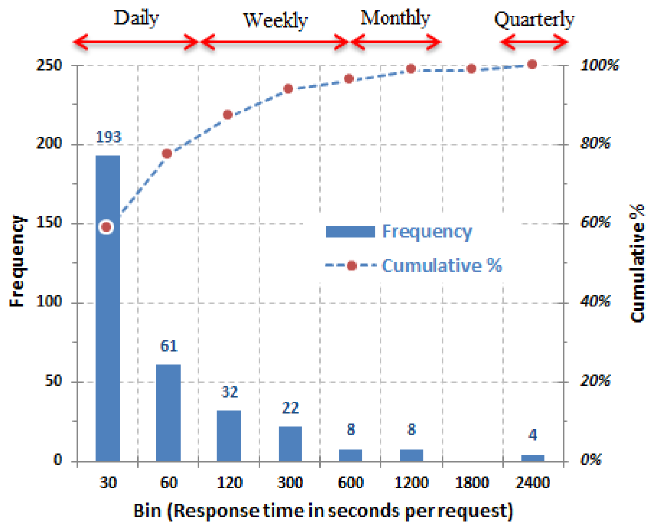

Fig. 22: Response time histogram for user requests

TABLE 2:
CONFIGURATION FOR TRILLION-SCALE DATA PROCESSING TEST

Request Type:	# of Requests	Billions of Messages
Daily	256	50.495
Weekly	56	50.495
Monthly	12	50.495
Quarterly	4	50.495
TOTAL:	**328**	**201.979**

7. DISCUSSION

We examine the performance of QuantCloud infrastructure for conducting big data analytics in the field of quantitative finance. The evaluation environment is built using today's commodity components. The results provide the community of algorithmic traders and financial engineers with numerical insights and exemplify the significance of parallelism in development.

- SSD is superior to HDD in providing disk-backed storage system (Section 6.1). In particular, SSD performs faster than HDD when more task processors of the client require data concurrently. In terms of latency, the SSD-backed and HDD-backed approaches feed market data at 3.6 and 5.3 nanoseconds per message, respectively – the former is 32% faster than the latter (Fig. 8). In addition, regarding aggregate throughput, the SSD-backed and HDD-backed approaches provide a sustained throughput at 276 and 173 million messages per second, respectively. In other words, an SSD-backed approach only needs 3.6 seconds to offer the service for 1 billion TAQ messages. In our study, SSD-backed storage demonstrates superiority in terms of speedup and parallel efficiency, which determine scalability. As predicted, utilizing the faster SSD storage system significantly increases throughput and reduces latency, resulting in dramatic improvements in database IO performance and database performance as a whole.
- Data decompression and process (DDP) is a more time-consuming procedure than the data access procedure (Section 6.2). We greatly accelerate DDP by adopting the parallel processing pattern (Section 6.2). In particular, multithreading is very effective in improving computing efficiency (Section 6.2.3).
- Lastly, we demonstrate the petabyte-level data analytics can be processed within only a few minutes (Section 6.3). In particular, an 11-petabyte application that consists of diverse data requests is completed in 53 minutes.

To summarize, these performance results demonstrate the applicability of QuantCloud infrastructure for processing Petabyte and Terabyte-level data analytics within an affordable computational timeframe (the minute timescale).

In the future, we would like to deploy this infrastructure on public cloud platforms. Furthermore, we would add an on-demand computing module for optimizing performance within a given operational budget. We would enhance data protection and privacy policies by hashing and encrypting user-defined trading strategies and parameters. This is a first step towards a high-performance big data infrastructure for quantitative finance.

8. CONCLUSION

In this paper, we present the QuantCloud design for handling the big data applications in the field of quantitative finance. We perform experiments and compare results on SSD and HDD datastore mediums. The results show that our design achieves a record of processing a billion tick messages in the second timescale, and that we require a parallel scheme for efficiently exploiting infrastructural benefits.

We provided computational insights for performing big data analytics in quantitative finance. We demonstrated the computational abilities of modern commodity products for processing a billion-scale problem in a second timescale. Specifically, our design exemplified the impact of parallelization on boosting performance. The aggregate impacts of design, parallel algorithms, and sophisticated implementations offer the community of algorithmic traders and developers' new hope and state-of-the-art insights for their research and development.

ACKNOWLEDGEMENTS

Dr. Zhang, P. developed the QuantCloud software. Dr. Yu, K built the hardware platform. Dr. Khan SU advised and improved this manuscript. Ms. Yu, J conducted some experiments, tuned the hardware system, and contributed to the writing and revising of this manuscript.

Samee U. Khan's work was supported by (while serving at) the National Science Foundation. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] W. Härdle, T. Kleinow, and G. Stahl, *Applied Quantitative Finance*: Springer Berlin, 2002.

[2] E. Thorp, "A perspective on quantitative finance: Models for beating the market," *The Best of Wilmott,* p. 33, 2005.

[3] B. Fang and P. Zhang, "Chapter 11: Big Data in Finance," in *Big Data Concepts, Theories, and Applications*, S. Yu and S. Guo, Eds., ed: Springer Switzerland, 2016. DOI: 10.1007/978-3-319-27763-9_11

[4] X. Shi, P. Zhang, and S. U. Khan, "Quantitative Data Analysis in Finance," in *Handbook of Big Data Technologies*, A. Zomaya and S. Sakr, Eds., ed: Springer International Publishing, 2016. DOI: 10.1007/978-3-319-49340-4_21

[5] J. Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh*, et al.*, *Big data: The next frontier for innovation, competition, and productivity*: McKinsey Global Institute, 2011.

[6] H. Chen, R. H. Chiang, and V. C. Storey, "Business Intelligence and Analytics: From Big Data to Big Impact," *MIS Quarterly,* vol. 36, pp. 1165-1188, 2012.

[7] J. J. Angel, L. E. Harris, and C. S. Spatt, "Equity trading in the 21st century," *The Quarterly Journal of Finance,* vol. 1, pp. 1-53, 2011.

[8] C. Alexander, *Market models: a guide to financial data analysis*: John Wiley & Sons, 2001.

[9] A. J. McNeil, R. Frey, and P. Embrechts, *Quantitative Risk Management: Concepts, Techniques and Tools: Concepts, Techniques and Tools*: Princeton University Press, 2015.

[10] A. J. Menkveld, "High frequency trading and the new market makers," *Journal of Financial Markets,* vol. 16, pp. 712-740, 2013.

[11] M. O'Hara, "What is a quote?," *The Journal of Trading,* vol. 5, pp. 10-16, 2010.

[12] A. P. Chaboud, B. Chiquoine, E. Hjalmarsson, and C. Vega, "Rise of the machines: Algorithmic trading in the foreign exchange market," *The Journal of Finance,* vol. 69, pp. 2045-2084, 2014.

[13] E. Boehmer, K. Y. Fong, and J. J. Wu, "International evidence on algorithmic trading," in *American Finance Association (AFA) 2013 San Diego Meetings*, 2014.

[14] M. Lewis, "An Adaptation From 'Flash Boys: A Wall Street Revolt'," in *The New York Times*, ed, 2014.

[15] A. Carrion, "Very fast money: High-frequency trading on the NASDAQ," *Journal of Financial Markets,* vol. 16, pp. 680-711, 2013.

[16] R. Kissell and R. Malamut, "Algorithmic decision-making framework," *The Journal of Trading,* vol. 1, pp. 12-21, 2006.

[17] M. Fox, "Thomson Reuters releases "Big Data in Capital Markets" survey results," in *Thomson Reuters*, ed, August 12, 2014.

[18] C. Beattie and B. Meara, "How Big Is Big Data? Big Data Usage and Attitudes among North American Financial Services Firms," in *Oliver Wyman*, ed, 2014.

[19] J. Coumaros, S. d. Roys, L. C. Leroyer, J. Buvat, and O. Auliard, "Big Data Alchemy: How can Banks Maximize the Value of their Customer Data?," Capgemini Consulting2014.

[20] H. Plattner and A. Zeier, *In-Memory data management: Technology and applications*: Springer Science & Business Media, 2012.

[21] S.-W. Lee, B. Moon, C. Park, J.-M. Kim, and S.-W. Kim, "A case for flash memory ssd in enterprise database applications," presented at the Proceedings of the 2008 ACM SIGMOD international conference on Management of data, Vancouver, Canada, 2008.

[22] K. Yu, Y. Gao, P. Zhang, and M. Qiu, "Design and Architecture of Dell Acceleration Appliances for Database (DAAD): A Practical Approach with High Availability Guaranteed," in *High Performance Computing and Communications (HPCC), 2015 IEEE 7th International Symposium on Cyberspace Safety and Security (CSS), 2015 IEEE 12th International Conferen on Embedded Software and Systems (ICESS), 2015 IEEE 17th International Conference on*, 2015, pp. 430-435.

[23] K. Yu, W. Vaske, and C. Murphy. (2014). *High Performance Oracle Workloads with the Dell Acceleration Appliances for Databases.* Available: http://en.community.dell.com/techcenter/enterprise-solutions/w/dell_acceleration_appliance_for_databases_daad

[24] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation Computer Systems,* vol. 25, pp. 599-616, Jun 2009.

[25] D. Kliazovich, P. Bouvry, and S. U. Khan, "GreenCloud: a packet-level simulator of energy-aware cloud computing data centers," *The Journal of Supercomputing,* vol. 62, pp. 1263-1283, 2012.

[26] I. A. T. Hashem, I. Yaqoob, N. B. Anuar, S. Mokhtar, A. Gani, and S. U. Khan, "The rise of "big data" on cloud computing: review and open research issues," *Information Systems,* vol. 47, pp. 98-115, 2015.

[27] Y. Deng, P. Zhang, C. Marques, R. Powell, and L. Zhang, "Analysis of Linpack and power efficiencies of the world's TOP500 supercomputers," *Parallel Computing,* vol. 39, pp. 271-279, 2013.

[28] TOP500. *Top 500 Supercomputer Site*. Available: http://www.top500.org

[29] P. Zhang, R. Powell, and Y. Deng, "Interlacing Bypass Rings to Torus Networks for More Efficient Networks," *Parallel and Distributed Systems, IEEE Transactions on,* vol. 22, pp. 287-295, 2011.

[30] E. Holley, "Cloud in financial services – what is it not good for?,"

in *Business Cloud News*, ed, June 2, 2014.

[31] A. N. Khan, M. M. Kiah, S. U. Khan, and S. A. Madani, "Towards secure mobile cloud computing: A survey," *Future Generation Computer Systems,* vol. 29, pp. 1278-1299, 2013.

[32] A. Sedighi, Y. Deng, and P. Zhang, "Fariness of task scheduling in high performance computing environments," *Scalable Computing: Practice and Experience,* vol. 15, pp. 273-285, 2014.

Peng Zhang received the BS degree in mathematics from Nankai University in 2003 and the MS degree in parallel computing from Nankai Institute of Scientific Computing in 2006, and the PhD degree in applied mathematics from Stony Brook University, New York, USA in 2012. Currently, he is a Senior Research Associate at Stony Brook University, NY, USA. Dr. Zhang's research interests include development and enhancement of models, algorithms, software and problem-solving environments for domain-specific applications that reply on the high performance computing (HPC) technologies. His work has appeared in over 50 publications and presentations. He is the member of IEEE.

Jessica Yu is a current undergraduate studying Computer Science at the Massachusetts Institute of Technology (MIT). Previously, she has contributed to high-performance computing projects at the Texas Advanced Computing Center (TACC) and worked in the IBM Tivoli Storage Productivity Center (TPC). She was named a Regional Finalist in the 2013 Siemens Competition in Math, Science & Technology as well as a National Runner-Up for the 2014 National Center for Women & Information Technology (NCWIT) Award for Aspirations in Computing. Her research interests include cloud computing and machine learning.

Kai Yu is a senior principal engineer and technologist in Dell Enterprise Solutions Engineering Lab. In the past 23 years, he has been focusing on architecture design of various enterprise computing solutions. He received MS degrees in Computer Science from both University of Wyoming and Huazhong University of Science and Technology. He has co-authored one book and authored 30 technology articles and whitepapers and given 120 presentations in the global IT technology conferences. He was awarded the Oracle ACE Director title in 2010 by Oracle Technology Network and the "2012 Oracle Excellence Award: Technologist of the Year for Cloud Architect" by Oracle Magazine.

Samee U. Khan received a BS degree in 1999 from Ghulam Ishaq Khan Institute of Engineering Sciences and Technology, Topi, Pakistan, and a PhD in 2007 from the University of Texas, Arlington, TX, USA. Currently, he is Associate Professor of Electrical and Computer Engineering at the North Dakota State University, Fargo, ND, USA. Prof. Khan's research interests include optimization, robustness, and security of: cloud, grid, cluster and big data computing, social networks, wired and wireless networks, power systems, smart grids, and optical networks. His work has appeared in over 300 publications. He is on the editorial boards of leading journals, such as IEEE Access, IEEE Cloud Computing, IEEE Communications Surveys and Tutorials, and IEEE IT Pro. He is an ACM Distinguished Speaker, an IEEE Distinguished Lecturer, a Fellow of the Institution of Engineering and Technology (IET, formerly IEE), and a Fellow of the British Computer Society (BCS).