International Journal of Database Theory and Application
Vol.7, No.1 (2014), pp.93-102
http://dx.doi.org/10.14257/ijdta.2014.7.1.09

Discovering Database Replication Techniques in RDBMS

Anees Hussain and M. N. A. Khan

Shaheed Zulifikar Ali Bhutto Institute of Science and Technology (SZABIST), Islamabad, Pakistan

anees.fastian@gmail.com, mnak2010@gmail.com

Abstract

Data replication is a key factor to achieve scalability and fault tolerance in databases as it maintains several clones of data objects. A change made in data automatically triggers carrying out similar changes in each of the replica. A number of data replication techniques have been proposed in the contemporary literature due to its large scale application in the real world like astronomy, high energy physics and biology. In this study we provide a critical analysis of these techniques.

Keywords: *Database Replication, Database Replication Strategies, Database Transactions, Replica Control, Snapshot Isolation, Scalability, Fault-tolerance, Performance*

1. Introduction

Digital data is considered as the more valuable asset of an organization, and the organizations assign more significance to it than the software and hardware assets. Database systems are computer-based record keeping systems, which have been developed to store data for efficient retrieval and processing [1].

Data replication is a process in which duplicated data objects are created and maintained in a distributed database systems scattered on different locations. Replication techniques provide fast and reliable data access and ascertain the availability of application from the alternate data access options even if one site becomes unavailable. Before forwarding database changes to the remote locations, they are captured and stored locally. Replication technique provide access to the local database objects instead of remote server which minimizes network traffic and make sure fast access to the data. In case of system failure, data replication is much more needed because it still works due to the availability of the secondary copy of data and user can still continue to query or update the remaining locations. A replication service is needed in order to make sure data consistency across the disparate environments. Proper data distribution is also very important because it reduces cost of query access across the network and application consistency and availability [2].

Types of Replication

In distributed databases, there are three types of database replications [3]:

(a) Snapshot Replication:

This form of replication takes snapshot of data on one server and moves it to another server. This technique simply distributes the data as it appears and does not cater for updating the data. This method of replication is best suited to the environments where data changes rarely occur. Entire database snapshot is created and delivered to subscriber when synchronization occurs. Based on the scheduled specified, the entire synchronization snapshot is refreshed in published tables periodically through replication. In case of setting-up database or its maintenance, it is the easiest way of replication because a table is refreshed each time when it requires copying all data.

(b) Transactional replication

Transactional replication copies data from publisher to subscriber and whenever a transaction is made at publisher side, it is delivered to the subscriber. This type of technique is useful in server-to-server replication and is known as dynamic replication as it results in automated periodic changes in databases. In this case, when data changes occur at the publisher side then all the individual transactions made at publisher side are propagated to the subscriber. Such applications should require minimal latency i.e., time of the data changes made at publisher and arrival of those changes at subscriber end should be low. Changes made at subscriber are made read only because they are not updated back to publisher.

(c) Merge replication

In the technique of merge replication, data is distributed from publisher to subscriber which allows for both the entities to update data whether they are connected or not. However when they are connected, updates are merged between the two sites. In merge replication at various times multiple subscribers might be updating the same data in server to client environment. Changes made at the subscriber end are propagated to the publisher and other subscribers receiving data make changes offline and synchronize changes later on. The conflict occurrences are resolved by the users for which every subscriber needs different partition of data.

2. Literature Review

Wahid *et al.*, [2] discuss distribution problem of replicated databases and its optimization in a computer network. As replication enables data availability in case of any site failure and also provides local access of data. This paper presents a bio-inspired replication management approach which is based on swarm intelligence. It is decentralized and suitable. Based on the state of stored data objects, each node has the authority to start a redistribution process at any time. In redistribution a node can create, update, replicate, delete and move any data object from one node to other on the network. Redistribution process runs in the background with lower priority and it dynamically balance the load. The replication model which is introduced in this paper has n nodes and each data object is replicated to different nodes. Data objects are dynamically created at each node and accessed frequently which reduces response time and inter node communication. Each data object is deleted if it is not being used for a long time and allocation of these replication are not known to others, or each node has knowledge of only its local state and don't know about its replica. This system is decentralized and each node trigger event if any transaction (query or update) occurs and these events modify the internal state (accessed data object). If any update is made to the replicated databases then these must be synchronized. During replication mechanism updates are propagated in the background by swarms of specialized pogo ants. System described in this paper tells that each event runs in background and results in dynamic load balancing. But this approach faces the challenge of diverging replicas and conflicts between concurrent operations and it is thus applicable for those applications that tolerate occasional conflicts and inconsistent data.

Chen *et al.*, [3] discuss structure for grid databases replication. As database replication enables data availability, fault tolerance and minimal access time in grids. Most systems that use grid replication now-a-days deals with only read files. Some of the relational database products offer transaction based replication but these tools cannot cope with the grid issues. The approach discussed in this paper provides metadata registry using grid mechanism and also makes replicas of data resources. The transaction based replication is managed by high level APIs. A framework named OGSADAI manages replication running across multiple domains. This paper presents transaction based grid (database replication) framework in which registration of Meta data mechanism is employed which helps in finding data resources and their existing replicas. The consistency which is achieved between the data source and its replica in RDBMS is achieved by a unified API. It also handles the changes made to data. So grid replication model and relational databases replication model is integrated into each other. In GDRM (Grid Database Replication Model) replication control services manages the replication actions which are also grid services. Whereas the meta-data information of data and their replicas are stored in metadata registry and grid transfer mechanism is handled by transfer services. Various mechanism of relational database replication like DB2, Oracle, MS SQL Server as a plugin can be used in the model which will monitor replication process and synchronizes if data updates. This model will help users to select favorable replication mechanism according to their replication requirements and concrete operational conditions.

According to Correia *et al.*, [4] databases do not provide enough support for third party replication, therefore, it compels either to modify or develop a server in middleware (server wrapper) that will capture client demands. But it is quite difficult to modify server and port it because server source code is not available to a normal user or developers. If the middleware wrapper works then it can easily redirect the client requests to its designated underlying RDBMS which results in performance overhead because it introduces an additional communication step and it also needs a large development effort. This problem is addressed by GAPI (GORDA application programming interface) which helps in the implementation of different replication strategies and deploy them in RDBMS. It is cost effective and enables reuse of replication protocols and components and it also allows close coupling with RDBMS intervals. This paper addresses the issues and proposes an interface and architecture (reflective), which represents transaction processing that by external replication protocol that can be modified and observed. It also tells how different replication algorithms are suitable to proposed Model. Secondly it also discusses that the interface is implemented on the architecture (Apache Derby, Sequoia server wrapper and Postgress SQL), which shows how cost effective and viable this approach is. Finally the Postgress SQL prototype is benchmarked using TPC-W industry standard and results shows that still it gives better results though the server architecture is different and miss-match.

In [5], Goel and Buyya state that replication is one of the known phenomena in which copies of data are stored at different locations. In a distributed environment through replication technique data is accessed efficiently. Replication provides data consistency and availability each time without bothering failure of any site because of its data replicas. If any request of data access is originated, it finds its closely located replica which increases performance of system. In case if the query is read only then replication increases performance but if the query needs some update request process then performance is not that much achieved because it also has to maintain consistency between the replicas. With the passage of time nature of computing is getting complex which creates challenging problems (Replication Environment). Quorum Based replication is also one of the protocol that replication uses. In this case a small subset of replica is updated and assigned a vote of non-negative. Vote assigned is based on certain threshold on read and write operation. Non quorum has guarantee of at least one data item (copy) between the read (quora) and write (quora), which avoids read/write or write/write conflicts. Synchronous Replication and Asynchronous Replication are the types of replication protocols. In synchronous systems before the transaction commits all the replicas are updated. Updates made to server data is similar to the changes made at each replica within the same time generating serialize able schedule throughput the domain. This paper gathers and discusses multiple replication strategies that play a vital role in different architectures. It also helps people working in different domains, to use replication strategies discussed in distributed systems. It also presents a comparative study of different existing replication theories and shows in a certain domain, which theory best suits.

Daudjee *et al.*, [6] shows how global SI in lazily synchronized replicated databases system takes advantage of local snapshot Isolation concurrency control ensures the order of update transaction. Start and commit transaction schedule is captured and concurrency control of transaction is make sure which took place at one site and these updates are installed at the replicated systems. One of the key draw back in global SI is that a client cannot see its own updates. Strong snapshot Isolation gives transactional guarantee. This paper discuss that a lazy replicated which guarantees global strong SI is costly and it cannot stop transaction inversion, whereas strong session SI within the client session can prevent transaction inversion but does not across session. In weak SI if data is fully replicated on secondary sites. Clients usually connect to the one of secondary site and request for a transaction. From the client request it is made clear that which transaction is read only and which transactions need to update the data. Those transactions which are read only, they are executed at the secondary site and secondary site forwards update transactions to the primary site and they executes there. Updating made at primary site is forwarded to secondary sites and they update them accordingly. Using a simulation model an experiment is conducted to find strong-session-SI algorithm effectiveness with respect to transaction response time and throughput. Two more algorithms are used to find the comparison between strong SI. Experiment showed that algorithm weak snapshot isolation simply forwards the transactions to primary for execution which shows global weak snapshot isolation where secondary site executes read only transaction. This paper discusses an algorithm and an architecture which in lazy replicated systems maintains global weak snapshot isolation by taking advantage of local concurrency controls. Strong session SI shows that how it can be used to prevent transaction in lazy replicated database systems.

Sears *et al.*, [7] propose a database replication engines which provides high throughput regardless of database size and query content. Three components are examined in order to perform lookups. In order to find match it starts from the in memory component and it moves to a larger tree (out of date) till match found. Two on disk lookups are involved till a merge is going on and there is much more probability of three lookups. The first contribution of Rose is the use of compression in order to increase throughput (merge). The number of sequential I/O that a merge required is reduced by compression techniques and it deals extra computational power for limited storage bandwidth. The number of searches which a storage system provides and page cache hit ratio determines the lookup performance and page cache effective size is increased by Rose's compression. Secondly, the application of LSM trees reduces workload of data replication to a great extent. To take advantage of LSM trees (write throughput), replication environment should not compel Rose to get the overwritten values pre images. The operations of Rose are concurrent where the readers and writers working independently in order to avoid blocking live lock and deadlocks. Tuples are stored by Rose in a sorted format which simplifies the compression and offers a member of new optimization opportunities. The Rose's primary bottleneck is compression which reduces sequential I/O. This paper has also presented the using of snapshot consistency in order to give concurrency control for LSM trees. The new approach of database replication discussed in this paper provides strength of LSM-Trees by avoiding searches of indexes during updates.

Agrawal *et al.*, [8] discuss the mechanism of implementing view and indexes used in large scale distributed databases. In web applications one of the critical issues is minimizing the update latency for which proper maintenance of views and updates is very important. Keeping in mind and examining the design space, two types of implementation views are proposed and those are RVTs (Remote View Tables) and LVT (Local View Tables) which provides tradeoff between throughputs of system and minimizing view moldiness. This paper also discusses how to make such efficient table view, selection views, equijoin view and group by aggregate views. A model is also introduced in this paper which analyzes the consistency and it also helps developers to efficiently maintain views, and how these views helps in improving the evaluation cost of complex queries. The approach discussed is to defer expensive maintained of view till the completion of base updates and clients on their updates experience low latency. But maintenance of deferred view also introduces some challenges which are i) A scalable architecture must be developed in order to maintain querying views and store ii) It is not possible to abort already committed transactions of client which has updated the base table iii) Views must provide consistency guarantees that might be out of date in complex ways iv) As data is replicated to different data centers, so there is also need of efficiently replication of views. One of the benefits of PNUTS architecture is that of scalability and according to the need more servers are added. It discusses maintenance of an asynchronous view in a very large scale distributed database which is horizontally partitioned. The contribution made is as under; for the maintenance of deferred views, two mechanisms are introduced which are local tables and remote view tables. Secondly different kind of characters of such indexes and views are maintained by the mechanism. A consistent model is made for view which can maintain it asynchronously. Then experimental cost is evaluated about the local and remote view and for their maintenance and the benefits in query optimization.

Calvin (a fast distributed transaction system) proposed by Thomson *et al.*, [9] schedules the transaction. It is also a data replication layer that generates the ordering which prohibits content cost in distributed transactions. Calvin also supports like disc based storage, scales nearly and it also does not have single point of failure. Calvin also supports consistency levels (multiple) by replicating the inputs of transactions. It also supports consistency to the distant replicas which are located far away geographically and transaction throughput remains the same. Calvin is designed to provide full ACID (Atomicity Consistency Isolation Durability) transactions. It runs alongside a storage system which is non-transactional that shares nothing and it is near linearly, which provides high availability. Calvin provides a layer which is above storage system and distributed transactions are scheduled by this layer. This layer also does the network communication and replication in the system. One of the major contributions of this paper is the design of layer which schedules the transactions and data replication. This layer transforms non transactional system (storage) into shared nothing database system (near linearly scalable) which provides strong consistency, high availability and ACID (full) transactions. Secondly the concurrency control protocol (deterministic) which is more scalable as compared to the previous approaches and it has data pre-fetching mechanism which plans before the transaction executes, that allows operation of transaction on disk residing data. Calvin determinism along with the fast check pointing scheme guarantees complete removal of physical redo logging and the overhead associated with it. In case if any hardware failure occurs, Calvin handles it by recovering the crashed machine from its most recent complete snapshot and replies to the most recent transaction.

Serrano *et al.*, [10] discuss the performance gain which is achieved by partial replication configurations is analyzed analytically, like the configuration of all sites which does not store all data. A partial replication protocol is also derived which provides 1-copy snapshot isolation, which is correctness criteria. Research done tries to overcome the limitations of scalability. 1-copy-serializability is the correct criterion that provides full database replication. Read/write conflicts occur in optimistic replication protocol which results in transaction abort whereas low concurrency is resulted by suspicious protocols. Snapshot Isolation controls the multi version concurrency and when the transaction starts a snapshot is generated. The solution provided by this paper helps in increasing the scalability (replication) issues. An analytical model presented helps in increasing the number of replicas in partially replicated databases. For partially replicated databases an eager replication protocol is proposed which is based on snapshot isolation. In this protocol different transactions at different sites are executed and protocols handle distributed transactions efficiently.

A partial replication protocol is introduced by Armendáriz *et al.*, [11] that provides a view (consistent) of database that provides adaptive replication technique which supports failures and recovery of replicas. Distributed databases provide services to a great number of users and become an attractive approach. A certification based technique is discussed in which each site itself determines whether the delivered transaction should be committed or aborted. If there does not exist any concurrent conflict in the system then the transaction is committed. It also discusses and algorithm for partial replication which is called SIPRe which provides Generalized Snapshot Isolation (GSI). Execution of distributed transactions is supported by SIPRe algorithm. Some operations of transactions are submitted to other replicas because partitioning is not done flawlessly accordingly.

A new workload aware approach for database replication and partitioning is presented by Curino *et al.*, [12] which improves scalability in shared-nothing databases (distributed). In OLTP (Online Transaction Processing) settings as distributed transactions are quite costly. The partitioner discussed in this paper reduces number of transactions in distributed systems which results in balanced partitioning. Schism has two phases. The first one is workload driven which is a graph based replication phase. It creates a graph (with a node per tuple) and between nodes there are edges through which the transaction accesses nodes and a balance partitioning is done by the partitioner which results in less number of cross partition transactions. The second phase is validation and explanation phase which uses machine learning techniques to find partitioning strategy explanation. Schism present in this paper is a partitioning system which is graph based and it does partitioning depending on the transactional workload. Graph presents database and workload where nodes represents tuples and edges represent transaction and graph partitioning algorithm is applied to find balanced partitioning. Schism also shows that it can also create replicas of records that are frequently updated, so in other words a portion of tables is also replicated by this approach. It does partitioning well if data sets are large and partitioning of millions of tuples is done in minutes.

Garcia *et al.*, [13] talk about BFT (Byzantine fault tolerance) database replication middleware (Byzantium) which gives semantics of snapshot isolation. BFT is the first database system which allows concurrent execution of transactions without relying on centralized component which provides better performance as well as robustness. Though this replication protocol gives moderate results on read-write workload but in case of read-only workload it gives much better results compared to a non-replicated database. This approach also works well in finding software bugs and tackles malicious attacks very responsibly. Another benefit is the concurrent execution of transactions without relying on the primary storage. In Byzantium there are different design features which are useful beyond database replication like snapshot isolation, replication based on middleware, execution of optimistic group of operations and striping with BFT replication. It also allows different vendors to implement their techniques in database replication environment without affecting or modifying them, which further improves the security checks. After evaluating Byzantium results it is showed that it gives 90% better results in read only workload compared to the database systems that does not provide replication.

Digital forensics also requires support for databases which need to be replicated for data integrity. Khan *et al.*, [14, 15] proposed a machine learning approaches for post-event timeline reconstruction. The proposed techniques however are based on static analysis of the data enclosures. Khan [16] suggests that Bayesian techniques are more promising than other conventional machine learning techniques for timeline reconstruction. Rafique and Khan [17] explored various methods, practices and tools being used for static and live digital forensics. Bashir and Khan [18] looked into triaging methodologies being used for live digital forensic analysis. Shahzad *et al.*, [19] use database support for protection against zero-day malware attacks. Khan *et al.*, [20] implemented storage pattern in the OR mapping framework, and Ali and Khan [21] proposed an ICT infrastructure framework for microfinance institutions. Shehzad and Khan [22] looked into the methods for integrating knowledge management with business intelligence processes. Umar and Khan [23, 24] suggested a fFramework to separate non-functional requirements. Khan and Khan [25] proposed an Internet content regulation framework by using a blacklist database.

3. Critical Analysis

The critical analysis of the data replication techniques of RDBMS is provided in Table I.

Table 1. Summary of Database Replication Techniques in RDBMS

Research Topic	**Author (s)**	**Strengths**	**Limitations**
Adaptive Distributed Database Replication Through Colonies of Pogo Ants	Abdul-Wahid *et al.*, [2]	It optimizes distribution of partially replicated databases in network systems which minimizes inter node communication and gives better response time and enables dynamic addition of replicas.	Though it is event driven system, but it does not synchronizes the learning process with events.
Transaction Based Grid Database Replication	Chen *et al.*, [3]	Existing grid mechanism is used to provide metadata registry and it defines high level APIs which manages transaction based replication across multiple domains and legacy software's are also moved to grid smoothly.	API of this framework is not generic enough to cope with systems from different vendors and optimization related to memory or processing is not discussed.
GORDA: An Open Architecture for Database Replication	Correia *et al.*, [4]	It provides third party replication introducing a middleware wrapper without modifying the database server. It does better solutions on PostgreSQL.	Some architecture does not provide expected results when the middleware interface is implemented on them like apache derby and sequoia.
Data Replication Strategies in wide area distributed systems	Goel & Buyya [5]	It gathers and presents different replication strategies in different architectural domains and recommends its applicability depending on the nature of domain (distributed, p2p, data grid, and www).	It gives a better replication comparison of distributed, p2p, data grid and www but does not recommend a specific technique that gives better results.
Lazy Database Replication with Snapshot Isolation	Daudjee *et al.*, [6]	Algorithm and architecture discussed in this paper prevents transaction inversion in lazy replicated database systems. It also provides better results on read only queries.	In case of read-write queries snapshot isolation takes too much time in transaction execution.
Rose: Compressed, log structured replication	Sears *et al.*, [7]	Rose is a database replication engines which provides high throughput regardless of database size and query content.It is particularly designed to run real time decision support and analytical processing queries.	Compression is the primary bottlenecks in Rose which reduces sequential I/O that results in less number of queries executed in a specific time.
Asynchronous View Maintenance forVLSD Databases	Agrawal *et al.*, [8]	This mechanism enables better Implementation and maintenance of views and indexes in massive scale databases and solves complex queries.	Maintenance of complex views and index requires resources (memory and processing) and in small scale databases it does not provide better results.
Calvin: Fast Distributed Transactions for Partitioned Database Systems	Thomson *et al.*, [9]	It is a data replication and transaction scheduling layer that works well in distant geographical replicas and has no single point of failure providing Atomicity, Consistency, Isolation and	As replicas are located at distant geographically areas so it does not perform well in HATs (Highly Available Transactions).

		durability to transactions.	
Boosting Database Replication Scalability through Partial Replication and 1-Copy-Snapshot-Isolation	Serrano *et al.*, [10]	This mechanism does better configuration of partially replicated data bases which gives better results providing scalability.	Probabilities of data miss at specific node increases because each systems contains partial replica of database.
SIPRe: A Partial Database Replication Protocol with SIReplicas	Armendáriz-Inigo *et al.*, [11]	A certification based technique enables each site to determine whether the delivered transaction should be committed oraborted.	If a site takes wrong decision regarding commits or aborts then inconsistency creates in the entire database.
Schism: a WorkloadDrivenApproach to Database Replication and Partitioning	Curino *et al.*, [12]	Schism is a graph based workload aware system which does partitioning based on transactional workload which reduces number of transactions in distributed systems.	Setting of schism in OLTP (Online transaction Processing) is costly and it is not effective in large datasets for analytical queries.
Efficient Middleware for ByzantineFault Tolerant Database Replication	Garcia *et al.*, [13]	BFT (middleware replication) is the first database system which allows concurrent execution of transactions without relying on centralized component that provides better performance as well as robustness.	This replication protocol gives moderate results on read-write workload.

4. Future Work

Most of the contemporary research on database replication highlights different models and techniques which are used to improve data replication in different systems. Each technique has its own advantages and limitations. As a future dimension to this research, we intended to propose a technique for database replication which could overcome performance issues. The focus would be to improve data replication in relational database management systems especially in distributed systems.

5. Conclusion

In this paper we have tried to present a review on database replication techniques used in RDBMS. This review can be useful to comprehend the pros and cons of the existing techniques while implementing a new technique in an effective manner keeping in mind limitations of different database replication techniques. Based on their utility and efficiency (throughput and response time) the merits of existing techniques are also critically analyzed.

References

[1] M. Khan and M. N. A. Khan, “Exploring Query Optimization Techniques in Relational Databases”, International Journal of Database Theory & Application, vol. 6, no. 3, **(2013)**.

[2] S. Abdul-Wahid, R. Andonie, J. Lemley, J. Schwing and J. Widger, “Adaptive Distributed Database Replication Through Colonies of Pogo Ants”, In Parallel and Distributed Processing Symposium, 2007.IPDPS 2007. IEEE International, **(2007)** March, pp. 1-8, IEEE.

[3] Y. Chen, D. Berry and P. Dantressangle, “Transaction based grid database replication”, In Proceedings UK e-Science All Hands Meeting, **(2007)** July, pp. 166-173.

[4] A. Correia, J. Pereira, L. Rodrigues, N. Carvalho, R. Vilaça, R. Oliveira and S. Guedes, “GORDA: An open architecture for database replication”, In Network Computing and Applications, 2007 Sixth IEEE International Symposium on NCA, **(2007)** July, pp. 287-290, IEEE.

[5] S. Goel and R. Buyya, “Data replication strategies in wide area distributed systems”, Enterprise Service Computing: From Concept to Deployment, vol. 17, **(2006)**.

[6] K. Daudjee and K. Salem, "Lazy database replication with snapshot isolation", In Proceedings of the 32nd international conference on Very large data bases, **(2006)** September, pp. 715-726, VLDB Endowment.
[7] R. Sears, M. Callaghan and E. Brewer, "Rose: Compressed, log-structured replication", Proceedings of the VLDB Endowment, vol. 1, no. 1, **(2008)**, pp. 526-537.
[8] P. Agrawal, A. Silberstein, B. F. Cooper, U. Srivastava and R. Ramakrishnan, "Asynchronous view maintenance for VLSD databases", In Proceedings of the 2009 ACM SIGMOD International Conference on Management of data, **(2009)** June, pp. 179-192, ACM.
[9] A. Thomson, T. Diamond, S. C. Weng, K. Ren, P. Shao and D. J. Abadi, "Calvin: fast distributed transactions for partitioned database systems", In Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data, **(2012)** May, pp. 1-12, ACM.
[10] D. Serrano, M. Patiño-Martínez, R. Jiménez-Peris and B. Kemme, "Boosting database replication scalability through partial replication and 1-copy-snapshot-isolation", In Dependable Computing, 2007 13th Pacific Rim International Symposium on PRDC, **(2007)** December, pp. 290-297, IEEE.
[11] J. E. Armendáriz-Inigo, A. Mauch-Goya, J. R. de Mendívil and F. D. Muñoz-Escoí, "SIPRe: a partial database replication protocol with SI replicas", In Proceedings of the 2008 ACM symposium on Applied computing, **(2008)** March, pp. 2181-2185, ACM.
[12] C. Curino, E. Jones, Y. Zhang and S. Madden, "Schism: a workload-driven approach to database replication and partitioning", Proceedings of the VLDB Endowment, vol. 3, no. 1-2, **(2010)**, pp. 48-57.
[13] R. Garcia, R. Rodrigues and N. Preguiça, "Efficient middleware for byzantine fault tolerant database replication", In Proceedings of the sixth conference on Computer systems, **(2011)** April, pp. 107-122, ACM.
[14] M. N. A. Khan, C. R. Chatwin and R. C. Young, "A framework for post-event timeline reconstruction using neural networks", digital investigation, vol. 4, no. 3, **(2007)**, pp. 146-157.
[15] M. N. A. Khan, C. R. Chatwin and R. C. Young, "Extracting Evidence from Filesystem Activity using Bayesian Networks", International journal of Forensic computer science, vol. 1, **(2007)**, pp. 50-63.
[16] M. N. A. Khan, "Performance analysis of Bayesian networks and neural networks in classification of file system activities", Computers & Security, vol. 31, no. 4, **(2012)**, pp. 391-401.
[17] M. Rafique and M. N. A. Khan, "Exploring Static and Live Digital Forensics: Methods, Practices and Tools".
[18] M. S. Bashir and M. N. A. Khan, "Triage in Live Digital Forensic Analysis", International journal of Forensic Computer Science, vol. 1, **(2013)**, pp. 35-44.
[19] A. Shahzad, M. Hussain and M. N. A. Khan, "Protecting from Zero-Day Malware Attacks", Middle-East Journal of Scientific Research, vol. 17, no. 4, **(2013)**, pp. 455-464.
[20] M. N. A. Khan, A. Shahid and S. Shafqat, "Implementing a Storage Pattern in the OR Mapping Framework", International Journal of Grid & Distributed Computing, vol. 6, no. 5, **(2013)**.
[21] S. S. Ali and M. N. A. Khan, "ICT Infrastructure Framework for Microfinance Institutions and Banks in Pakistan: An Optimized Approach", International Journal of Online Marketing (IJOM), vol. 3, no. 2, **(2013)**, pp. 75-86.
[22] R. Shehzad, A. Khan and M. Naeem, "Integrating Knowledge Management with Business Intelligence Processes for Enhanced Organizational Learning", International Journal of Software Engineering & Its Applications, vol. 7, no. 2, **(2013)**.
[23] M. Umar and n. A. Khan, "Analyzing Non-Functional Requirements (NFRs) for software development", 2011 IEEE 2nd International Conference on Software Engineering and Service Science (ICSESS), **(2011)** July, pp. 675-678, IEEE.
[24] M. Umar and M. N. A. Khan, "A Framework to Separate Non-Functional Requirements for System Maintainability", Kuwait Journal of Science & Engineering, vol. 39, no. 1B, **(2012)**, pp. 211-231.
[25] A. A. Khan and M. Khan, "Internet Content Regulation Framework", International Journal of U-& E-Service, Science & Technology, vol. 4, no. 3, **(2011)**.