ISSN: 0976-3031

Available Online at http://www.recentscientific.com

CODEN: IJRSFP (USA)

International Journal of Recent Scientific Research
Vol. 8, Issue, 12, pp. 22456-22465, December, 2017

International Journal of Recent Scientific Research

DOI: 10.24327/IJRSR

Research Article

DESIGNING A NOVEL DATABASE ENGINE FOR HYBRID WORKLOADS

Kanagalakshmi S*[1] and Ramar K[2]

[1]Department of Computer Applications, Manonmaniam Sundaranar University, Tirunelveli
[2]Einstein College of Engineering, Sir C.V. Raman Nagar, Tirunelveli

DOI: http://dx.doi.org/10.24327/ijrsr.2017.0812.1269

ARTICLE INFO

Article History:

Received 10th September, 2017
Received in revised form 14th October, 2017
Accepted 08th November, 2017
Published online 28th December, 2017

Key Words:

Database, Workload, Replication, OLTP and OLAP

ABSTRACT

Database systems were mainly used for online transaction processing in which queries for On-Line Transaction Processing (OLTP) systems that typically access only a small portion of a database, Online Analytical Processing (OLAP) queries may need to aggregate large portions of a database which often leads to performance issues. In this paper, a proposed BatchDB new memory database engine is designed and implemented in hybrid OLTP and OLAP workloads for distributed system. This method is chosen because of high level of data freshness and minimizes load interaction between the transactional and analytical engines. It facilitates real time analysis over fresh data under fixed SLAs for both OLTP and OLAP workloads and it dependent on replication, workload type (OLTP and OLAP) and a light-weight propagation of transactional updates. The experimental results are carried out on standard benchmarks of TPC-C and TPC-H, it is observed that the proposed BatchDB achieves better throughput and latency for the corresponding transactional and analytical workloads.

INTRODUCTION

The workload database engines need to efficiently handle both Transactional (OLTP) and Analytical (OLAP) workloads with assurances for throughput, latency and data freshness. At present the running analytics on the data, must not fit for OLTP performance which is bound by severe SLAs for response time and throughput. The OLAP does not handle the small set of house users without requirement for guaranteed performance. It is necessary to implement the hybrid OLAP-OLTP as a service to large number of users with SLAs on data-freshness and performance for various applications such as business.

The OLTP and OLAP workloads is difficult to manage because they require different algorithms and data structures for implementation. A common approach for handling the workloads is proposed to keep a separate data warehouse for OLAP which is isolated from the OLTP system. Data warehouse systems are optimized for read only analytical workloads and which are periodically refreshed through a batch of job containing the latest data updates. This provides efficient performance for both the workloads and the ability to tune each system independently.

Several researchers had done lot of implementations in this domain and several alternatives have been recently introduced for some techniques such as SAP HANA [17], HyPer [28], SQL Server [31], MemSQL [12], Oracle [29], etc. However, it shows limitations on performance impacted of database workloads. This paper presents Batch DB, an alternative design of a database engine architecture, which handles hybrid workloads with efficient performance in terms of data freshness, consistency, and elasticity.

To accommodate both OLAP and OLTP, BatchDB primarily relies on replication, trading off space for performance isolation, with a replica for both OLTP and OLAP workloads. This allows for workload specific optimizations for every replica and physical isolation of resources dedicated for each workload. To efficiently maintain the replica up-to-date without affecting OLTP and OLAP performance, BatchDB relies on lightweight update extraction and isolated execution of queries and updates at the OLAP replica. This process can be achieved by incoming OLAP queries first queued and then scheduled in batches, one batch-at-time. Execution of each batch of queries is shared and done as part of a single read-only transaction on the latest version of the data.

Propagated OLTP updates are also first queued and then efficiently executed in-between two batches of queries. This enables version agnostic scan processing at the OLAP replica and logical isolation between the query processor and update

Corresponding author:* **Kanagalakshmi S
Department of Computer Applications, Manonmaniam Sundaranar University, Tirunelveli

propagation. The BatchDB uses an efficient way of extracting, propagating and applying updates both within one and across multiple machines using Remote Direct Memory Access (RDMA) over InfiniBand. All these features help BatchDB achieve performance isolation for hybrid workloads with efficient throughput and response times. The experimental results, based on a hybrid Transaction Process Control (TPC) TPC-C and TPC-H workload [11], show that BatchDB achieves good performance isolation between the hybrid workloads, having a negligible ten percent overhead on each other's performance. Moreover, the replication mechanism is capable of propagating and applying the updates at the analytical replica at a rate higher than any TPC-C throughput achieved to date and can thus be integrated with other transactional engines. Finally, the system has competitive performance for individual OLTP and OLAP workloads and is superior to existing systems designed for hybrid workloads in terms of overall throughput.

Objective

The main objectives of this work

- To identify the multiple resource sharing which cause unstable performance of OLTP and OLAP workloads includes
- Explicit resource sharing depends on scheduling polices of the database engine and it can be avoided by scheduling OLTP and OLAP requests on dedicated resources.
- Implicit resources haring (e.g., of memory bandwidth and CPU caches)requires the need for having separate replicas of OLTP and OLAP workloads
- To implement a logical separation for analytical and transactional queries can be update by using a batch of queries scheduling, single snapshot replica, and efficient algorithms are deployed for executing the updates.

Design Goals

The key requirements for design engines which aims to handle Hybrid Transactional and Analytical Processing workloads (HTAP).

Performance Isolation - The SLA required efficient Latency and throughput which are provide by Database engines. The performance of OLTPare unpredictable by OLAP queries in hybrid workloads which leads to a revenue losses.

Workload Specific Optimizations- The Database engines should influence workload specific optimizations wherever applicable and it use datastructures to operate on dataformats for the given workload. It is essential to implement for delivering good and stable performance.

Update propagation & datafreshness- The OLAP queries are employed on recent version of data and also on many serious business decisions based on real-time analytics[15]. The low-latency communication are required between the individual systems for fast propagation with update mechanisms within the components.

Consistency Guarantees–To confirm that queries would have a consistent vision on the data analytical queries with high consistency guarantees (i.e., snapshot isolation).

Single System Interface- Unlike using a separate system with single interface for each workload, a single system interface which supports both analytical queries and transactions which provides the easiness of using system.

Elasticity-The efficient elasticity should be achieved by the database engine by scaling dynamically with increasing number of machines and it should take advantage of all there sources provided by modern distributed environments

The rest of the paper is organized as follows. Section II deals the related works about the BatchDB and OLAP process and limitations. Section III deals with the System Architecture of proposed method OLAP-OLTP for distributed system. Section IV presents the process of Transactional OLTP replica and OLAP replica. Section V, gives the result and performance analysis. Finally, the overall proposed method concludes in section VI.

Related Works

Currently, customers with high rates of mission critical transactions have split their data into two separate systems, one database for OLTP and one so-called data warehouse for OLAP. Dehne *et.al* (2015) introduced CR-OLAP, a scalable Cloud based Real-time OLAP system based on a new distributed index structure for OLAP, the distributed PDCR tree. CR-OLAP utilizes a scalable cloud infrastructure consisting of multiple commodity servers (processors). With increasing database size, CR-OLAP dynamically increases the number of processors to maintain performance. Distributed PDCR tree data structure supports multiple dimension hierarchies and efficient query processing on the elaborate dimension hierarchies which are so central to OLAP systems.

Benker *et.al* (2013) presented a proposal of a software architecture that enables the integration of OLTP- and operational OLAP-capabilities for right-time decision making. The second contribution is the identification of concepts for the application of non-relational technologies (NoSQL) in enterprise application systems in order to realize benefits of polyglot persistence. For that first step the proposed architecture focuses on a single business process scenario. Enterprise application systems can be categorized as OLTP and OLAP systems. OLTP systems are used to realize the functionality of operational business processes. OLAP systems perform business analysis and deliver decision-relevant information. Data extracted from operational data structures offers a common and valuable input for those systems.

While allowing for decent transaction rates, this separation has many disadvantages including data freshness issues due to the delay caused by only periodically initiating the Extract Transform Load data (ETL) staging and excessive resource consumption due to maintaining two separate information systems. Kemper *et.al* (2011) developed an efficient hybrid system, called Hyper that can handle both OLTP and OLAP simultaneously by using hardware-assisted replication mechanisms to maintain consistent snapshots of the transactional data. Hyper is a main memory database system that guarantees the ACID properties of OLTP transactions and executes OLAP query sessions (multiple queries) on the same, arbitrarily current and consistent snapshot. The TPC-C benchmark was designed to evaluate OLTP database system performance and the TPC-H benchmark for analyzing OLAP

query performance. The utilization of the processor-inherent support for virtual memory management (address translation, caching, and copy on update) yields both at the same time: unprecedentedly high transaction rates as high as 100000 per second and very fast OLAP query response times on a single system executing both workloads in parallel.

Lu *et.al* (2016) introduced a VERTICA and SAP HANA to implement ACID transaction functionality in a distributed columnar database. In telecom companies, MPP columnar database provides fast analyzing ability with built in ACID support. By testing several optimization methods on Vertica and SAP HANA, it is possible to enhance the OLAP performance of MPP database. A high speed execution engine is implemented to deal with OLAP workloads. On higher level applications runs by different group of users share a common interface for transactions of database to get results from it.

Conn *et.al* (2005) discussed that advantage of the OLAP data repository is that is has a long time horizon from which to perform analysis and discover trends and patterns within the business, but the disadvantage is that data may not be (relatively) recent enough to qualify as real-time data for business intelligence purposes. This research will be conducted as a qualitative study by drawing on relevant literature and other scholarly documentation to investigate any proposed architectures or processes for integrating the OLTP and OLAP environments.

System Architecture

The BatchDB's [26] key principles and methodology are presented below, to address the design goals and the flaws of previous methods are mentioned and also includes the assumptions and trade-off methods.

Proposed BatchDB Components

BatchDB's architecture is based on replication and it uses a primary replica as OLTP to handles all updating transactions (Figure 1 left), and secondary replica is OLAP it performs analytical queries (Figure 1 right).

Each replica system has a set of resources and it allows each replica to be optimized for the given workload which is located in BatchDB, either on the same shared-memory machine (on different NUMA nodes) or multiple memory machines. (e.g., Both OLTP and OLAP have separate execution engines in which OLTP is considered for pre-compiled stored actions and the OLAP for handling the ad-hocqueries). The OLTP dispatcher are assigned to schedules the requests and also it is responsible for conveying requests to worker threads for the successful transactions. Worker threads export a physical log of updates containing information on the snapshot version for each affected tuple to be used for propagating the updates to the analytical replica. The OLAP dispatcher schedules OLAP queries and makes sure that the OLAP replica operates on one-batch-at-a-time processing with data kept in a single-snapshot-replica from the primary replica, without negatively affecting the OLAP queries performance. For this method, physical storage is version-agnostic and maintains only a single data version (batch) and removing the dealing with numerous snapshots for Batch DB scan processing and eliminates the overhead of logical contention and synchronization with the transactional component for applying the updates. Figure1 shows an BatchDB's architecture.

Addressing the Design Goals

Performance Isolation-Initially it addresses the design goal on replication by using individual replicas for dissimilar workloads and it allows for physical isolation of the allocated hardware resources, either by locating the replicas on different machines, or executed on the same machine on separate NUMA nodes. Secondly, the main objective is to remove the logical contention

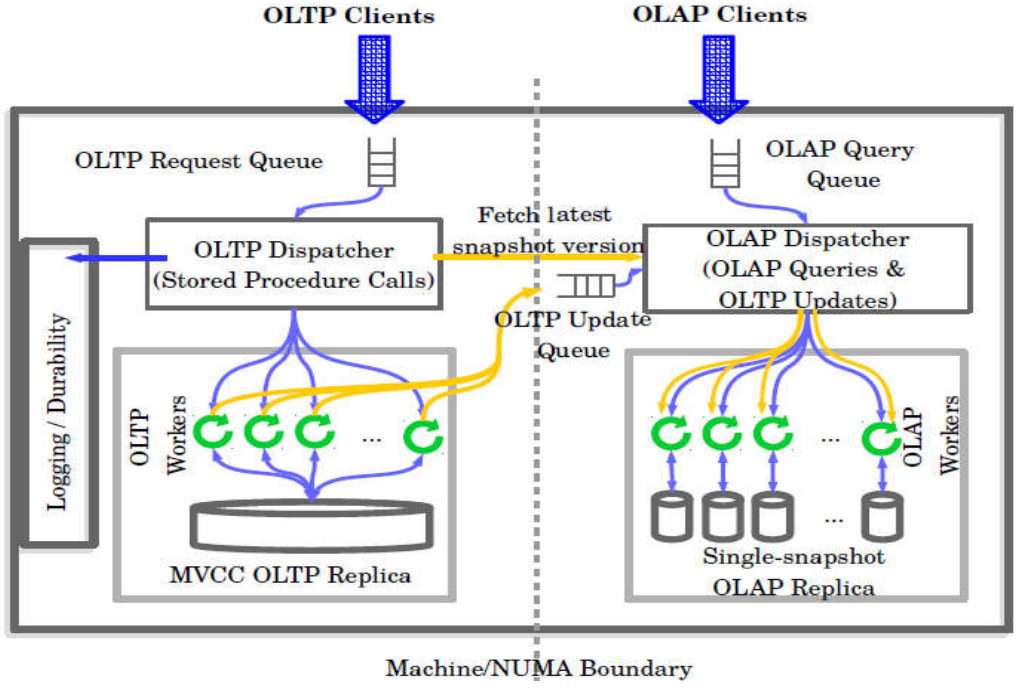

Figure 1 System Architecture

among the two components by executing the performance isolation by the batch-based processing which is done by the OLAP dispatcher.

Data freshness **-**Instantly the OLTP worker threads forwards the updates to the OLAP replica to achieve an efficient data freshness. At the end of OLTP branch execution the updates are pushed for transactions in two ways are if OLAP dispatcher requested for the latest snapshot version and latest updates leads so longer than a limited(configurable) period, (which is set to200ms). So the execution time for a transaction's updates of OLAP replica depends on the OLTP transactions durations (10s of milliseconds) which is normally less than the execution time of analytical queries, then the data freshness are observed by the OLAP queries response time.

Workload Specific Optimizations-By design, BatchDB's workload specific replicas are applicable for optimization technique for workload. In this situation the design and implementation of both the OLAP and OLTP components are employed to improve the workload performance manually.

Consistency Guarantees **-** The OLTP and OLAP requires the snapshot isolation which is provided by BatchDB. The transactional requests, are processed by MVCC in which the OLAP replica uses a single snapshot to address the analyticalqueries.

Interfaces – BatchDB disclosures the interface to users of both workloads. There is no explicit requirement for users to distinguish requests for the two replicas. The OLAP replicas operate on a single snapshot version as one batch of read-only queries at a time and it can be viewed as indexes.

Elasticity **-** BatchDB can scale by conveying more replicas as the quantity of machines increments. Notwithstanding unique replicas sorts, various cases of a similar part can be made so as to circulate the load evenly over the entire framework. The high bandwidth capacity of modern networks in blend with RDMA primitives makes it conceivable to circulate updates to countless. The design principles wan also be applied to provide specialized replicas to other workload types (e.g., offline batch analytics for long running queries, graph processing, and so forth).

Trade-offs

Individual Query Latency **-** One trade-off in BatchDB is the individual query latency for OLAP workloads. Specifically, the prerequisite that all simultaneous OLAP questions must be co-planned together to separate them from execution of OLTP updates. The impact of this is query latency relies upon the latency of other simultaneous OLAP queries.

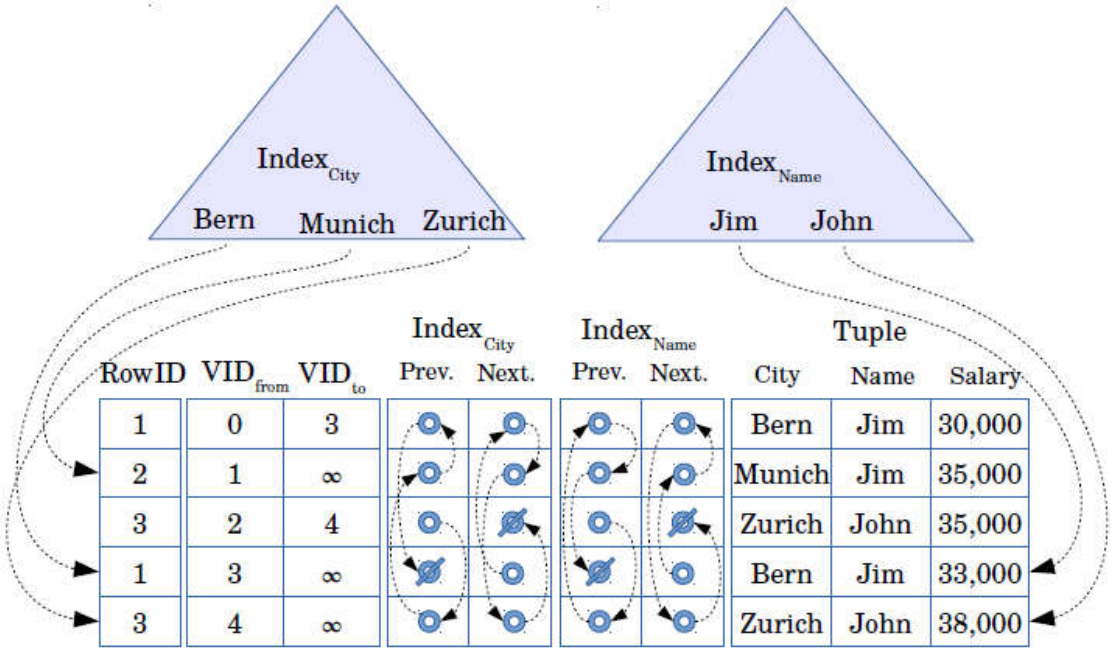

Figure 2 OLTP Record Storage Format and Index Layout

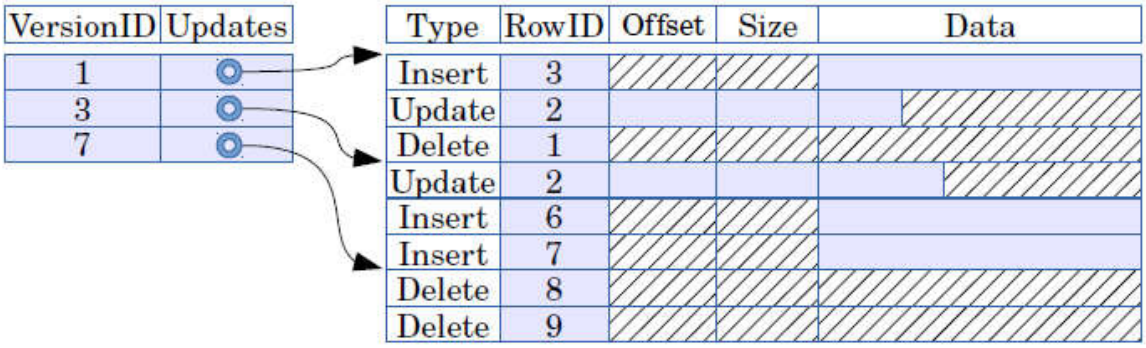

Figure 3 Propagated Update Format for a Specific Table from a Single OLTP Worker Thread

This is especially valid for a framework offering on the online analytics on real-time information as it is normally accepted (and frequently required because of SLAs) that all queries are executed within couple of moments. In the event that a client wishes to run a query that would take more time to execute, for example minutes or hours, this query can be dealt with diversely as an offline non- real-time query.

Transaction Semantics- While our framework gives snapshot isolation certifications to OLAP queries it doesn't give full transactional flexibility. For example, the data version on which queries are handled is chosen by the framework when the query (alongside different query in its cluster) begins executing. To detach the execution of OLAP queries and OLTP updates, the OLAP replica is updated at a coarser granularity and clients cannot pick the correct variant of data to run their queries on. Subsequently, interactive sessions with long running read only transactions where clients submit queries in a steady progression are impractical in present system.

Transactional Component (Oltp Replica)

The Transactional (OLTP) segment appeared in Figure 1, contains the primary replica of the database motor whose intention is to deal with both updates containing transactions and short latency-sensitive read-only requests. Aside from preparing updates to be spread to the OLAP replica, the plan choices for actualizing the OLTP replica can be careless in regards to the prerequisites of OLAP workloads. In this manner, no compromise should be made when managing the OLTP workloads and any OLTP particular optimizations are appropriate.

The implementation of BatchDB's OLTP component on Hekaton[13], multi-version concurrency control[6]and lock-free indexing data-structures, as opposed to partitioning to achieves cal ability in multi-core systems. This is unlike the approaches taken by H-Store [27], or HyPer [29], which are more suitable for partition able workloads.

Storage Layout and Indexes – Figure 2 shows the storage layout and indexing data-structures for a sample relation. Basically, row-oriented storage is used as it is most productive for the point lookups and updates, which are necessary to transaction handling. Like Hekaton a hash-based and a tree-based index with respect to the lock- free Bw-Tree [30]. A simplified version of the Bw-Tree that depends on atomic multi-word compare-and-swap updates. Moreover, the physical records contain a twofold connected rundown for every indexes to encourage simpler traversal into the indexes.

Transaction Execution and Concurrency Control - For effective handling of OLTP requests for the framework locally aggregated put away methods with customers sending their requests in the form of stored procedure calls. To execute the stored procedures, the OLTP part possesses a committed arrangement of worker threads. The worker threads are assembled on a single NUMA node to dodge high synchronization overheads over the interconnect for delicate parts of the design, for example, epoch management, garbage collection, memory allocation, and so on. The present usage gives snapshot isolation assurances to exchanges utilizing a multi-version concurrency control.

Scheduling - The responsibility for allocation of OLTP requests (stored procedure calls) to worker threads is appointed to the OLTP dispatcher that, like the OLAP dispatcher, plans operations in batches taking a shot at one cluster at any given moment. As appeared in Figure 1, approaching OLTP requests for are first lined up in the OLTP queue while the framework is caught up with executing the present batch of OLTP quires. At the point when the present batch is done, the OLTP dispatcher de-queues all requests from the OLTP quires and enquires them to the OLTP worker threads in a round robin form. This trade off individual query latency to acquire benefits that emerge in assessing many requests as a bunch. For example the logging of updates to tough storage, the epoch management for the lock-free data- structure, the junk gathering and the proliferation of the updates to the OLAP replicas all benefit from such batch based execution. In such mode, threads can join numerous operations in a similar epoch and amortize the cost of changing challenged atomic counters. Not yet investigated utilizing the OLTP request for grouping for advancing the execution of the concurrency protocol, as proposed by BOHM [16].

Logging - For strength the OLTP dispatcher logs the successful update transactions to a durable storage medium before the reactions are sent to the clients. To limit the impact of logging on execution. By utilizing snapshot isolation, the data on the read and snapshot versions needs should be additionally logged for correct recovery. Besides, logging is performed on a batch premise (as gathering confer [12]) to conceal the I/O dormancy for numerous OLTP requests. Note that as the OLAP replica isn't moved down by a strong medium, if there should arise an occurrence of failures it should be recouped by reading a snapshot and catching up with new updates from the primary replica.

Update propagation - To keep the secondary replica steady and up-to-date, the OLTP part likewise sends out a log of updates isolate from the durable log. Not at all like the durable log that contains legitimate updates, have the engendered updates contained the physical updates to singular records. This empowers proficient application of the updates on the secondary replica. To avoid expensive synchronization among OLTP worker threads, each threads generates its own arrangement of updates to be engendered. A case set of updates from a worker threads is portrayed in Figure 3. This case contains eight propagated updates from three committed transactions. Updates from a single thread might be interleaved with updates from different threads during the propagation process. For example, in this case the updates from a transaction with version ID 2 are a piece of the update set of an alternate threads. Each updates contains:

The Type of the update can be either a newly inserted tuple, an update to an existing tuple or a delete of an existing tuple; The Row ID integer which uniquely specifies the tuple that corresponds to this update. The RowID is equivalent to the primary key of the relation and is used to efficiently locate the corresponding tuple at the secondary replica; The Offset and Size in bytes which are used to update existing tuples on finer sub-tuple granularity; The Data which contains either the data of the newly inserted tuple or the payload of the update to an existing record.

Analytical Component (OLAP Replica)

The analytical component is shown in Figure 1, and contains the secondary replica of the database engine.

Query scheduling **-** In order to avoid synchronization and execution connection between running the OLAP queries and applying the OLTP updates, the OLAP dispatcher executes quires in batches. A batch is executed as a read-only transaction on the latest snapshot version. Before executing the following batch of quires, the dispatcher recovers from the OLTP segment the most recent conferred snapshot form, and applies the propagated updates on its replica up-to that version. Moreover, as just a single batch of queries is executed at once, the OLAP engine does not have to store more than a single version of the data, i.e., can be form careless.

The batch scheduling of BatchDB is like the one utilized by Crescando [29] and SharedDB [19]. The method of BatchDB contrasts from these frameworks as it batches all simultaneous OLAP quires in the framework to confine the OLAP quires preparing from the up-dates spread by the primary OLTP replica. The main adaptation of HyPer additionally gathered the OLAP quires by session, and ensured that all quires in a session worked on a similar snapshot version. Query execution: Since the query scheduler executes query in batches, despite the fact that it isn't important, by utilizing a query handling motor that likewise takes advantage of shared execution. The OLAP segment utilizes thoughts exhibited by earlier work on shared scans (to share memory transmission bandwidth crosswise over outputs and query predicate assessment); more perplexing quires preparing (to share execution of join operations for more productive use of CPU and DRAM bandwidth); and scheduling optimizations [20]. Earlier trials on this segment [16] demonstrate that, for extensive workloads, it can give higher throughput than best in commercial engines in analytical query processing.

utilized at the OLAP replica to particularly recognize the tuples alluded by the updates. Moreover, as show in Figure 1, the data in the OLAP replica is on a horizontally (delicate) apportioned in view of a hash estimation of the RowID quality. This empowers both proficient (NUMA-neighborhood) scan handling and quick utilization of OLTP updates on present day multi-center machines. Comparable soft based partitioning has additionally been utilized by various different frameworks.

To facilitate efficient matching of OLTP updates and tuple locations, the OLAP component maintains a hash index of the data on the RowID attribute. The process of applying the propagated up-dates using the RowID and the hash indexes consists of three steps which are illustrated in Figure 4

Step1 **-** It update sets from multiple OLTP threads are ordered by the snapshot versioned (VID). This step is the fastest as it only orders the update pointers using as can with complexity linear in the number of new snapshot versions.

Step2- It is executed when the OLAP dispatcher obtains the latest committed snapshot versioned to be used for the current batch of queries. Thereafter, the updates corresponding to snapshot versions upto and including the VID are propagated to the corresponding partitions based on a hash value of the RowID attribute.

Step 3 **-** The updates for each partition are applied using the hash index on RowID to find the location of tuples for operations that either update or delete a tuple. When a tuple is deleted, the slot for that tuple is marked as empty, as a signal for the scan process or to ignore the tuple at that location. In case of inserts, the tuple is inserted into the next free slot of the partition (possibly at a location where a tuple was recently deleted) and the hash index is populated accordingly.

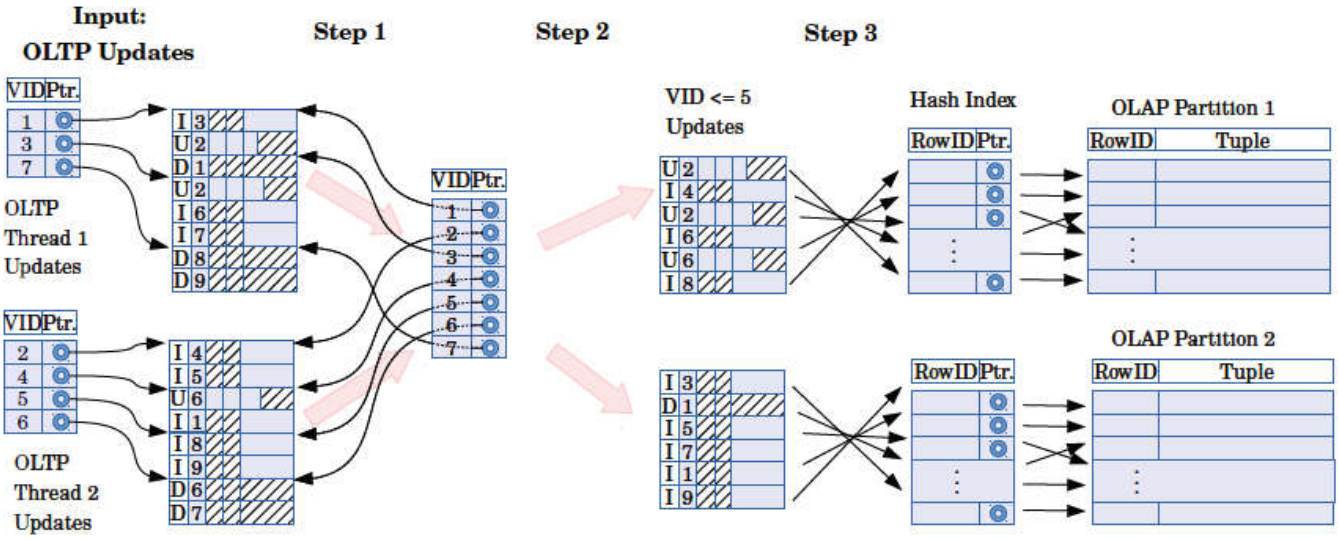

Figure 4 Propagated Updates from OLTP Replica into OLAP Replica

Update propagation **-** Lastly, to empower quick use of updates, all tuples of replicated tables are clarified with a RowID integer attribute in both replicas. The RowID is basically an essential key trait escaped the client. As portrayed before, all propagated updates from the OLTP replica contain the RowID trait which is

In case of updates, tuples are updated inplace at the granularity of single attributes to avoid rewriting the entire tuple. The attribute to be modified is identified using the offset and size fields depicted in Figure3.

All three steps are easily parallelizable. The most time for consuming step is step 3, as it contains multiple random accesses. Technically, this step resembles a hash join with the hash index used as a hash table to join the updates with the existing data. In general, there has been a significant work on speeding up joins with state-of-the-art algorithms reaching hundreds of millions of tuples per second on modern multi-cores[3]. To put these numbers into perspective the highest reported performance for the industry standard OLTP benchmark (TPC-C) would correspond to about tens of millions updated tuples per second. Therefore, this approach of propagating updates from an OLTP to an OLAP replica can satisfy the needs of most systems. Our algorithm optimizes for both main-memory bandwidth and latency. Hash buckets from the hash index a restore din a single cache line and accessed using a grouped software prefetching technique [10] to minimize high random main-memory access latencies. The propagating updates with an array approach, where the RowID is directly used as an offset in the storage of the OLAP replica. However, it did not bring significant performance gains compared to an optimized hash index approach, while on the other side requires coordination among OLTP worker threads in the use and reuse of RowID integers.

Storage Layout and Data Transformation- The current implementation of the OLAP component uses an uncompressed row-oriented storage (as shown in Figure4), similar as to the OLTP replica. Note, however, that this is not a design requirement and as part of future workplan to optimize it using a column-store format, which has been shown to be more efficient for analytical workloads, and can further benefit from compression, reduced I/O and memory costs.

Network Communication

BatchDB allows the OLAP replica to be either co-located on the same physical machine as the primary replica, or to reside on a different node. In the latter setup, the system makes use of modern low-latency networks to efficiently propagate the updates.

Remote Direct Memory Access (RDMA) [23] is used to improve the latency of small messages and reduce the CPU overhead for large data transfers by avoiding intermediate copy operations in-side the network stack. RDMA has been used for several database systems and the ideas are based on the platforms developed for running query-at-a-time joins at large scales.

RDMA offers one-sided and two-sided operations. When using one-sided read or write operations, the initiat or of the request directly manipulates a section of memory which has previously been exposed by the remote machine. The CPU on the target machine is not notified nor involved in the data transfer. Two-sided operations on the other hand represent traditional message passing semantics. The receiver has to allocate several receive buffers an disnotified when a new message has been placed in any of these locations. A downside of RDMA is that the application is burdened with extra complexity of network buffer management[18].

In BatchDB, each machine registers several RDMA receive buffers with the network card. Small messages of less than1024KB are directly written to these buffers using two-sided RDMA operations. Larger messages cannot be transmitted directly, and require a hand shake operation. During the handshake, the sender transmits the required buffer size to the receiver, which in turn allocates a new buffer and register sit with the network card. The receiver responds with the buffer address and the necessary access credentials. After this exchange, the sender can initiate the data transfer using a one-sided RDMA write operation. Once the transfer is complete, the receiver is notified by the sender. To reduce the overhead of memory allocation and registration, large RDMA-buffers are pre-allocated and cached in a buffer pool.

In addition to a low latency, modern networks also provide high band width, which is important for data-intensive applications. The communication mechanisms mentioned above allow for optimal use of the network bandwidth for both small and large messages. Propagating updates from the primary replica to one secondary copy does not fully saturate the through put of a 4xFDR Infini B and network. Not being limited by the network bandwidth enables the primary node to simultaneously transmit updates to multiple secondary copies, thus making the system elastic and scalable.

Experimental Results

The Experimental results of proposed hybrid OLTP and OLAP are simulated using Apache Hadoop. In the experiments, BatchDB runs on one or two machines, and the client work load is generated on separate machines that communicate via a1Gbit Ether net network. The hybrid bench mark is a mix of standard OLTP and OLAP benchmarks, namely TPC-C and TPC-H. In particular, the OLTP part is an unmodified version of the TPC-C benchmark, while the OLAP part contains a set of analytical queries inspired by TPC-H. The latter are modified to match the TPC-Cschema, plusafew missing TPC-H relations.

Throughput

The experimental results of proposed hybrid OLTP and OLAP are evaluated in terms of throughput with respect to clients. The proposed method are run with 200 warehouses and 2000 clients and compared with HyPer [29] and SAP HANA [17]. Table 1 shows the throughput for proposed hybrid OLTP-OLAP for different warehouse such as 50, 100 and 200 warehouse with respect to clients.

Table 1 Throughput for Hybrid OLTP-OLAP

Techniques	Clients	50W	100W	200W
Hyper	0	0	9	15
	500	9	15	23
	1000	15	25	43
	1500	20	35	62
	2000	25	40	70
SAP HANA	0	15	20	25
	500	25	45	39
	1000	35	55	62
	1500	42	65	69
	2000	49	75	79
Proposed Hybrid OLTP-OLAP	0	25	32	35
	500	42	49	55
	1000	56	62	72
	1500	67	75	80
	2000	79	82	85

From the table it clearly shows that the proposed Hybrid OLTP-OLAP provides the high throughput for 200 warehouse with 2000 clients respect to increasing clients.

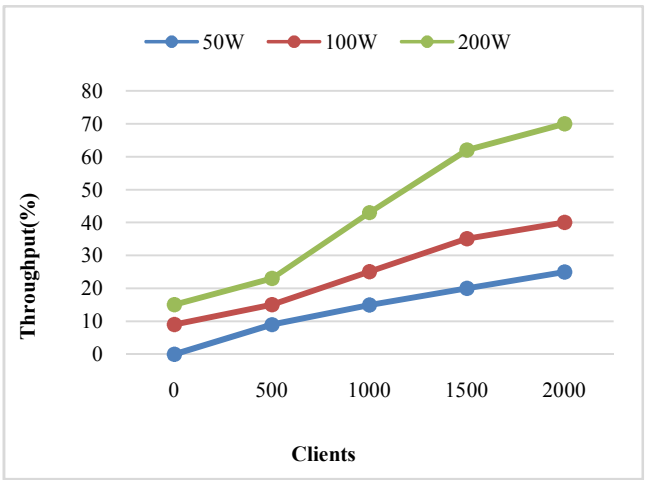

Figure 5 Throughput for HyPer

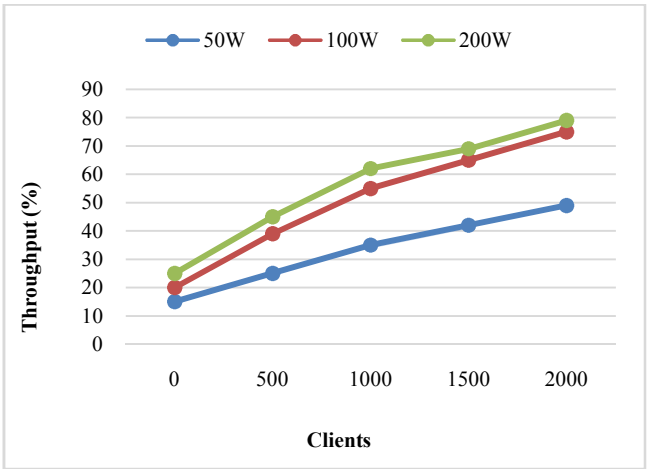

Figure 6 Throughput for SAPHANA

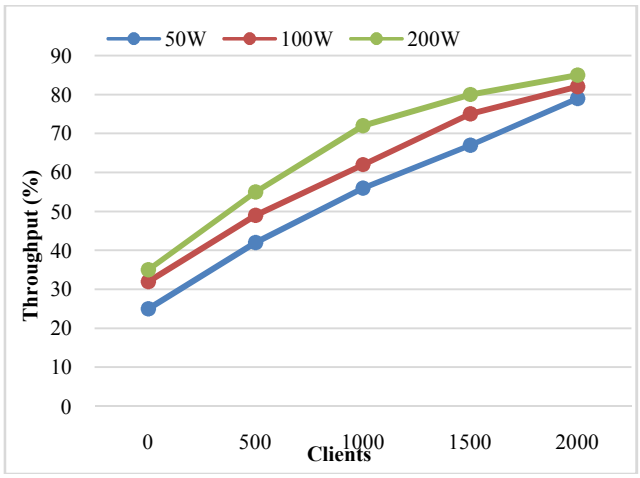

Figure 7 Throughput for Proposed Hybrid OLTP-OLAP

Figure 5-7 shows that throughput of HyPer, SAP HANA and proposed Hybrid OLTP-OLAP. It is clear that the proposed method achieve a maximum of 110 thousand TPC-C transactions with 200 warehouses and 2000 clients when compared to existing HyPer and SAP HANA.

Table 2 shows the transaction latencies for 200 warehouse with respect to client for 50th percentile, 90th percentile and 99th percentile.

Table 2 Transaction Latency

Techniques	Clients	50th	90th	99th
HyPer	200	20	25	30
	500	25	29	40
	1000	32	40	45
	2000	40	50	55
SAP HANA	200	15	10	15
	500	15	20	23
	1000	20	35	39
	2000	25	40	45
Proposed Hybrid OLTP-OLAP	200	11	15	18
	500	15	25	25
	1000	22	29	30
	2000	30	35	40

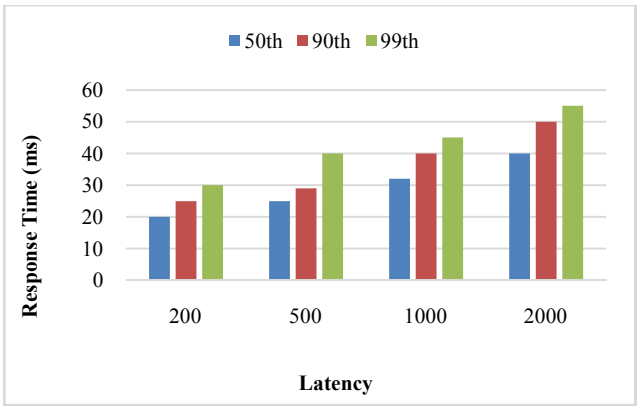

Figure 8 Transaction Latency for Hyper

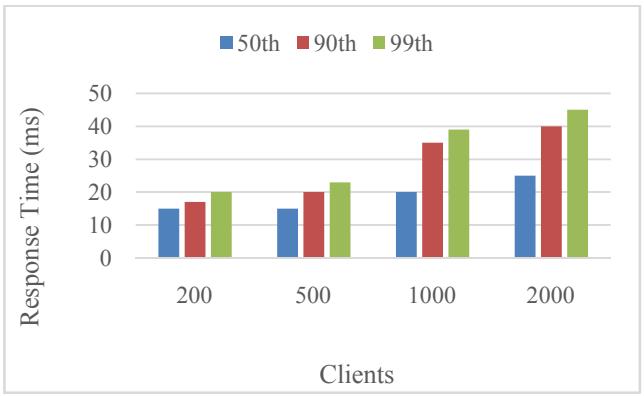

Figure 9 Transaction Latency for SAP HANA

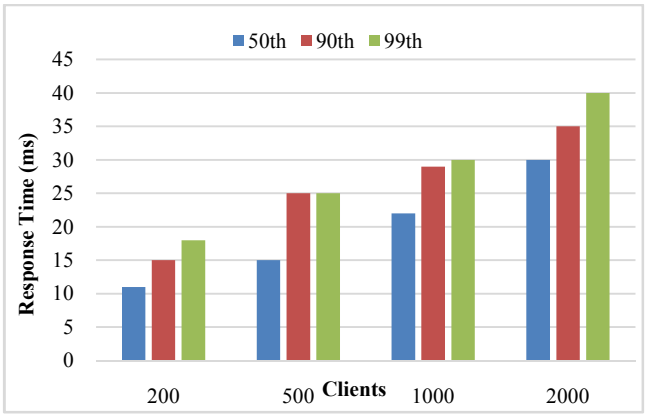

Figure 10 Transaction Latency for Proposed Hybrid OLTP-OLAP

Figure 8-10 shows the impact of batching queries on transaction latencies of proposed Hybrid OLTP-OLAP. It is noted that when maximum throughput is reached, the 99th percentile shows 40ms, which is well below the 5 second limit

for 90th percentile as per the TPC-C specification. The proposed method shows efficient transaction latency when compared to other techniques.

CONCLUSION

In this paper a BatchDB, system that efficiently handles hybrid OLTP and OLAP workloads with strong performance isolation is implemented. BatchDB relies on primary-secondary form of replication with the primary replica handling OLTP workloads and updates propagated to a secondary replica that handles OLAP workloads. To enable query processing on latest data with snapshot isolation guarantees and minimum impact on query performance, the queries and updates at the OLAP replica are queued and scheduled in batches with the system working on one batch at-a-time. Furthermore, updates are extracted and applied from the OLTP replica at the OLAP replica efficiently, incurring a small overhead to overall execution time. Updates can also be propagated to a remote replica via RDMA over InfiniBand for added scalability and isolation of performance. The results confirm that, unlike existing systems, BatchDB is able provide high performance isolation between the workloads leading to predictable performance.

References

1. J. Arulraj, A. Pavlo, and P. Menon. Bridging the Archipelago Between Row-Stores and Column-Stores for Hybrid Workloads. SIGMOD, pages 583-598, 2016.
2. S. Arumugam, A. Dobra, C. M. Jermaine, N. Pansare, and L. Perez. The DataPath System: A Data-centric Analytic Processing Engine for Large Data Warehouses. In SIGMOD, pages 519-530, 2010.
3. C. Balkesen, J. Teubner, G. Alonso, and M. T. Özsu. Main-memory hash joins on multi-core CPUs: Tuning to the underlying hardware. In ICDE, pages 362-373, 2013.
4. C. Barthels, S. Loesing, G. Alonso, and D. Kossmann. Rack-Scale In-Memory Join Processing using RDMA. In SIGMOD, pages 1463-1475, 2015.
5. C. Barthels, I. Müller, T. Schneider, G. Alonso, and T. Hoefler.Distributed join algorithms on thousands of cores. PVLDB, 10(5):517-528, 2017.
6. 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.
7. L. Braun, T. Etter, G. Gasparis, M. Kaufmann, D. Kossmann, D. Widmer, A. Avitzur, A. Iliopoulos, E. Levy, and N. Liang. Analytics in Motion: High Performance Event-Processing AND Real-Time Analytics in the Same Database. In SIGMOD, pages 251-264, 2015.
8. G. Candea, N. Polyzotis, and R. Vingralek. A Scalable, Predictable Join Operator for Highly Concurrent Data Warehouses. VLDB, pages 277-288, 2009.
9. G. Candea, N. Polyzotis, and R. Vingralek. Predictable Performance and High Query Concurrency for Data Analytics. VLDBJ, pages 227-248, 2011.
10. S. Chen, A. Ailamaki, P. B. Gibbons, and T. C. Mowry. Improving Hash Join Performance Through Prefetching. In Proc. ICDE 2004, pages 116-, 2004.
11. R. Cole, F. Funke, L. Giakoumakis, W. Guy, A. Kemper,S. Krompass, H. Kuno, R. Nambiar, T. Neumann, M. Poess, K.-U. Sattler, M. Seibold, E. Simon, and F. Waas. The Mixed Workload CH-benCHmark. In DBTest, pages 8:1-8:6, 2011.
12. 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 Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data, SIGMOD '84, pages 1-8, New York, NY, USA, 1984. ACM.
13. C. Diaconu, C. Freedman, E. Ismert, P.-A. Larson, P. Mittal,R. Stonecipher, N. Verma, and M. Zwilling. Hekaton: SQL Server's Memory-optimized OLTP Engine. In SIGMOD, pages 1243-1254, 2013.
14. J. Dittrich and A. Jindal. Towards a One Size Fits All Database Architecture. In CIDR, 2011.
15. Elmore, J. Duggan, M. Stonebraker, M. Balazinska,U. Cetintemel, V. Gadepally, J. Heer, B. Howe, J. Kepner, T. Kraska,S. Madden, D. Maier, T. Mattson, S. Papadopoulos, J. Parkhurst,N. Tatbul, M. Vartak, and S. Zdonik. A Demonstration of the BigDAWGPolystore System. PVLDB, 8(12):1908-1911, Aug. 2015.
16. J. M. Faleiro and D. J. Abadi. Rethinking Serializable Multiversion Concurrency Control. PVLDB, 8(11):1190-1201, July 2015.
17. F. Färber, N. May, W. Lehner, P. Große, I. Müller, H. Rauhe, and J. Dees. The SAP HANA Database – An Architecture Overview. IEEE Data Eng. Bull., 35(1):28-33, 2012.
18. P. W. Frey and G. Alonso. Minimizing the hidden cost of RDMA. In ICDCS, pages 553-560, 2009.
19. G. Giannikis, G. Alonso, and D. Kossmann. SharedDB: Killing One Thousand Queries with One Stone. VLDB, pages 526-537, 2012.
20. J. Giceva, G. Alonso, T. Roscoe, and T. Harris. Deployment of Query Plans on Multicores. PVLDB, 8(3):233-244, 2014.
21. K. Goel, J. Pound, N. Auch, P. Bumbulis, S. MacLean, F. Färber,F. Gropengiesser, C. Mathis, T. Bodner, and W. Lehner. Towards Scalable Real-time Analytics: An Architecture for Scale-out of OLxP Workloads. PVLDB, 8(12):1716-1727, Aug. 2015.
22. M. Grund, J. Krüger, H. Plattner, A. Zeier, P. Cudre-Mauroux, and S. Madden. HYRISE: A Main Memory Hybrid Storage Engine. PVLDB, 4(2):105–116, 2010.
23. J. Hilland, P. Cully, J. Pinkerton, and R. Recio. RDMA Protocol Verbs Specification (Version 1.0). Technical report, RDMA Consortium, 04 2003.
24. C. Hong, D. Zhou, M. Yang, C. Kuo, L. Zhang, and L. Zhou. KuaFu: Closing the parallelism gap in database replication. In Proc. ICDE 2013, pages 1186-1195, 2013.
25. R. Jiménez-Peris, M. Patiño Martínez, and G. Alonso. Non-Intrusive, Parallel Recovery of Replicated Data. In SRDS '02, pages 150-, 2002.
26. D. Makreshanski, J. Giceva, Claude Barthels, and Gustavo Alonso. BatchDB: Efficient Isolated Execution of Hybrid OLTP+OLAP Workloads for Interactive Applications. In Proceedings of the 2017 ACM International Conference on Management of Data

(SIGMOD '17). ACM, New York, NY, USA, 37-50, 2017.

27. R. Kallman, H. Kimura, J. Natkins, A. Pavlo, A. Rasin, S. Zdonik,. P. C. Jones, S. Madden, M. Stonebraker, Y. Zhang, J. Hugg, andD. J. Abadi. H-store: A High-performance, Distributed Main Memory Transaction Processing System. PVLDB, 1(2):1496-1499, Aug. 2008.
28. B. Kemme, A. Bartoli, and O. Babaoglu. Online reconfiguration in replicated databases based on group communication. In DSN '01, pages 117-126, July 2001.
29. Kemper and T. Neumann. HyPer: A hybrid OLTP & OLAP main memory database system based on virtual memory snapshots. In ICDE, pages 195-206, 2011.
30. S. Muthulingam, V. Raja, M. Roth, E. Soylemez, and M. Zait. Oracle Database In-Memory: A dual format in-memory database. In ICDE, pages 1253-1258, April 2015.
31. P.-A. Larson, A. Birka, E. N. Hanson, W. Huang, M. Nowakiewicz, and V. Papadimos. Real-time Analytical Processing with SQL Server. PVLDB, 8(12):1740-1751, 2015.

How to cite this article:

Kanagalakshmi S and Ramar K.2017, Designing A Novel Database Engine For Hybrid Workloads. *Int J Recent Sci Res.* 8(12), pp. 22456-22465. DOI: http://dx.doi.org/10.24327/ijrsr.2017.0812.1269

****