ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET) Vol. 5, Special Issue 12, April 2018

MULTICORE PROCESSING IN THE CLOUD MEMORY DATA USING SINGLE TRANSACTION(VM)

P.Anbuselvi
Research scholar, Department of Computer Science, Govt. Arts College, Ariyalur, Tamilnadu, India.

P.Selvakumar
Research Supervisor, Asst. Prof. of Computer Science, Govt. Arts College, Ariyalur, Tamilnadu, India.

ABSTRACT
In the concept of data facts running, Data is a broad term for figures set as a result great or compound that usual data processing applications be insufficient. Challenge take in study, incarcerate, hunt, division, cargo space, transport, apparition, and in arrange privacy. The term often refers simply to the use of predictive analytics or other certain advanced methods to extract value from records, and not often to a fussy amount of information set. The main idea of this projected occupation be spread storage space arrangement and Cache remembrance. During the data meting out, its check the size of the file whether if its inside the extent wealth its stock up on the folder if its exceeds means its stored on the collection. During user handing out the data its retrieves the data since the together recollection administration user can download the documents by means of near mechanism

Keywords: Transaction management, concurrency control, single-threaded model, multi-core, in-memory OLTP system

INTRODUCTION
Traditional concurrency control protocols can be categorized into lock-based protocols and timestamp based protocols. For lock-based protocols, lock thrashing and deadlock avoidance are the main challenges. For timestamp-based protocols, the main issues are the high abort rate and the need for a well-coordinated timestamp allocation. Recently, the single threaded model is widely used to improve the efficiency of in-memory database systems. In this model, each worker thread is uniquely assigned to a static partition of the database. When a transaction arrives, a employee that is accountable for its dispensation will get all the tresses of the partition to be accessed. By doing so, all the critical sections that are contended for can be resolved in advance. In a situation where the majority transactions only require contacting data in a single partition, the threads can work independently, achieving a high degree of parallelism. However, in practice, no matter how well the database has been partitioned, transactions that span multiple partitions cannot be totally avoided. Such transactions typically cause a substantial amount of blocking as threads tend to contend with each other to acquire all the necessary locks, resulting in more CPU idle time. Furthermore, continuously changing workloads may cause load imbalance where some worker threads experience significant blocking while others have too much work to handle. For a statically partitioned database where each divider is assign to a strand, a slanted workload could lead to twisted CPU utilization. Consequently, it significantly reduces the efficiency of the entire system.

As can be seen from the result, the percentage of CPU idle time of H-Store increases with the increment of cross-partition transactions. When there is no cross partition transaction in the workload, each worker thread can process transactions independently without any blocking. When the workload contains cross-partition transactions, possible disputation among worker threads appears with its prospect increasing along the increase of cross-partition transactions. To resolve contention, thread blocking is usually inevitable, which degrades the CPU utilization.

The data structure in the lock manger is typically very large and complex, which incurs both storage and processing overheads. Light-weight Intent Lock (LIL) maintains a set of lightweight counters in a global lock table instead of lock queues for intent locks. Although LIL simplifies the data structure of intent locks, transactions that cannot obtain all the locks have to be

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET) Vol. 5, Special Issue 12, April 2018

blocked until a release message from the other transactions is received. In order to reduce the overhead of a global lock manager, associating the lock states with each data record has been proposed. However, this technique requires each record to maintain a lock queue, and hence increases the burden of record management. By compressing all the lock states at one record into a pair of integers, Very Lightweight Locking simplifies the data structure to some extent. However, it achieves this by dividing the database into disjoint partitions, which affects the performance on workloads that cannot be well partitioned.

RELATED WORKS

In [1] Raymond H. Y. Louie, Yonghui Li, and Branka Vucetic et al present the presentation of sensible physical-layer network coding (PNC) scheme for joint relay channels. We first believe a network consisting of two source nodes and a single relay node, which is used to aid communication between the two source nodes. For these circumstances, we investigate transmission over two, three or four time slots. We show that the two time slot PNC scheme offers a higher maximum sum rate, but a lower sum-bit error rate (BER) than the four time slot transmission technique for a number of practical scenarios. We also demonstrate that the three time slot PNC arrangement offers a good cooperation between the two and four time slot broadcast schemes, and also attain the best maximum sum-rate and/or sum-BER in certain practical scenarios. To facilitate comparison, we gain new closed-form expressions for the outage probability, greatest sum-rate and sum-BER.

In [2] Hao Zhang, Gang Chen, Kian-Lee Tan, Beng Chin Ooi et al presents Growing major memory capacity has fueled the enlargement of in-memory big data organization and dispensation. By eliminate disk I/O bottleneck, it is now probable to sustain interactive data analytics. However, in-memory systems are much more disposed to other sources of overhead that do not matter in traditional I/O-bounded disk-based systems. Some issues such as fault-tolerance and consistency are also more demanding to handle in in-memory environment. We are witnessing a revolution in the design of database systems that exploits major memory as its data storage layer. Many of these researches have focused along several dimensions: modern CPU and memory hierarchy utilization, time/space efficiency, parallelism, and concurrency organize. In this review, we aim to provide a thorough review of a wide variety of in-memory data management and dispensation proposals and systems, counting both data storage systems and data processing frameworks.

In [3] Alfons Kemper, Thomas Neumann et al presents The two areas of online transaction processing (OLTP) and online analytical processing (OLAP) present dissimilar dare for verification architectures. Currently, customers with high rates of mission-critical transactions have opening their data into two divide systems, one database for OLTP and one so-called data warehouse for OLAP. While permit for truthful deal rates, this severance has numerous disadvantages counting data freshness issues due to the delay cause by only occasionally initiating the Extract Transform Load-data staging and extreme resource consumption due to maintaining two separate information systems. We present a capable hybrid format, called Hyper that can button both OLTP and OLAP concomitantly by using hardware-assisted duplication tackle to prolong reliable snapshots of the transactional data.

In [4] Kun Ren, Alexander Thomson, Daniel J. Abadi et al presents Locking is generally worn as a concurrency control mechanism in database systems. As more OLTP databases are stored typically or completely in memory, transactional throughput is less and fewer incomplete by disk IO, and lock managers increasingly become performance bottlenecks. In this paper, we introduce very insubstantial locking (VLL), an substitute approach to distrustful concurrency control for main-memory database systems that avoids almost all transparency associated with traditional lock manager operations. We also proposition a procedure called discerning contention analysis (SCA), which enable systems implement VLL to accomplish prominent transactional throughput underneath high contention workloads. We implement these protocols both in a traditional single-machine multi-core database server setting and in a distributed database where data is partitioned across numerous product machines in a shared-nothing cluster.

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

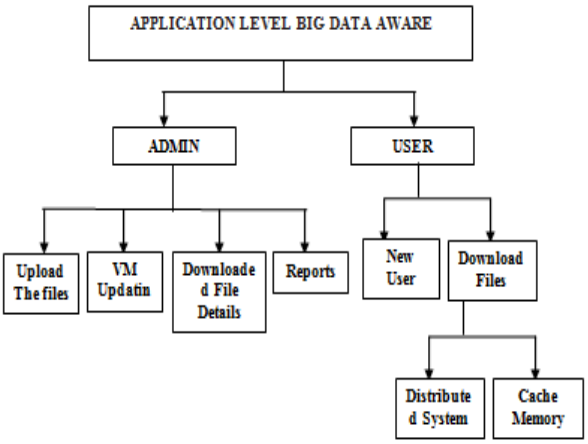
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET) Vol. 5, Special Issue 12, April 2018

In [5] Thomas Neumann, Tobias Mühlbauer, Alfons Kemper et al presents Multi-Version Concurrency organize (MVCC) is a lengthily working concurrency control mechanism, as it allow for execution modes where reader never block writers. However, most systems execute only snapshot isolation (SI) instead of full serializability. Adding serializability guarantees to presented SI implementations tends to be prohibitively costly. We present a narrative MVCC accomplishment for main-memory database systems that has incredibly little transparency contrast to sequential execution with single-version concurrency control, even when maintaining serializability guarantees. Updating data in-place and store versions as before-image deltas in undo buffers not only allow us to retain the high scan performance of single-version systems but also form the basis of our despicable and fine-grained serializability validation mechanism. The novel idea is based on an adaptation of precision lock and verifies that the (extensional) write of recently committed transactions do not intersect with the (intensional) read predicate space of a commit transaction.

PROPOSED SYSTEM

In this paper, inspect the design of multi-core in-memory OLTP systems with the objective of improving the throughput of transaction dispensation through better fitting of the single threaded model onto modern multi-core hardware. In exacting, we propose LADS, a dynamic single-threaded OLTP arrangement, which separates the concurrency control from the actual transaction execution. LADS opening determine a lot of dealings into a position of confirmation actions. Record action is a consecutive sequence of (atomic) operations on the same tuple within a transaction. LADS make use of dependency graphs to imprison dependence relations among evidence actions within a batch of transactions. It then decomposes the dependency graphs into sub-graphs such that the sub graphs are about the same size, and the numeral of limits across these sub-graphs is minimized.

ARCHITECTURE DIAGRAM



MODULES

I) ADMIN

A) UPLOAD THE FILES

In the Admin process it can be categorized into two ways are, (1) Distributed System (2) Cache Memory. During upload the files it checks the size of the files based on the file size Admin upload the files. If the size is valid, Admin upload the files in the distributed system, if the size varies means Admin stored on the cache memory.

B) VM UPDATION

Virtual memory is a feature of an operating system (OS) that allows a computer to compensate for shortages of physical memory by temporarily transferring pages of data from random access memory (RAM) to disk storage. Here, VM Updating can be viewed by Admin.

C) FILE DOWNLOADING ACCURACY

The Admin can View the downloaded file details.

D) REPORTS

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET) Vol. 5, Special Issue 12, April 2018

The Report Module Consist uploaded files report, Virtual Memory Updating report, New User Registration Report and Downloaded Files Report.

II) USER

A) NEW USER REGISTRATION

This module consist the registration of new user. New user registration can use for user login process.

B) DOWNLOAD THE FILES

i) SEARCH A FILE

Based on User requirements the admin can upload the files the user can search the files from the admin.

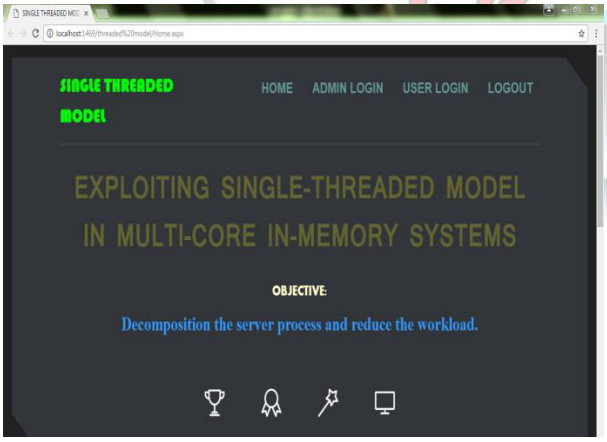
ii) DOWNLOAD

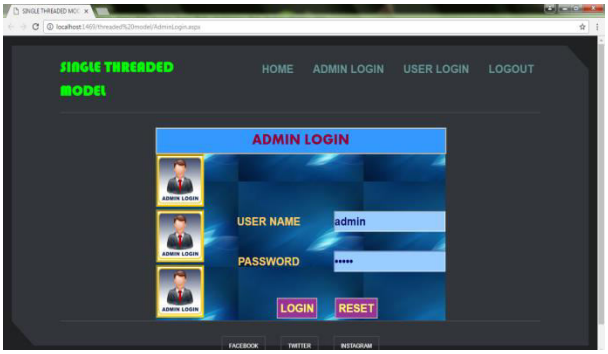
The User can download the files which can be stored in the distributed system by using Virtual Memory. It can be partitioned VM1, VM2, and VM3. User can download the files via Virtual Memory. The Virtual Memory can be updated by the file Size.

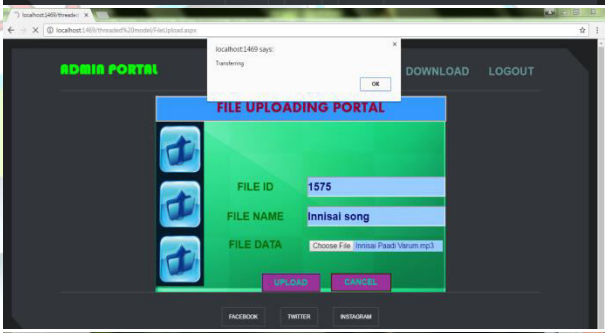
iii) CACHE FILES DOWNLOAD

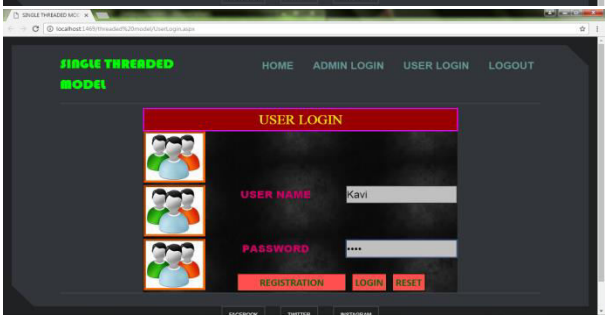
The files which are uploaded into Cache Memory it also downloaded to User.

OUTPUT RESULT



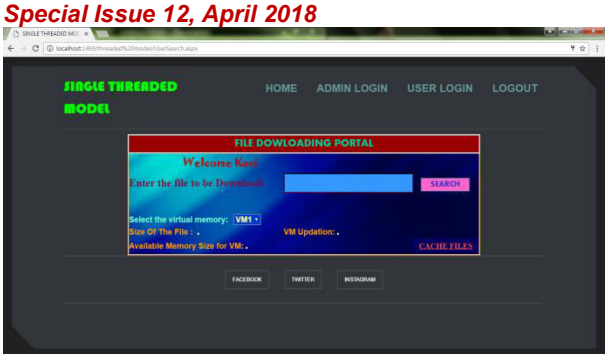


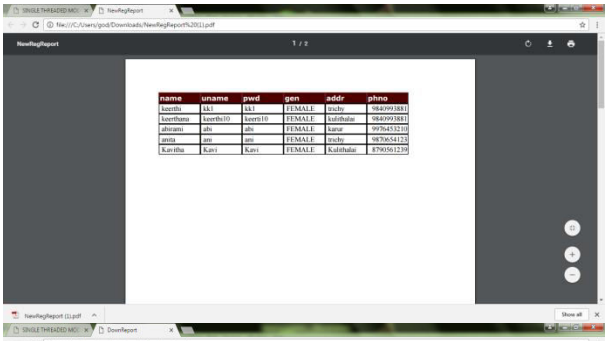


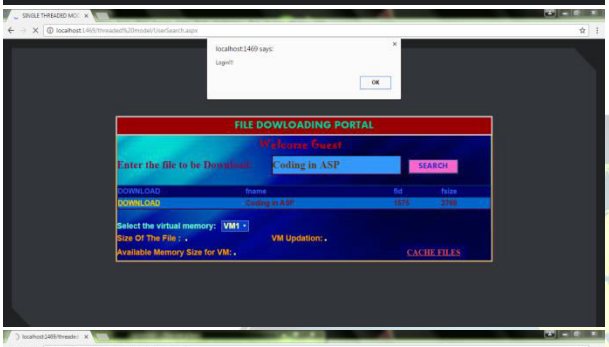


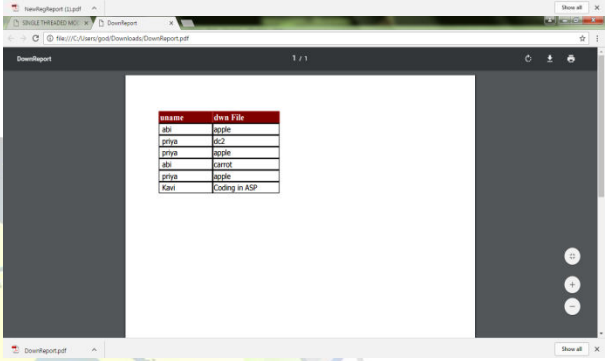
ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET) Vol. 5, Special Issue 12, April 2018









CONCLUSION

In this paper, we projected LADS, an energetic particular threaded OLTP system. A LAD extends the ease of the single-threaded model while overcoming the latter's robustness matter under diverse kinds of workloads. A LAD resolves conflicts among transactions by construct dependency graphs for which there are no aborts that may begin due to conflicts during the transaction execution. It distributes inward workloads to available workers in an impartial manner to achieve higher parallelism and efficiency. LADS also leverages modern hardware features. Our extensive experimental study shows that LADS can attain up to 20 superior throughputs than three state-of-the-art systems.

REFERENCE

[1] R. R. Schaller, "Moore's law: Past, present, and future," IEEE Spectrum, vol. 34, no. 6, pp. 52–59, 1997.

[2] H. Kimura, "Foedus: Oltp engine for a thousand cores and nvram," in Proc. Spec. Int. Group Manag. Data, 2015, pp. 691–706.

[3] X. Yu, G. Bezerra, A. Pavlo, S. Devadas, and M. Stonebraker, "Staring into the abyss: An evaluation of concurrency control with one thousand cores," Proc.

Very Large Data Base Endowment, vol. 8, no. 3, 2014, pp. 209– 220.

[4] S. Harizopoulos, D. J. Abadi, S. Madden, and M. Stonebraker, "Oltp through the looking glass, and what we found there," in Proc. Spec. Int. Group Manag. Data, 2008.

[5] I. Pandis, R. Johnson, N. Hardavellas, and A. Ailamaki, "Data-oriented transaction execution," Proc. Very Large Data Base Endowment, vol. 3, no. 1–2, 2010, pp. 928–939.

[6] R. Kallman, H. Kimura, J. Natkins, A. Pavlo, A. Rasin, S. Zdonik, E. P. C. Jones, S. Madden, M. Stonebraker, Y. Zhang, J. Hugg, and D. J. Abadi, "H-store: A high-performance, distributed main memory transaction processing system," in Proc. Very Large Data Base Endowment, vol. 1, no. 2, 2008, pp. 1496–1499.

[7] A. Kemper and T. Neumann, "Hyper: A hybrid oltp&olap main memory database system based on virtual memory snapshots," in Proc. Int. Conf. Data Eng., 2011.

[8] T.-I. Salomie, I. E. Subasu, J. Giceva, and G. Alonso, "Database engines on multicores, why parallelize when you can distribute?" in Proc. EuroSys, 2011, pp. 17–30.

[9] A. Whitney, D. Shasha, and S. Apter, "High volume transaction processing without concurrency control, two phase commit, sql or c++," in Proc. 7th Int. Workshop High Performance Trans. Syst., 1997, pp. 211–217.

[10] R. H. Louie, Y. Li, and B. Vucetic, "Practical physical layer network coding for two-way relay channels: performance analysis and comparison," IEEE Trans. Wireless Commun., vol. 9, no. 2, pp. 764–777, Feb. 2010.

[11] D. J. DeWitt, R. H. Katz, F. Olken, L. D. Shapiro, M. R. Stonebraker, and D. A. Wood, "Implementation techniques for main memory database systems," in Proc. Spec. Int. Group Manag. Data, 1984, pp. 1–8.

[12] H. Zhang, G. Chen, B. C. Ooi, K.-L. Tan, and M. Zhang, "In-memory big data management and processing: A survey," IEEE Trans. Knowl. Data Eng., vol. 27, no. 7, pp. 1920–1948, Jun. 2015.

[13] K.-L. Tan, Q. Cai, B. C. Ooi, W.-F. Wong, C. Yao, and H. Zhang, "In-memory databases: Challenges and opportunities from software and hardware perspectives," ACM Spec. Int. Group Manag. Data Record, vol. 44, no. 2, pp. 35–40, 2015.

[14] H. Kimura, G. Graefe, and H. A. Kuno, "Efficient locking techniques for databases on modern hardware." in Proc. ADMS, 2012, pp. 1–12.

[15] V. Gottemukkala and T. J. Lehman, "Locking and latching in a memory-resident database system," in Proc. Very Large Data Base, 1992, pp. 533–544.

[16] K. Ren, A. Thomson, and D. J. Abadi, "Lightweight locking for main memory database systems," Proc. Very Large Data Base Endowment, vol. 6, no. 2,2012, pp. 145–156.

[17] H. T. Kung and J. T. Robinson, "On optimistic methods for concurrency control," ACM Trans. Database Syst.ems, vol. 6, no. 2, pp. 213–226, 1981.

[18] R. Agrawal, M. J. Carey, and M. Livny, "Concurrency control performance modeling: Alternatives and implications," ACM Trans. Database Syst, vol. 12, no. 4, pp. 609–654, 1987.

[19] P. A. Bernstein, V. Hadzilacos, and N. Goodman, "Concurrency control and recovery in database systems," Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1987.

[20] C. Mohan, H. Pirahesh, and R. Lorie, "Efficient and flexible methods for transient versioning of records to avoid locking by read-only transactions," in Proc. Spec. Int. Group Manag. Data, 1992, pp. 124–133.

