INTERNATIONAL JOURNAL OF RESEARCH IN COMPUTER APPLICATIONS AND ROBOTICS
ISSN 2320-7345

BIG DATA SOURCE ANALYSIS DATA SERVICES IN GEO-DISTRIBUTED DATA CENTERS

[1]S. Paul Steven, [2]X. Arogya Presskila, [3]Dr.K.Ramesh

[1]S. Paul Steven, PG Scholar, E-mail: paulsteven92@gmail.com
[2]X. Arogya Presskila, Assistant Professor, E-mail: presskila@gmail.com
[3]Dr.K.Ramesh, Assistant Professor, E-mail: rameshk7n@yahoo.co.in

Abstract: - With the launch of the database systems, several companies have been using it over the last few decades that were mainly driven by improvement in hardware, availability of an oversized quantity of data, collection of data at an infinite rate, rise in applications and so on. The sketch of data management systems is gently fragmented based on application domains (i.e., applications depending on relational data, graph-based data, and stream data). State-of-the-art commercial and educational systems use disk-related and in-memory operations. In this paper, an in-memory system is aimed for storing huge amount of data and serving faster accesses on database systems. In business operations, quickness is not an option, however a required one. Hence, each pathway in database systems is exploited to further improve performance, including decreasing dependency on the hard disk, increasing lots of memory to make a lot of data reside within the memory, and even setting up an in-memory system where large quantity of data's will be hold on within the memory. Most commercial database vendors have recently launched in-memory database processing systems. Efficient in-memory data management may be a necessity for various applications. Nevertheless, in-memory data management remains at its early stage, and is probably going to evolve over the next few years.

Keywords: Big Data, In-Memory System, In-memory Database, Efficiency, Memory

1. Introduction

Big data give rise to several researches to develop systems for helping ultra-low latency service and real-time data analytics. Current disk-based systems can no longer offer timely response due to the high access latency to hard disks. The unsatisfactory performance was initially detected by internet companies such as Amazon, Google, Facebook and Twitter, but now it is becoming an hurdle for other companies/organizations that desire to provide a meaningful service (e.g., advertising, social gaming). For example, trading companies demand to detect a quick change in the trading prices and react immediately (in several milliseconds), which is no way to achieve using traditional disk-based processing/storage systems [1]. To fit the strict real-time requirements for evaluating mass amount of data and fulfilling requests within milliseconds, an in-memory system/ database that carries the data in the random access memory (RAM) all the time is necessary [3]

In the last decade, multi-core processors and the vacancy of massive amounts of main memory at plummeting cost are creating new breakthroughs, making it feasible to build in-memory systems which is a significant part, if not the entirety, of the database fits in memory. For instance, memory storage size and

bandwidth have been dualize roughly every three years, while its price has been slashed by a factor of 10 every five years. Likewise, there have been significant advances in non-volatile memory (NVM) like SSD and the impending introduction of various NVMs that is phase change memory (PCM). The amount of I/O operations per second in such devices is far higher than hard disks. Modernized high-end servers sometimes have multiple sockets, each of which might have quite a whole lot of gigabytes of DRAM, and tens of cores, and in overall, a server may have a lot of terabytes of DRAM and hundreds of cores. Moreover, in a distributed environment, it is accessible to aggregate the memories from a cluster of server nodes to the extent that the aggregated memory is able to keep up all the data for a different variety of large-scale applications [16].

2. Problem Statement

In the existing database systems, the large amount of data cannot be kept in the system. If the large amount of data is stored, then the system becomes slow and it hangs. Also, it increases the cost of the memory. In order to access large amount of data, the multi-core processor and large amount of memory is needed. To overcome these problems, an in-memory database is introduced, which can manage large amount of data.

3. Proposed Technique

The in-memory database helps us to store large datasets in order to provide faster accesses and real-time analytics. With the introduction of NV-RAM technology, in-memory databases will be able to run at full speed and maintain data in the event of power failure Extremely fast compared to disk access. Scale-up parallelism can take advantage of the multi-core architecture of super computers or even commodity computers.

Advantages

- Deploying trusted hardware for data processing,
- A trusted hardware based database with full data confidentiality

4. Related Work

There have been many proposals in the area of privacy preserving data mining recently. It proposed an algorithm to securely compute the weighted average problem (WAP) for the 2-party privacy-preserving K-means algorithm extended WAP to multi-party case. In Mert's method, if some parties collude, they can easily deduce the private information of other parties. It proposed another solution of computation of the ratio of (two) summations of data spreading over *m*parties to solve the issue of privacy-preserving naive Bayes. The result can be securely computed without revealing any private information if there is no collusion. Unfortunately, if some of parties collude, privacy may be revealed [15].

Column Storage is Best Suited for Modern CPU's

Modernized CPUs with multi-core architecture provide an gigantic amount of computing power. Blades with 8 CPUs and 16 cores per CPU will extend next-generation blade servers. That deliver us 128 computing units with up to approximately 500 GB of main memory. To enhance the use of these computing devices we have to know memory hierarchies, cache sizes, and how to set up parallel processing within one program. The memory situation is considered. Enterprise applications are to a huge extent memory bound, that shows the program execution time is proportional to the amount of memory used for read and write operations or memory being moved [8].

Entire calculations on the tuple level will regularly be parallelized, since they are fully independent of each other. The beginning of an aggregation will be executed synchronously for every quailed tuple. The synchronization between the core processes is littlest. Further aggregation along given hierarchies obtain as a second step on the accumulated data. The same assigned to sorting by attributes or sequencing by time.

Database Locks

With the insert-only approach the renew of tuples by the application could be disposed with the exception of binary status variables. Having more versions of the same tuple in the database needs that the older ones be marked as invalid. Each inserted tuple has the timestamp of its creation and in case it is frequently updated, the timestamp of the update [8] . Only the recent version of a tuple carries no update timestamp and so, it is easily identifiable. The benefit of this concept is any state of the tuple can be remodeled by using the 2 timestamps with regards to a base date for the query. This approach has been adopted before in POSTGRES in 1987 and was referred to as time-travel [8]. The extended SQL has to support a base date parameter through which the accurate version of a tuple can be identified. To load all older versions of a tuple in the same table has definite application benefits particularly in coming up with applications, where retrieving older versions of data is common. In addition to that it fully eliminates the requirements of making a separate log of the changes. The increasing storage capacity requirements will be unheeded. The timestamp attributes are don't seem to be taking part in any compression algorithm and doesn't lead to any reorganization of the column when updated. Since multiple queries can collide with inserts and updates, high level care has to be taken to avoid an excessive amount of lockup on table, column or dictionary level [15].

5. Experiments

Coprocessor as a System Server:

In this sequence, we used to supplement the functions of the primary processor (Server).Operations performed by the coprocessor are decryption or I/O Interfacing with peripheral devices. RSA Algorithm is used for decrypting the System Data. The decrypted data is loaded into the server physical setting to transform the System in to Server& Server Memory utility is configured. In Server Memory Utility, the system memory is spitted in to two ones. The Big Data process run is based on one of the memory. If first one fails, second one will be used to process the data. The Threading technique is used to split the Processors. Finally the System is transformed in to a Server[15]. This module absolutely used for decrypting system information, configuring Server and communicating with Admin

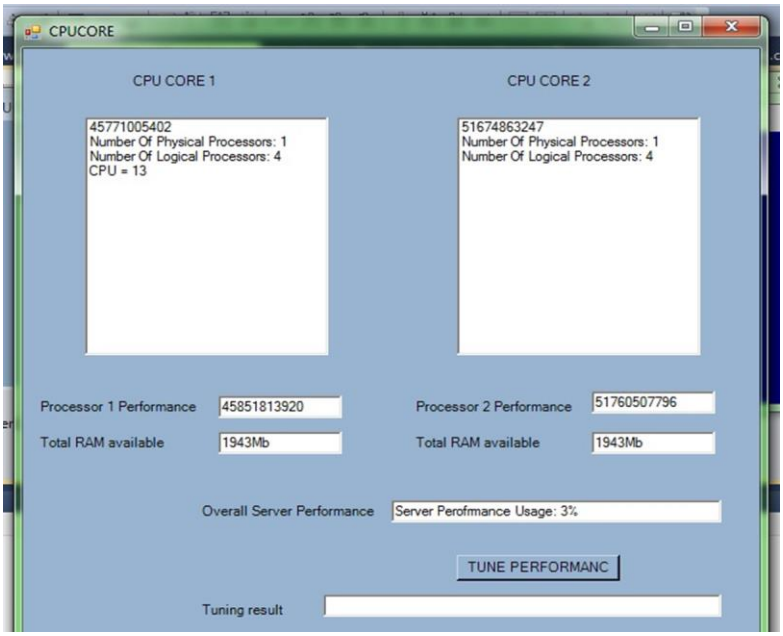

Fig 1. Spitting CPU Memory

SQL & Hadoop Process

Database is to be secure because hackers can hack user information. We are using encryption and decryption both for secure purpose. This security purpose is available in Hadoop. Here I am using Stock Market Datasets as an example for storing large Data Sets into PC.Several Companies; Stock Market Data's are provided. User has an option to select their need. After selecting the user need, it can be imported to PC.In this SQL Database, the large amount of datasets stored in Hadoop[2],[6], can be imported to SQL and compressed to a specific size and then it is provided to the user. There is no loss of information. Ordinarily SQL will store up to 5 GB of Data. After linking with Hadoop [2], [4], it can store large amount of Data

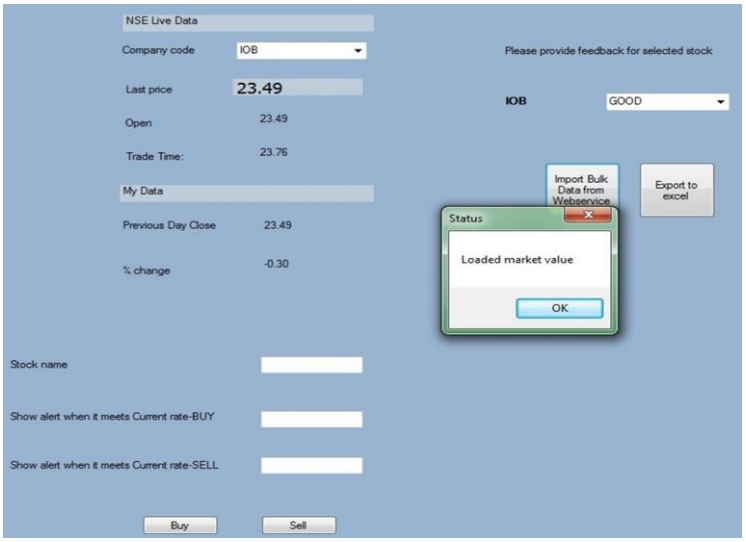

Fig 2. Transferring Big Data from Web to System

Roc Process & Performance Monitor:

The Remote Operations Controllers from the ROC family has established an business benchmark for flexibility, robustness, ease of use and reliability The bugs occurred during execution can be cleared by ROC process. After the removal of bugs, the performance of System can be measured. It is ideal for the monitoring, measurement and control of processes [11] it also compares the performance of the system before and after the removal of bug and the result is obtained.

Task Execution:

There is a delay in the execution of background tasks if you run to import operation. This delay is due to the import of large amount of data to the system. These large amount of datasets are gathered in Hadoop [2], [4], [6] and stored in SQL and finally it is compressed to a specific size. Then it is available to the user. Task execution time is nearly constant, with minor deviations due to cache, memory access and interrupts latency.

6. Conclusion

The shifting to big data triggers a rethinking of the planning of ancient systems, particularly for databases, within the side of data structures, indexes, parallelism, concurrency management, query process, fault-tolerance, etc. Modern CPU usability's and memory-hierarchy-conscious optimization play a major role within the design of in-memory systems, and advanced hardware technologies like HTM and RDMA offer a promising chance to resolve issues encountered by software package solutions in in-memory data management and processing. Additionally, we also discussed some pioneering in-memory New SQL and No SQL databases as well as cache systems, batch and online/continuous process systems. We tend to highlighted some favorable design techniques

thoroughly, from that we will learn the practical and concrete system design principles. This paper contributes a comprehensive review of vital technology in memory management and analysis of connected works thus far, that are a helpful resource for more memory-oriented system analysis.

Acknowledgement

This was supported by part of ***The 27th Annual Conference of the Japanese Society for Artificial Intelligence, 2013*** **3L3-OS-06b-1**

REFERENCES:

[1] In-Memory Big Data Management and Processing: A Survey Hao Zhang, Gang Chen, Member, IEEE, Beng Chin Ooi, Fellow, IEEE,Kian-Lee Tan, Member, IEEE, and Meihui Zhang, Member, IEEE,2015
[2] K. Shvachko, H. Kuang, S. Radia, and R. Chansler,"The Hadoop Distributed File System",2010
[3] J. Ousterhout, P. Agrawal, D. Erickson, C. Kozyrakis, J. Leverich, D. Mazi_eres, S. Mitra, A. Narayanan, G. Parulkar, M. Rosenblum, S. M. Rumble, E. Stratmann, and R. Stutsman, "The case for RAMClouds: Scalable high-performance storage entirely in dram," ACM SIGOPS Operating Syst. Rev., vol. 43, pp. 92–105, 2010.
[4] F. Li, B. C. Ooi, M. T. € Ozsu, and S. Wu, "Distributed data management using MapReduce," ACM Comput. Surv., vol. 46, pp. 31:1–31:42, 2014.
[5] V. Borkar, M. Carey, R. Grover, N. Onose, and R. Vernica, "Hyracks: A Flexible and Extensible Foundation for Data-Intensive Computing",2011
[6] HadaptInc.. (2011). Hadapt: Sql on hadoop [Online]. Available: http://hadapt.com/
[7] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, and R. Murthy, "Hive: A warehousing solution over a map-reduce framework," in Proc. VLDB Endowment, vol. 2, pp. 1626–1629, 2009.
[8] HassoPlattner," A Common Database Approach for OLTP and OLAP Usingan In-Memory Column Database,2009
[9] J. C. Corbett, J. Dean, M. Epstein, A. Fikes, C. Frost, J. J. Furman, S. Ghemawat, A. Gubarev, C. Heiser, P. Hochschild, W. Hsieh, S. Kanthak, E. Kogan, H. Li, A. Lloyd, S. Melnik, D. Mwaura, D. Nagle, S. Quinlan, R. Rao, L. Rolig, Y. Saito, M. Szymaniak, C. Taylor, R. Wang, and D. Woodford, "Spanner: Google's globallydistributed database," in Proc. USENIX Symp. Operating Syst.Des. Implementation, 2012, pp. 251–264.
[10] S. Alsubaiee, Y. Altowim, H. Altwaijry, A. Behm, V. R. Borkar, Y. Bu, M. J. Carey, I. Cetindil, M. Cheelangi, K. Faraaz, E. Gabrielova, R. Grover, Z. Heilbron, Y. Kim, C. Li, G. Li, J. M. Ok, N. Onose, P. Pirzadeh, V. J. Tsotras, R. Vernica, J. Wen, and T. Westmann, "Asterixdb: A scalable, open source BDMS," in Proc. Very Large Database, pp. 1905–1916, 2014.
[11] MySQL AB. (1995). Mysql: The world's most popular open source database [Online]. Available: http://www.mysql.com/
[12] Apache. (2008). Apache cassandra [Online]. Available: http:// cassandra.apache.org/
[13] Oracle. (2013). Oracle database 12c [Online]. Available: https://www.oracle.com/database/index.html
[14] HP. (2011). Vertica systems [Online]. Available: http://www. vertica.com
[15] H. T. Vo, S. Wang, D. Agrawal, G. Chen, and B. C. Ooi,"LogBase: A Scalable Log structured Database System in the Cloud",2012

A Brief Author Biography

1. **S. Paul Steven** - PG Scholar, Department of Computer Science, SCAD College of Engineering and Technology, Tirunelveli, Tamil Nadu, India
2**. Ms. X. Arogya Presskila** - Assistant Professor, Department of Computer Science, SCAD College of Engineering and Technology, Tirunelveli, Tamil Nadu, India
3. **Dr. K.Ramesh** - Assistant Professor Department of Computer Applications, Regional Office of Anna University, Tirunelveli, Tamil Nadu, India