Ke Ai
ADVANCING RESEARCH
EVOLVING SCIENCE

CrossMark

Available online at www.sciencedirect.com

ScienceDirect

The Journal of Finance and Data Science 1 (2015) 33–41

http://www.keaipublishing.com/en/journals/jfds/

Latency critical big data computing in finance

Xinhui Tian [a,b,*], Rui Han [a], Lei Wang [a,b], Gang Lu [a,b], Jianfeng Zhan [a]

[a] *State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, China*
[b] *University of Chinese Academy of Sciences, China*

Received 6 March 2015; revised 14 March 2015; accepted 30 March 2015
Available online 28 August 2015

Abstract

Analytics based on big data computing can benefit today's banking and financial organizations on many aspects, and provide much valuable information for organizations to achieve more intelligent trading, which can help them to gain a great competitive advantage. However, the large scale of data and the critical latency analytics requirement in finance poses a great challenge for current system architecture. In this paper, we first analyze the challenges brought by the financial latency critical big data computing, then propose a discussion on how to handle these challenges from a perspective of multi-level system. We also talk about current researches on low latency in different system levels. The discussions and conclusions in the paper can be useful to the banking and financial organizations with the critical latency requirement of big data analytics.

Keywords: Big data; Financial analytics; Latency

1. Introduction

As one of the essential factors of system and network performance, latency indicates how fast a user can get a response after the user sent out a request. Low latency, which means systems response quickly to actions, can make users feel more natural and fluid than long response time.[1] In the financial market, as more and more business trades and banking operations are executed online, lower latency now means more revenues, especially for companies which adopt high frequency trading to earn huge profit. High frequency trading means to rapidly trade large volumes of securities by using automated financial tools.[2] A millisecond decrease in a trade delay may boost a high-speed firm's earnings by about 100 million per year,[3] and also helps a firm to gain great competition advantage.

Traditionally, financial organizations can achieve low latency via adopting high performance computers, which provide great processing capability, especially the capability of floating-point processing. When the processing capability is not enough, high performance computers can also be scaled via two methods, which are scale up (adding more CPUs or memory to a single computer) and scale out (adding more computing nodes, and connecting them with

* Corresponding author. State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, China.
E-mail address: tianxinhui@ict.ac.cn (X. Tian).
Peer review under responsibility of China Science Publishing & Media Ltd.

http://dx.doi.org/10.1016/j.jfds.2015.07.002

high performance interconnects). However, as the size of data needed to be analyzed is growing dramatically in the last few years, the primary bottleneck has shifted to the performance of storage system, and the frequent data movement in traditional high performance computing can significantly impact the latency when the volume of processing data is huge. Therefore, the system architecture for financial computing needs to be improved in such a situation.

Such data explosion problem can also be called as the big data problem, which has been a hot trend in recent years. The big data means that the collected data sets are becoming too large and complex to be processing via traditional data processing applications.[4] IDC forecasts that the world's data will be doubling every two years with 1.8 trillion gigabytes expected to be created,[5] and the global volume of data will increase from 130 to 40,000 exabytes by 2020.[6] Another report from HP also presents that the data size in financial world is really big now. For example, there are more than 10,000 payment card transactions executing per second across the world in 2012, and total number of U.S. online banking households is about 66 million in 2014.[7] The New York Stock Exchange also has to process about 2 TB data daily in 2012, and expects to exceed 10 PB a day by 2015.[8]

The large scale of data contain enormously valuable information, and analytics based on big data can provide financial organizations with more business opportunities and the possibility to gain a more holistic view of both market and customers. Big data analytics can benefit banking and financial market firms in many aspects, such as accurate customer analytics, risk analysis and fraud detection. These approaches can lead to smarter and more intelligent trading, which can help organizations to avoid latent risks and provide more personalized services, thus to get a higher degree of competition advantage.

According to the report from SAP,[3] the profitability keeps falling in recent years, and organizations are now evolving towards smart trading based on big data analytics. Besides designing more complex computing model and system, how to make such large scale computation real time is still an very important problem that is needed to be considered seriously. In fact, many approaches of big data analytics would not be beneficial if the latency were not be controlled in a low level, especially for some high speed analytics such as the risk management of stock trading. Another fact is that, the rapid growing data flood also make the political, social and economic events that can impact the financial markets, now take only a few minutes, which may take days before. Such a challenge requires financial organizations process the events immediately as they are acquired. A complex computation may also demand process the new coming data together with some historical records, which can involve a huge size of input data. For example, NYSE Euronext has employed big data analytics to detect new patterns of illegal trading, and they need to process about two terabytes of data daily and get the analytics result in near real time.[9] Hence, this is difference from traditional data stream problem. Researches on low latency and high throughout data stream processing on a big data store are desired.

Consequently, the latency critical big data analytics in finance, which require the latency to be kept in a critical level, requires the system to satisfy many different types of latency demands. One is the absolute end-to-end latency, which directly decides the speed of market access and real time trading. Another is the new demand on how to process complex analytics on huge amounts of data faster than competitors to capture trading opportunities. As the data size keeps growing rapidly, many organizations require the analytics latency close to near real time, thus to be able to extract valuable information for competitive advantage. This demand poses a great challenge to current system architecture with many difficult problems. For example, how to organize such large scale of data for history analytics, how to process the streaming data immediately, and how to effectively execute jobs with different priorities. All these problems are the hurdles that has to be moved for smarter trading. In this paper, we mainly discuss how to guarantee the second latency demand from a multi-level system perspective. We first give a detailed discussion on the challenges to achieve low latency big data computing in section 2. In Section 3, we give a discussion about a multi-level system solution for this challenge. We also discuss recent researches on latency critical big data computing in both industry and academia in Section 4.

2. Challenges

In this section we try to discuss and summary the challenges to achieve low latency financial big data analytics. The first problem is how to deal with the problem of massive data storage and organization. Many organizations need to keep historic data of many years for trend prediction and other complex analytics, which poses a great challenge for the reliability of the storage system. The rapidly growing data also require the storage architecture to provide good scalability to support scaling out when the data size increases up to the storage boundary. Another problem is the

various data types. Most of big data is unstructured today, as the data are mostly generated from many sources including web pages, media and users' logs. This kind of data does not have a pre-defined data model, thus cannot fit into the schemas of existing relational databases, and require enterprises to shift to more suitable solutions such as NoSQL databases.

The big data storage problem can be solved via developing a distributed file system deploying on hundreds or thousands of computers, with effective fault tolerance and data balance algorithm. This kind of storage architecture can also provide high I/O bandwidth to improve data load performance. Many successful approaches from Internet services, such as Hadoop distributed file system (HDFS)[10] are based on this idea. The rapidly growing size of unstructured data also drive the development of distributed NoSQL database, which do not use a fixed schema to organize data. Approaches such as BigTable,[11] and Dynamo[12] have been developed and widely used in many different enterprises. Financial and banking organizations can directly adopt well developed solutions for their own scenarios, and optimize the data loading procedure to achieve low latency data access.

The main purpose of financial big data analytics is to effectively extract valuable information, thus another problem is how to design a high efficient computing system to process the distributed historical and incoming data. The traditional computation-centric model is no longer suitable for this scenario due to the frequent data movement. To reduce this I/O overhead, researchers suggest to move computation to the server where data locate, which means the computations need to be split into many little tasks to be sent to each data slice. Programming models based on this pattern, such as MapReduce,[13] can significantly reduce the great network I/O overheads caused by big data processing, and fully utilize the high disk bandwidth. However, the main purpose of MapReduce is not to support low latency processing. Even though later approaches such as Spark[14] and Impala[15] try to utilize local memory to speed up computation, the improvement is still not enough for real time analytics.

Many issues remain to be researched in this field. For example, how to make good use of high performance hardware, such as GPU and SSD to accurate the local computation on each node, thus to speed up the whole computation.

For many financial organizations, however, data should be processed immediately as it is ingested for more critical latency computations. This feature is required by many financial analytics, such as risk management and illegal trading detection. Moreover, the analytics of new coming data may need to fetch historical data, so the approaches can be built based on a high performance programming model. To achieve low latency data stream analytics, how to optimize the usage of memory to keep temporal data in memory is also an important aspect.

Another problem created by the increasing scale of the data centers is the tail latency problem.[16] A big data computation job is always split into multiple stages, while each stage is pipelined to execute on each node. A slow node can cause significant increase on response time, as the whole job has to wait for the partial result generated by this straggler. This straggler problem can cause the variability of response time, and therefore lengthen the tail of latency distribution.

Another challenge is the problem of concurrent jobs in the computing system. To achieve high throughput, it is normal to allow large batch jobs and small interactive jobs to converge in the computing environment. If the system can not effectively schedule different types of jobs, the small jobs may have to wait for the execution of large ones, which can cause unpredictable delay. Therefore, the computing platform should be able to automatically dedicate the priorities of jobs. The scheduling overhead should also be controlled to promise to process interactive jobs in near real time.

3. Multi-level system architecture for latency-critical financial analytics

In this section, we will discuss about how to handle challenges mentioned in the previous section. First we give a brief description of high performance computing, and discuss the advantages that can be used in low latency financial big data analytics.

When we talk about high performance computing, we mainly mean the super computing in this paper. Super computing focuses more on the capacity, which tries to use cost-effective computing power to solve a problem with the size and complexity hardly handled by regular computers. The performance of super computers is measured by Flops, which is short for Floating-point Operations Per Second, and Flops/W is used to benchmark the power efficiency. Linpack benchmark[17] is a widely used measurement of the system floating point performance, which computes a

dense system of linear equations. This benchmark is also used for the top500 list, which shows the rank of 500 most powerful super computers in the world.

Some vendors have made efforts to bring the advantages of super computing to big data analysis. Nvidia and IBM are trying to use GPU for database accelerating. This new system is taking a CPU and GPU heterogeneous architecture and the performance can reach about 12 times than today.[18] Intel also brought HPC support to the Apache Hadoop software, and built a HPC distribution for Hadoop. The new generation Intel CPU provides developers with the ability of running applications entirely natively, rather than offloading data to the coprocessor, which can significantly reduce the complexity of programming, and decrease latencies caused by memory, or other I/O devices.[19] Moreover, network technologies developed for super computing, such as InfiniBand,[20] can also benefit large scale cluster infrastructure with high network throughput.

Another essential factor that have to be considered when designing hardware systems is the power cost. If the profit from decreasing latency is less than the increment of power cost, vendors and enterprises will refuse the change. Consequently, the calculating capability of data center should be evaluated related to the cost of power. For instance, the request number of Google Search in one day is about 3.9 billion. We assume that the average latency requirement is 0.1 s. With the power less than 1 MW, the calculating capability should be about 5 Gbps/W.

Therefore, current researches on HPC have provide many great features to support low latency big data analytics. As the calculating capability increases, banking and financial organizations can gain great ability to run more complex event processing with less power cost.

The lowest level of the software system architecture is the single machine operating system (OS), which is responsible for hardware resource management and scheduling. Resources are shared by many concurrent processes running on the same machine, and if the OS scheduler can not work well, the resource utilization can not be kept in a high level, thus to waste the advantages of high performance hardware. Low resource utilization can also cause tail latency and impact the scalability on the single machine, hence we suggest that the scheduling strategy in OS level should be selected and optimized carefully according to the hardware resources and application types. Meanwhile, the scheduling strategy should not bring extra overheads.

The computation of Big data analytics relies on a data center consisting of large number of servers. Many internet services vendors have provide new parallel programming models, which provide users with simple programming interfaces, and hide the details of fault tolerance, parallelism and job scheduling. Such models mainly focus on how to define a powerful data and job abstraction which can present different types of big data applications and be effectively executed in a distributed environment. In general, jobs are split and assigned by a lower level scheduler. This system layer is a bridge between OS and data scientists, and provides a specific view of data to the upper level. Thus systems of this layer should consider problems about both resource management and data abstraction. For effective resource management, designers should consider problems such as how to use memory effectively during a computation, avoid unnecessary network overhead, and make good use of the high disk bandwidth to achieve high performance. For data abstraction, designers should consider the features of specific applications, and provide simple operators and good mechanism of resource optimization. Meanwhile, a high performance data streaming system can also be developed based on the thought of this “shared nothing” architecture. Systems on this layer can be summaries as the data management system, as most of the design principles and optimization policies are towards effective data management.

As the scale of data center keeps growing, the overhead of job scheduling becomes overwhelmingly significant. Many researches[21,22] from industry suggest to separate the scheduling duty from the data management systems. Moreover, via building a lower level scheduler, enterprises can run different data management systems on one data center to satisfy different analytics requirements. This kind of scheduler can be implemented in a data center system level, which is between the single OS and the data management system level. The challenge of concurrent jobs can be solved in this layer, by a smart scheduler.

The tail latency can become very serious as the scale of data center increases, and can significantly impact latency critical computing. Jeffery Dean[16] pointed out that many software level factors impacted this problem, involving resource sharing between servers, multiple layers queueing and system maintenance activities (periodic log compactions and garbage collection). Dean also proposed that “tail tolerant” should be considered as important as fault tolerant in data center.

The variability of response time can be reduced by techniques of different system levels. For example, scheduling requests in high level queues instead of using the operating system's queue is easy to perform service class based

priority policy, which allows systems to give high priority to interactive and latency critical requests. Moreover, systems should recognize resource requirement of different workload classes, thus to choose appropriate mechanism and computation pattern to guarantee high performance.

According to the discussion above, we propose that the software systems should be split to multiple levels and perform improvements independently, when building a computer system for latency-critical applications. The latency-critical system include data management system, data center system, and the operating system of single server. To evaluate the performance and latency of these systems, benchmarks of different kinds of financial applications are also essential. Consequently, the latency-critical computer system should be built based on a five-level architecture, which shows in Fig. 1. We will discuss recent researches of different levels in the next section.

4. Researches on tail latency

4.1. Benchmark

As the data scale increases, the communities of architecture, systems and data management pay more attention on developing new big data systems to satisfy requirements from different areas. The evaluation, measurement and comparison of these systems are important and really difficult problems. Meanwhile, the growing data volume makes applications more complex and diverse, which raises the difficulty of covering the diversity of real-world workloads for big data benchmark. In the domain of finance, a deep understanding of the features of analytics models is also very essential for system design and optimization.

We have developed a benchmark called BigDataBench[23] to solve such problems. It not only covers broad application scenarios, but also includes diverse and representative data sets. BigDataBench contains more than 20 big data benchmarks from dimensions of application scenarios, algorithms, data types, data sources, software stacks, and application types. These benchmarks include online services, offline analytics, and realtime analytics, with the data sets support of structured, semi-structured and unstructured data. Supported software stacks include Apache Hadoop, Spark, and MPI.

With the features mentioned above, BigDataBench has the ability to evaluate big data systems from different levels, including software stack, hardware, and algorithm. BigDataBench is also very helpful on tuning systems, such as using micro workloads to measure the hardware access behavior, thus to find the bottlenecks. How to generate huge volume of data is also considered in BigDataBench. The data generation tool provided by it can generate 10 TB test data sets on an 8-node cluster in about 5 h.

4.2. Hardware

A data center typically adopts a network fabric of two-level hierarchy. Low-end servers are mounted within several racks, while servers in each rack are interconnected via a local Ethernet switch.[24] These rack-level switches connect with one or more cluster-level Ethernet switches. The latency of network fabric can be considered as the time interval it takes when a packet traverse the network between the source to the destination. In general, a packet traverses 5–6 switches during a network communication,[25] which results in a in-switch delay. Moreover, packets may have to queue

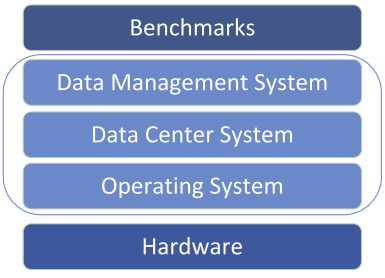

Fig. 1. The levels of a latency-critical computing system.

and wait for the NIC response when network congestion occurs, which causes a queueing delay, and leads to an increment of tail latency.

The straight way to reduce the in-switch latency is to increase the link speed. Many switches vendors such as Cisco have developed 10 Gigabit rack-level switches, and simplify the network fabric via devices consolidations.[26] Switches supporting 40 Gpbs and 100 Gbps link speed are also developed during past several years.

HULL (High-bandwidth Ultra-Low Latency)[27] is an architecture for low latency and high bandwidth utilization in data center fabric. The main idea of HULL is to trade plentiful bandwidth resource for the buffer space that is expensive to deploy. A 'bandwidth headroom' is reserved to cap the network utilization at less than link capacity. In this way, HULL leaves room for latency sensitive traffic, thus to promise low latency and avoid buffering. A new algorithm called DCTCP is also adopted in HULL for congestion control.

4.3. Operating system

The operating system plays a central role on improving resource utilization and guaranteeing quality of services, which promise high computing performance. To avoid the overheads brought by resource sharing, many researches have been done on how to design an effective model for better resource allocation and scalability. These approaches can be categorized into two types, which are respectively the centralized and decentralized OS models.

Most operating systems based on centralized model focus on how to reduce the resource sharing between processes in one shared-memory system. Tornado[28] and K42[29] try to optimize the data sharing via partitioning and replication, while other approaches, such as Corey,[30] give some privileges to the user applications and allow applications to control and reduce inter-core sharing.

Virtualization is one of the main implementations of decentralized OS models. The main idea of virtualization is to subdivide the resources of a single computer. This subdivision can be implemented in different levels ranging from hardware (e.g., Intel's VT), hardware abstraction layer(e.g., Xen[31]) to hosted VMs (e.g., KVM[32]). The Docker project[33] focuses on making user applications portable via separating them into different containers based on Linux container. This decentralized OS model can provide good resource isolation leading to effective resource sharing. Moreover, the additional hypervisor or other resource management layer can significantly increase the overhead of resource allocation and scheduling. To achieve lower latency, more works need to be done to optimize current OS models.

4.4. Data center system

Workloads of big data analytics need to run on a cluster full of computers. Researchers from google have pointed that these data centers are not simply a collection of co-located servers, but one massive warehouse-scale computer.[24] The main difference between such data centers and traditional clusters is that the servers in one WSC all belong to a single organization, and share a common systems management layer, while traditional data centers are shared by many different companies. Cluster scheduler is one of the most essential components of the system management layer, which controls the assignment of user tasks, priorities and resource management.

The requirement of low tail latency computation and large scale of data sets also pose a major challenge for data center scheduler. To achieve low tail latency, shorter duration of task and larger degree of parallelism are required. These two factors pose a big challenge of data center schedulers. This challenge requires the schedulers to not only make scheduling decisions at very high throughput, but also perform low latency scheduling and promise efficient system availability.[34] Current data center schedulers such as Mesos[21] and YARN,[22] adopt a centralized design, which schedules and assigns all tasks through a single node.

Some researchers tried to think this problem in another way. Sparrow[34] is a scalable, stateless and resilient decentralized task scheduler, which aims at highly parallel jobs with low latency requirement, and targets sub-second level task response time. Traditional decentralized architectures use a two choices scheduler, which two random servers are probed, and the scheduler picks the less loaded one according to performance sampling. Such design faces poor scheduling performance as the parallelism of jobs increasing, and the estimation of tasks' durations is very difficult.

Sparrow provides extended sampling approaches with two core techniques called *batch sampling* and *virtual reservations*. Batch sampling uses a multiple choices approach for parallel job scheduling, which places the m tasks in

a job on the least loaded of *dm* worker machines selected randomly (d > 1). Virtual reservations make the node monitors queue probe only when they are ready to run the task. This can remove task durations estimation, and avoid race conditions of multiple schedulers.

4.5. Big data system

The growing demand for large-scale data analysis applications presents a difficult challenge for both the industry and academic institutions. Traditional parallel database systems perform poorly as they are very expensive, and lack fault tolerance for long running workloads. Fault tolerance becomes a necessary issue as the scale of data center increasing to several thousand or even larger. Many new frameworks targeting big data workloads have been developed for big data workloads considering issues about data distribution, scheduling, and fault tolerance.

Google MapReduce[13] is developed to support offline batch processing on massive data sets. Data are stored in a distributed file system called Google File System (GFS) in a distributed way. MapReduce provides users with a simple interface to write distributed applications, and hide the details of fault tolerance, task scheduling, and inter-machine communication from developers. An MapReduce application contains multiple jobs, and each job is made up of two operations called Map and Reduce. Generally, map operations generate key value pairs according to user defined functions. All the data belonging to each key are sent to a same work node, and executed by a reduce function for each key in parallel. To achieve fault tolerance, all the intermediate data generated by map functions are written to the local disks, and result of each MapReduce job must be written back to the distributed file system.

MapReduce works well for offline long-running batch processing, such as log analysis and web crawl. However, its mechanism is not really suitable for latency critical workloads. Many other big data systems were developed based on some basic issues performed by MapReduce. Spark[14] is a distributed data management system developed by Berkeley AMPLab. It targets on reducing the latency of interactively iterative processing on big data. It provides fast data sharing across parallel jobs by caching data which need to be reused. The programming abstraction of Spark is called resilient distributed datasets (RDD). Different from MapReduce, RDD provides developers with many flexible operators, such as count, order and map. Each job is consist of a directed acyclic graph (DAG). Each DAG contains multiple operators on one RDD. In this way, all the intermediate data generated during the execution is kept in memory, if the data size is larger than the total memory, a block manager then moves some blocks to disks according to a cache algorithm. The jobs execute in a lazy schedule way, which means a job only begins to execute when a defined action operation is called, hence to leave enough space for execution optimization.

Big data systems also inspire researchers to build databases in a new way. Hive[35] is a scalable data warehouse built by Facebook to support massive data analysis. Hive adopts a SQL-like language called HQL for users to write queries. The engine is Apache Hadoop, an open source MapReduce implementation. A query will be translated into multiple MapReduce jobs, and submitted to Hadoop for execution. Hive combines the scalability of Hadoop and the sql parse techniques from database community, hence getting the ability to handle analysis processing on large scale data sets.

However, Hive is still unable to satisfy the low latency requirement for interactive query processing, due to the weakness of MapReduce framework. A Hive query request needs to be translated into multiple MapReduce jobs for execution. The intermediate data between each jobs are stored in the HDFS, which significantly increases the overhead of disk I/O. Shark[36] is a data analysis system towards low latency and fine grained fault tolerance. It is built on top of Spark, thus to benefit from the speed of in-memory computing. Similar to Hive, Shark also implements a SQL parser on a scalable engine. To achieve lower latency, Shark also support the features of caching tables in memory, dynamic query plan optimization and memory-based shuffle. Moreover, a SQL query can be executed as a series of operations on one RDD in Spark, so the intermediate data can be kept in memory during the execution. These features and the advances of Spark engine make Shark perform great on data analysis. Shark can run about 100× faster than Hive.

Even though the fault tolerance feature has become one of the recognized elements that should be considered for big data system design, some systems choose to remove it for extreme performance. Impala[15] is a massive parallel processing system inspired by Dremel.[37] The main purpose of Impala is to process real time analysis processing of large data sets. Impala adopts a coarser-grained recovery model, which means the query is resubmitted if it fails during execution. For data store, however, Impala distributes data on the Hadoop distributed file system (HDFS), or HBase, which provides node level fault tolerance. These choices make Impala be still able to work on mass data, but can not fit for long running workloads which always keep running for days or months. However, such considerations can make

Impala achieve great performance for short queries and real time analysis, of which users may want to get the result immediately. Unlike MapReduce and Spark, Impala does not transfer a query into a job that is represented by operations of one underlying engine, but converts the query to a serving tree-like structure which consists of different plan fragments. When executing, these plan fragments of the tree will be mapped to the physical nodes in the cluster, and the leaf nodes are first executed, which are responsible for table scanning. After the scanning is done, leaf nodes send the partial results to the nodes of next level, which perform more complex operations such as aggregation and join. In this way, original data are processed and passed from low nodes up to the top, until the top node completes its final computation and return the result to the user.

To handle the real-time big data stream processing problem, many organizations have developed their data streaming frameworks. Storm[38] is an open source data stream processing framework, which runs in-memory and is able to process large scale of data with in-memory speed. Storm involves the coordinator, state manager, and process nodes. The data flow model implemented in Storm is called stream, which is an unbounded sequence of tuples, and a tuple is the basic data structure representing standard data types or user-defined types. The data sources are called the spouts, while the data sinks are called bolts. Each bolt implements a single transformation on a stream. Blots can be used to implement operations like aggregation, filtering and MapReduce operators. A stream is used to build the data processing topologies between spouts and bolts. Storm needs a data source for input and a data store for output. A memory-based data store, such as HBase,[39] can help storm to achieve great processing speed.

5. Future work and conclusion

The demand of low latency analytics on huge amount of data is rapidly increasing, and faster response time means more users and revenues. This requirement poses a great challenge for both hardware and software stack design. In this paper, we point out the challenges of latency critical big data computing in financial, and propose that the latency demand should be considered in five system levels from the benchmark design down to the hardware. We also present some examples in each level for low latency, which includes the BigDataBench for evaluating big data systems, different types of big data management systems for latency tolerant, the data center operating system designed for tail latency mitigating, and different operating system models.

In the future, we plan to research on the common features of current big data analytics, and improve the BigDataBench. We also try to develop a new high performance data management system, which supports different types of big data analytics and uses memory and other hardware resources effectively to provide great computing speed. Finally, We hope the discussions in this paper can be helpful for future latency-critical system design and evaluation.

References

1. Card SK, Robertson GG, Mackinlay JD. The information visualizer, an information workspace. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. CHI'91*. New York, NY, USA: ACM; 1991:181−186.
2. Fawcett J. *The New Finance Era: From Hft to Big Data*; 2013. http://www.wallstreetandtech.com/electronic-trading/the-new-finance-era-from-hft-to-big-data/a/d-id/1268323?.
3. SAP. *Big Data and Smart Trading: How a Real-time Data Platform Maximizes Trading Opportunities*; 2012. http://wallstreetandtech.com/whitepaper/Data-Latency/Trading-Infrastructure-Technology/big-data-and-smart-trading-wp1348004765?articleID=191705725.
4. Jewell D, Barros RD, Diederichs S, et al. *Performance and Capacity Implications for Big Data*; 2014. http://www.redbooks.ibm.com/Redbooks.nsf/RedbookAbstracts/redp5070.html?OpenDocument.
5. Paul G-J. *How the Financial Services Sector Uses Big Data Analytics to Predict Client Behaviour*; 2011. http://www.computerweekly.com/feature/How-the-financial-services-sector-uses-big-data-analytics-to-predict-client-behaviour.
6. Zyl G. *Sas Finance Sector Banking More on Big Data*; 2014. http://www.fin24.com/Tech/Companies/SAs-finance-sector-banking-more-on-big-data-20141207.
7. Michael V, Karen M. *The Case for Big Data in the Financial Services Industry*. 2012.
8. C B, A M, G J. Data stream management systems for computational finance. In: *Computer*. 2010:45−52.
9. David T, Michael S, Rebecca S. *Analytics: The Real-world Use of Big Data in Financial Services*; 2013. https://www14.software.ibm.com/webapp/iwm/web/signup.do?source=swg-NA_ISDP_Standard-Reg&S_PKG=ov14713.
10. Cutting D, Cafarella M. *Hadoop*; 2005. http://hadoop.apache.org/core/.
11. Chang F, Dean J, Ghemawat S, et al. Bigtable: a distributed storage system for structured data. In: *Proceedings of the 7th Symposium on Operating Systems Design and Implementation. OSDI'06*. Berkeley, CA, USA: USENIX Association; 2006:205−218. http://dl.acm.org/citation.cfm?id=1298455.1298475.

12. DeCandia G, Hastorun D, Jampani M, et al. Dynamo: Amazon's highly available key-value store. *SIGOPS Oper Syst Rev*. Oct. 2007;41:205−220.
13. Dean J, Ghemawat S. Mapreduce: simplified data processing on large clusters. In: *Proceedings of the 6th Conference on Symposium on Opearting Systems Design & Implementation − Volume 6. OSDI'04*. Berkeley, CA, USA: USENIX Association; 2004, 10−10 http://dl.acm.org/citation.cfm?id=1251254.1251264.
14. Zaharia M, Chowdhury M, Das T, et al. Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing. In: *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation. NSDI'12*. Berkeley, CA, USA: USENIX Association; 2012, 2−2.
15. Cloudera. *Impala*; 2012. http://www.cloudera.com/content/cloudera/en/products-and-services/cdh/impala.html.
16. Dean J, Barroso LA. The tail at scale. *Commun ACM*. Feb. 2013;56:74−80.
17. Dongarra J, Bunch J, Moler C, Stewart G. *Linpack*; 1970. http://www.netlib.org/linpack/.
18. Gupta S. *Nvidia and Ibm Bring Supercomputing to Big Data Analytics*; 2014. http://blogs.nvidia.com/blog/2014/10/06/nvidia-ibm-big-data-analytics/.
19. PR I. *Intel Brings Supercomputing Horsepower to Big Data Analytics*; 2013. http://newsroom.intel.com/community/intel_newsroom/blog/2013/11/19/intel-brings-supercomputing-horsepower-to-big-data-analytics.
20. IBTA. *Infiniband*; 1999. www.infinibandta.org.
21. Hindman B, Konwinski A, Zaharia M, et al. Mesos: a platform for fine-grained resource sharing in the data center. In: *Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation. NSDI'11*. Berkeley, CA, USA: USENIX Association; 2011:295−308.
22. Vavilapalli VK, Murthy AC, Douglas C, et al. Apache hadoop yarn: yet another resource negotiator. In: *Proceedings of the 4th Annual Symposium on Cloud Computing. SOCC'13*. New York, NY, USA: ACM; 2013:5, 1−5:16.
23. Wang L, Zhan J, Luo C, et al. Bigdatabench: a big data benchmark suite from internet services. In: *High Performance Computer Architecture (HPCA), 2014 IEEE 20th International Symposium on*. Feb 2014:488−499.
24. Hoelzle U, Barroso LA. *The Datacenter as a Computer: An Introduction to the Design of Warehouse-scale Machines*. 1st ed. Morgan and Claypool Publishers; 2009.
25. Al-Fares M, Loukissas A, Vahdat A. A scalable, commodity data center network architecture. In: *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication. SIGCOMM'08*. New York, NY, USA: ACM; 2008:63−74.
26. Cisco. *Cisco nexus 5000 Series Switches: Decrease Data Center Costs with Consolidated I/O*; 2006. http://www.cisco.com/c/en/us/products/collateral/switches/nexus-5020-switch/white_paper_c11-468838.html.
27. Alizadeh M, Kabbani A, Edsall T, Prabhakar B, Vahdat A, Yasuda M. Less is more: trading a little bandwidth for ultra-low latency in the data center. In: *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation. NSDI'12*. Berkeley, CA, USA: USENIX Association; 2012.
28. Gamsa B, Krieger O, Appavoo J, Stumm M. Tornado: maximizing locality and concurrency in a shared memory multiprocessor operating system. In: *Proceedings of the Third Symposium on Operating Systems Design and Implementation. OSDI'99*. Berkeley, CA, USA: USENIX Association; 1999.
29. Krieger O, Auslander M, Rosenburg B, et al. K42: building a complete operating system. *SIGOPS Oper Syst Rev*. Apr. 2006;40.
30. Boyd-Wickizer S, Chen H, Chen R, et al. Corey: an operating system for many cores. In: *Proceedings of the 8th USENIX Conference on Operating Systems Design and Implementation. OSDI'08*. Berkeley, CA, USA: USENIX Association; 2008.
31. Barham P, Dragovic B, Fraser K, et al. Xen and the art of virtualization. In: *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles. SOSP '03*. New York, NY, USA: ACM; 2003:164−177.
32. Soltesz S, Pötzl H, Fiuczynski ME, Bavier A, Peterson L. Container-based operating system virtualization: a scalable, high-performance alternative to hypervisors. *SIGOPS Oper Syst Rev*. Mar. 2007;41.
33. Merkel D. Docker: lightweight linux containers for consistent development and deployment. *Linux J*. Mar. 2014;2014.
34. Ousterhout K, Wendell P, Zaharia M, Stoica I. Sparrow: distributed, low latency scheduling. In: *Proceedings of the Twenty-fourth ACM Symposium on Operating Systems Principles. SOSP'13*. New York, NY, USA: ACM; 2013:69−84.
35. Thusoo A, Sarma JS, Jain N, et al. Hive − a petabyte scale data warehouse using hadoop. In: *ICDE*. 2010.
36. Xin RS, Rosen J, Zaharia M, Franklin MJ, Shenker S, Stoica I. Shark: sql and rich analytics at scale. In: *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data. SIGMOD'13*. New York, NY, USA: ACM; 2013:13−24.
37. Melnik S, Gubarev A, Long JJ, et al. Dremel: interactive analysis of web-scale datasets. *Commun ACM*. Jun. 2011;54:114−123. http://dx.doi.org/10.1145/1953122.1953148.
38. Toshniwal A, Taneja S, Shukla A, et al. Storm@twitter. In: *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data. SIGMOD'14*. New York, NY, USA: ACM; 2014:147−156.
39. Borthakur D, Gray J, Sarma JS, et al. Apache hadoop goes realtime at facebook. In: *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data. SIGMOD'11. New York, NY, USA*. 2011.