In-Memory Data processing using Redis Database

Gurpreet Kaur Spal
Department of Computer Science and Engineering
Baba Banda Singh Bahadur Engineering College,
Fatehgarh Sahib, Punjab, India

Jatinder Kaur
Professor
Department of Computer Science and Engineering
Baba Banda Singh Bahadur Engineering College,
Fatehgarh Sahib, Punjab, India

ABSTRACT
In present, in-memory data processing is becoming more popular due to examine a huge amount of information in shorter duration of time. Previously all servers utilize their own particular memory which is time consuming. To resolve this problem by using distributed cache, servers using cache memory for storing and retrieving data frequently. In present Big data processing, in-memory enumerate has become famous due to increase capacity and high throughput of main memory. Both relational and NoSQL databases are in-memory database that provides different mechanism for data storage and retrieval. In this paper, they make use of an in-memory key-value data storage system is Redis which works on a large data. Also, Redis server makes use of cache memory for increasing scalability and high throughput of main memory. Redis database helps in getting data from applications more frequently which improves the system performance as compared to relational database.

General Terms
Relational database, NoSQL, Data Processing

Keywords
Key-Value Stores, Redis, Document Stores, Column-Family Stores, Graph databases.

1. INTRODUCTION
In recent period, in-memory data processing is becoming more and more valuable as it is essential to examine a huge amount of information in shorter duration of time. A previous Client Server framework gives poor execution on read and write process regarding throughput and latency since all servers utilize their own particular memory to deal with the whole procedure, which is time consuming. To resolve this problem by using distributed cache, servers using cache memory for storing and retrieving data frequently. In-memory data processing systems primarily focus on those objectives which gives help to information processing. In present Big data processing, in-memory enumerate has become popular because to increase capacity and high throughput of main memory. Both relational and NoSQL databases are in-memory databases[1] that provides different mechanism for data storage and retrieval.

Relational database store data in structure like tabular format, where each relational table consists of rows (tuples) and columns, therefore it depends on the relational model. In the past, relational database[2] were popularly used for storing information like business documents, financial information, government records, personal data and so on. Relational database systems do not support unstructured data and do not scale easily[3]. Data can be retrieving from relational database using structure query language (SQL). Key features of relational database systems organized data into relations and provide ACID (Atomicity, Consistency, Isolation and Durability) transactional properties.

Numerously recent applications that rely on storing and processing large amount of information, wants high availability and scalability which added more difficulties to relational database. Therefore an increasing number of companies have followed different categories of NoSQL data stores or non-relational databases, generally termed as NoSQL databases. NoSQL[2] is non-relational data storage system which does not require a fixed table schema, to replicate and distribute (partition) data over many servers. Today, NoSQL is used by large number of companies named as Adobe, Digg, Facebook, Foursquare, Google, Mozilla, etc.

2. TYPES OF NOSQL
According to NoSQL data model[4], the data stores are grouped into four categories are key-value data stores, document stores, column-family stores and graph databases.

2.1 Key-value stores
The key-value stores provide simple data structure and do not require any fixed schema, although they still face many problems such as single node failure, data inconsistency and so on. The system need to understand their design and implementation which helps in resolving these problems. Information in main memories is volatile so the system is unreliable because the information can lost due to unexpected system crash. Now to resolve this problem by data replication and traditionally, to ensure information prevention by saving information as image files or save onto disks. Data replication is that makes copies of data over different nodes to improve reliability of the system. Basically key-value stores are the most general categories of NoSQL database[5] that can store data in the form of key and value pairs in primary memory. Probably the most important key-value stores such as Redis[6], Riak, Scalaris, etc.

2.1.1 Redis
The one of the most popular used in-memory non-relational database is Redis[6], as an open source, single-threaded server[7], advanced key-value cache and store. Redis is blazing fast in speed as compared to the relational database. In Redis, the data processing time ranges in nanoseconds or milliseconds. The use of Redis is easier as compared to relational database. Redis database has many options of data storage like strings, lists, sets, hashes, sorted sets and some advanced functions including publish/subscribe, master/slave replication, disk persistence and scripting. Data stored in Redis as plain text and is not supported data encryption[8].

2.2 Document stores

The document stores contain collection of documents including JSON (JavaScript Object Notation), XML (Extensible Markup Language) and so on; can help in storing and retrieving documents[4]. No joins are available in document databases as compared to relational database. In general, it is used for storing and managing big data- size collections of complex documents. As compared to relational database, ACID transactional properties are not supported by document stores. There is some of the most important document stores are SimpleDB, CouchDB, etc.

2.3 Column family stores

The column family stores are very sparse which primarily work on columns where each column is treated independently. Column family stores are specifically useful in handling large amounts of information distributed over different nodes. These systems can easily replicating data to increase the number of nodes in the cluster. Most efficiently used column family stores are Cassandra, HBase, HyperTable, etc.

2.4 Graph databases

The graph databases[4] are category of the NoSQL database, stores data in the form of a graph. In graph databases, the graph is made up of two things: nodes act as the entities or objects and edges act as the relationship between the entities or objects. The graph databases are gaining attraction as they are currently being handled in organizations for managing information within applications like social networking applications[4], content management, security and access control, networking and cloud management etc. There are some of the graph databases are Neo4j, OrientDB, etc.

3. DESIGN AND IMPLEMENTATION

In this section, they will give detail about two proposed designs are **Read data from relational database** and **Read data from redis database**. They will show their details and how they are implemented in the following subsections.

3.1 Read data from relational database

In this subsection, they will give detail about the proposed design of "Read data from relational database" and also discuss the implementation steps of the following design in the flow chart.

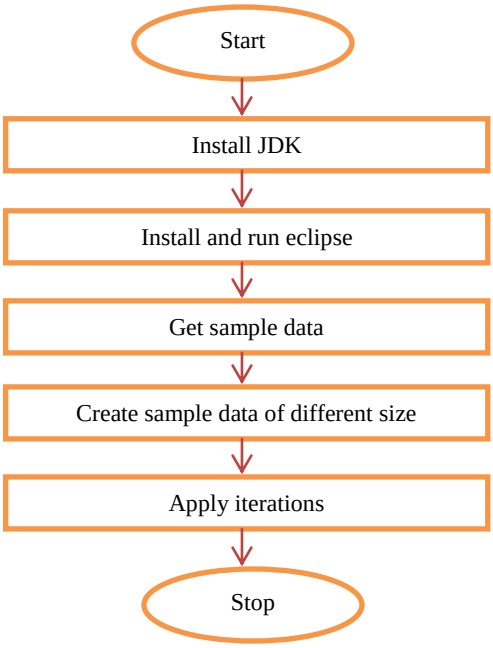


Fig 1: Flow chart of read sample data from relational database

Fig. 1 shows how this method works. Initially, they will install JDK (Java Development Kit) contains in addition the development tools to create java programs. Then install and run eclipse is an integrated development environment which may be used to develop in different programming languages. Then get sample data in CSV (Comma Separated Values) format which has a strict tabular structure. Creating sample data of different size by adding or deleting values from the sample data for testing the execution time. Finally apply iterations and calculate the execution time.

3.2 Read data from redis database

In this paper, they mainly consider the method which improves the system performance by reading data more frequently from applications and also saves time.

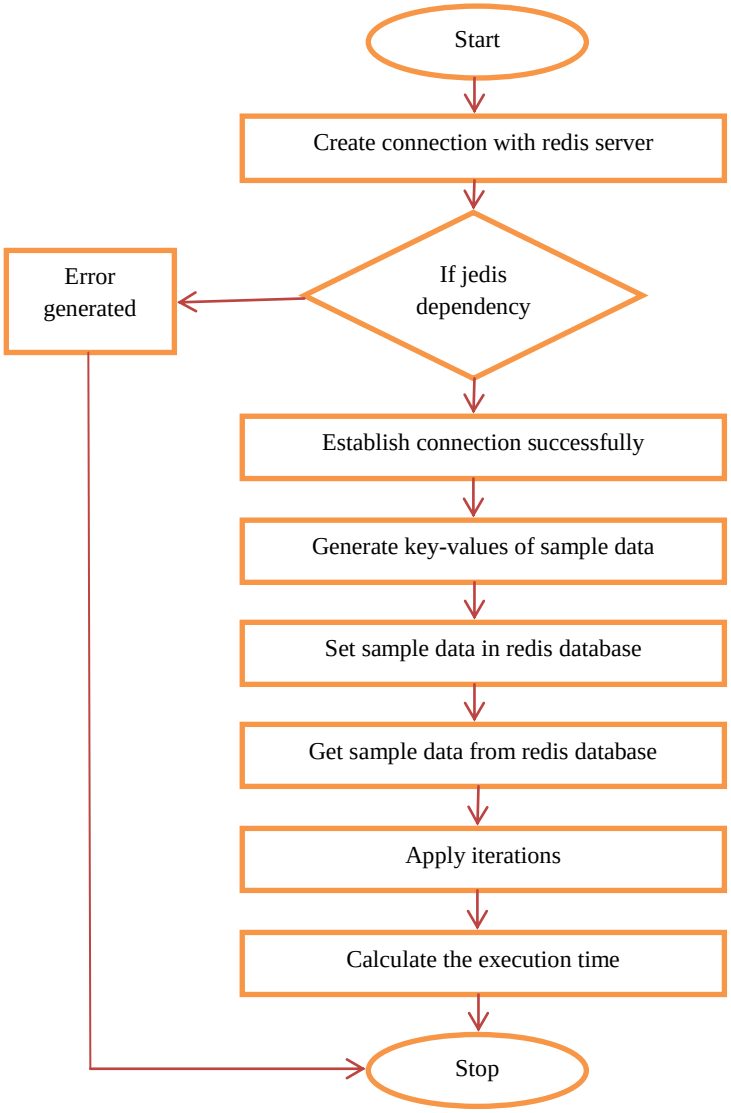


Fig 2: Flow chart of read sample data from redis database

Fig. 2 shows how to read sample data from redis database. Implementation of this method is developed in Java as code in the backend provide application program interface with the redis-client and redis server. If there is no jedis dependency then it gives error and stop. For redis, a client library in Java is termed as jedis, is driven by a key store data structure to persist data. By Adding jedis dependency that helps in making connection successful with redis database. Create sample data in key-value format and store into it. SET command sets the value stored at the given key. The redis database stored data in the form of key-value pairs but it can also view as JSON and plain text. GET command fetches the value stored at the given key. Get the sample data from redis database and then apply iterations where different sizes of sample data are getting (reading) from it. Finally, calculate the execution time and the formula for calculating the execution time is:

Execution time = End time – Start time

4. EXPERIMENTAL RESULTS

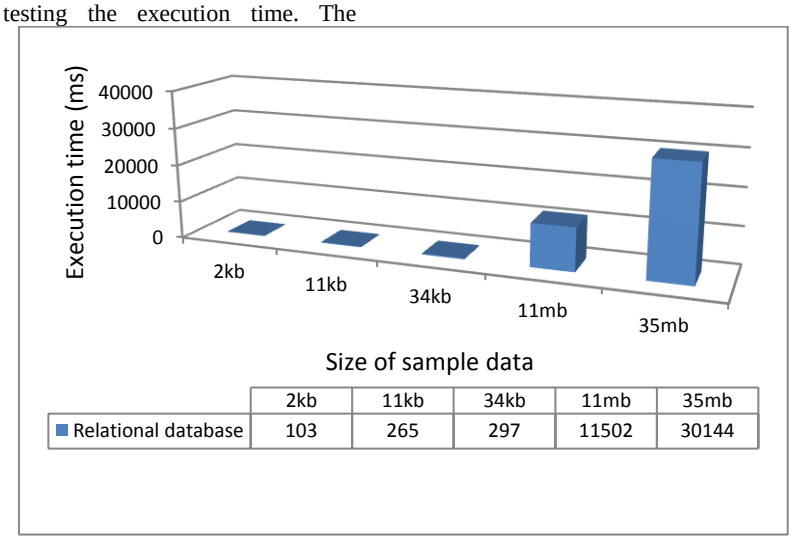
4.1 Test environment

In this experiment, the use Redis Desktop Manager 0.8.8.384 as a baseline for performance evaluation. The proposed methods are also implemented on this version of Redis Desktop manager. They use their own benchmark implemented in Java to test the performance of Redis database and relational database. The software requirements are: operating system (Windows (64-bit)), platform (Windows), and tool (Eclipse) and hardware requirements are Minimum Dual core processor operating at 3.6 GHz or above, 160 GB hard disk and RAM (2 GB or above).

4.2 Comparative evaluation of relational database and redis database on a single machine

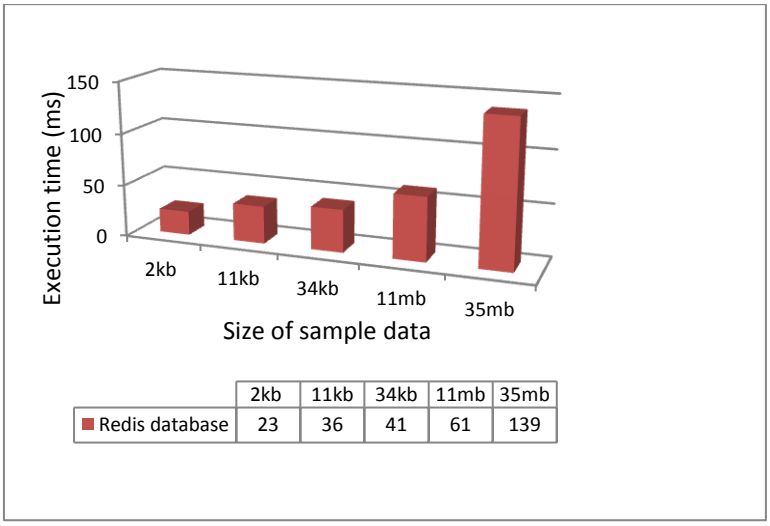
The performance of proposed work that read sample data from relational database has been evaluated the performances parameters: size of sample data and execution time. Fig 3

shows the execution of relational database in ms (milliseconds) and the sample data in CSV (Comma Separated Values) format which has a strict tabular structure. It also shows the sample data of different size for testing the execution time. Fig 4 shows the execution time in ms (milliseconds) of getting key-value data and the sample data of different size for testing the execution time. The comparisons of execution time of Relational database with Redis database as shown in Fig. 5. Redis database supports getting multiple values in a single command to speed up communication with the client libraries. A very high read speed is achieved by Redis database as compared to Relational database.



	2kb	11kb	34kb	11mb	35mb
Relational database	103	265	297	11502	30144

Fig 3: Time taken during reading of sample data from relational database



	2kb	11kb	34kb	11mb	35mb
Redis database	23	36	41	61	139

Fig 4: Time taken during reading of sample data from redis database

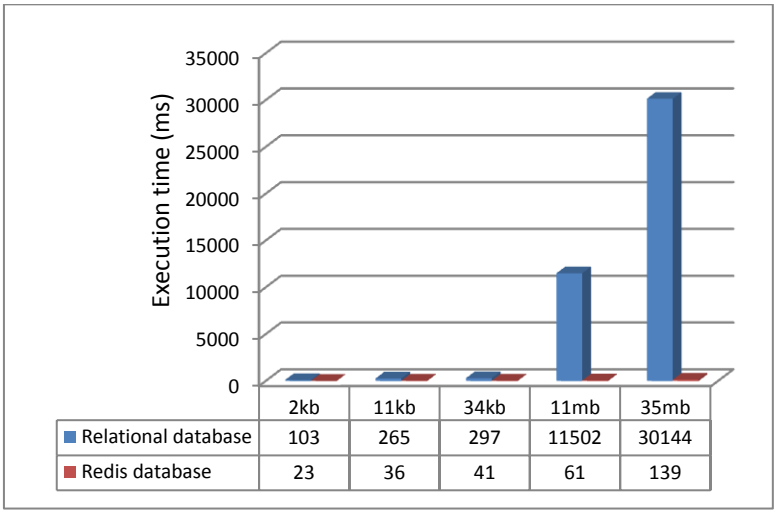


Fig 5: Compare the reading time of relational and redis database

5. RELATED WORKS

The qualitative comparison of in-memory data management systems[1] are Relational databases, NoSQL databases, Graph databases, Cache systems, Big data analysis systems and real time processing systems on multiple dimensions. The comparison of Relational and NoSQL databases on the basis of the security issues[2], where security is necessary today. NoSQL databases are very popular today because of their ability to support for structured and unstructured data and perform heavy write operations with low latency[3]. The feature analysis of different categories of NoSQL databases and selecting databases on the basis of query handling in social networks[4]. They has been examined that how much time taken by applications during inserting and reading operations. Rick Cattell have discussed a number of SQL and NoSQL data stores[5] on their data models with examples. J.L. Carlson proposed that how to use Redis[6] on the system and also explain simple interaction with it using example of key-value data storage system which is useful in solving real problems. A comparison between several NoSQL databases with comments and notes[9]; to offer high performance on the basis of their speed.

6. CONCLUSION

In this paper, they make use of an in-memory key-value data storage system is Redis which works on a large data. Redis is blazing fast in speed on reading process as compared to the relational database. The major problem is a previous Client Server framework gives poor execution on read and write process regarding throughput and latency since all servers utilize their own particular memory to deal with the whole procedure, which is time consuming. To resolve this problem by using distributed cache, redis server using cache memory for storing and retrieving data frequently. Also, Redis server makes use of cache memory for increasing scalability and high throughput of main memory. In Redis, the data processing time ranges in nanoseconds or milliseconds as shown in these experimental results.

The future work includes other NoSQL databases such as MongoDB, Riak, CouchDB and so on, will create cloud environment for reading and writing large amount of data. NoSQL databases will proceed to command and be adjusted to the necessities.

7. REFERENCES

[1] Zhang, H., Chen, G., Ooi, B.C., Tan, K.L. and Zhang, M., 2015. In-memory big data management and processing: A survey. IEEE Transactions on Knowledge and Data Engineering, 27(7), pp. 1920-1948.

[2] Mohamed, M.A., Altrafi, O.G. and Ismail, M.O., 2014. Relational vs. nosql databases: A survey. International Journal of Computer and Information Technology, 3(03), pp. 598-601.

[3] Jogi, V.D. and Sinha, A., 2016, March. Performance evaluation of MySQL, Cassandra and HBase for heavy write operation. In Recent Advances in Information Technology (RAIT), 2016 3rd International Conference on (pp. 586-590). IEEE.

[4] Mathew, A.B. and Kumar, S.M., 2015, August. Analysis of data management and query handling in social networks using NoSQL databases. In Advances in Computing, Communications and Informatics (ICACCI), 2015 International Conference on (pp. 800-806). IEEE.

[5] Cattell, R., 2011. Scalable SQL and NoSQL data stores. Acm Sigmod Record, 39(4), pp. 12-27.

[6] Carlson, J.L., 2013. Redis in Action. Manning Publications Co..

[7] Lubis, R. and Sagala, A., 2015, October. Multi-thread performance on a single thread in-memory database. In Information Technology and Electrical Engineering (ICITEE), 2015 7th International Conference on (pp.571-575). IEEE.

[8] Sahafizadeh, E. and Nematbakhsh, M.A., 2015. A Survey on Security Issues in Big Data and NoSQL. Advances in Computer Science: an International Journal, 4(4), pp. 68-72.

[9] Tudorica, B.G. and Bucur, C., 2011, June. A comparison between several NoSQL databases with comments and notes. In Roedunet International Conference (RoEduNet), 2011 10th (pp. 1-5). IEEE.

[10] Zaki, A.K. and Indiramma, M., 2015, March. A novel redis security extension for NoSQL database using authentication and encryption. In Electrical, Computer and Communication Technologies (ICECCT), 2015 IEEE International Conference on (pp. 1-6). IEEE.

[11] Chen, S., Tang, X., Wang, H., Zhao, H. and Guo, M., 2016, August. Towards Scalable and Reliable In-Memory Storage System: A Case Study with Redis. In Trustcom/BigDataSE/I SPA, 2016 IEEE (pp. 1660-1667). IEEE.

[12] Saad, W., Abidi, L., Abbes, H., Cérin, C. and Jemni, M., 2014, October. Wide Area bonjougrid as a data desktop grid: Modeling and implementation on top of redis. In Computer Architecture and High Performance Computing (SBAC-PAD), 2014 IEEE 26th International Symposium on (pp. 286-293). IEEE.

[13] Wu, X., Long, X. and Wang, L., 2013, December. Optimizing Event Polling for Network-Intensive Applications: A Case Study on Redis. In Parallel and Distributed Systems (ICPADS), 2013 International Conference on (pp. 687-692). IEEE.