

Neo4J

SOWMYA RAVI¹

¹ School of Informatics and Computing, Bloomington, IN 47408, U.S.A.

* Corresponding authors: sowravi@iu.edu.com

project-000, April 3, 2017

Neo4J is a graph database designed for fast data access and management. The data is stored in the form of nodes and relationships in Neo4J. The unique approach it takes to store data makes it far more efficient compared to relational databases when the number of relationships within the data increases. Moreover, it has the ability to store trillions of data entries in a compact manner. Neo4J comes along with Cypher, a highly readable querying language. The paper elaborates the clustering activities used by Neo4J to achieve distributed computing its uses [1].

© 2017 <https://creativecommons.org/licenses/>. The authors verify that the text is not plagiarized.

Keywords: Cloud, I524

<https://github.com/cloudmesh/classes/blob/master/docs/source/format/report/report.pdf>

INTRODUCTION

Certain problems present in the world cannot be solved by using relational databases. For e.g. a Social graph representing a the network of friends in a social networking website. In this case the number of relationships in the data is too extensive and the relational databases perform poorly. Graph data bases on the other hand make the task of storing huge amounts of data relatively simple and efficient. Neo4J is one such NoSQL, graph database which was developed to be used in the kind of problems mentioned before [2]. Fig.1 illustrates a simple social network graph.

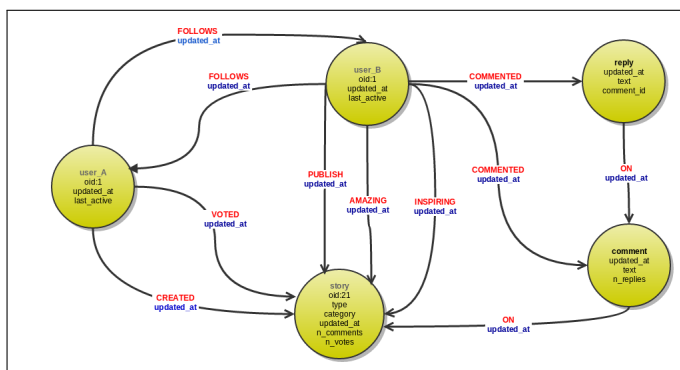


Fig. 1. A Social Network graph [3]

Neo4j is an open source data management software. At its core, Neo4j stores data in the form of nodes and relationships. It is often deployed in a production environment as a fault tolerant

cluster of machines. The high scalability and slow traversal times make it far more efficient than the conventional relational databases [1].

CYPHER PROGRAMING LANGUAGE

Neo4j uses its own programming language, Cypher, for data creation as well as querying. Cypher is capable of doing SQL like actions. In addition, it can specifically perform a powerful query called traversals. Traversal involves moving along a specific set of nodes in the database thereby tracing a path. This allows to leverage the spatial structuring of the data to get valuable information, similar to network analysis [4].

CLUSTERING FOR THE ENTERPRISE

This section discusses Neo4js architecture with respect to clustering. Neo4j uses clustering of machines to achieve high throughput, availability and disaster recovery [5]. Neo4j offers two kind of clustering

1. Causal Clustering
2. Highly Available clustering

Causal Clustering

The Causal clustering of machines in Neo4j is aimed at providing two important features [6]

1. **Safety:** The core servers of Neo4j ensure fault tolerance.
2. **Scalability:** Achieved using Read replicas that make massive scaling possible.

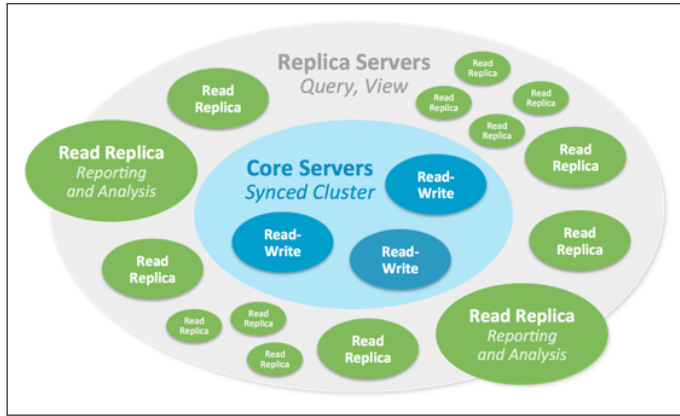


Fig. 2. Architecture of Causal Clustering [6]

The architecture of causal clustering is shown in Fig.2.

For operational purposes, the cluster is usually separated into two components: the core servers and the read replicas.

Core Servers

The Core servers are responsible for safe data storage. This is achieved by replicating all incoming queries/transactions using Raft protocol (A log replication protocol) [6]. The protocol ensures the durability of data before committing to the query request. Usually, a transaction is accepted only when a majority of the servers, calculated as $N + 1/2$, have accepted it. This number is directly proportional to the number of core servers N . Hence, as the number of core servers grows, the size of majority required for committing to an end user also increases. increases [6].

In practice few machines in the core server cluster is enough to provide fault tolerance. This number is calculated using the formula: $N = 2F + 1$ where N is the number of servers required to tolerate F faults [6]. When a core server suffers a large number of faults, it is automatically converted to a read-only server for safety purposes.

Read replicas

Read Replicas are Neo4j databases that scale out the incoming queries and procedures. They act like cache memories to the core servers which safeguard the data. Even though the read replicas are full-fledged databases, they are equipped to perform arbitrary read-only activities [6].

Read Replicas are created asynchronously by core servers through log-shipping [6]. Log shipping occurs when the read replicas poll the core servers for new transactions and the transactions are shipped from the core servers to the read replicas. This polling occurs periodically. Usually, a small number of core servers ship out queries to a relatively large number of read replicas, allowing a large fan out of workload thereby, achieving scalability [6]. The read replicas unlike the core servers do not participate in deciding the cluster topology.

Causal Consistency

In applications, data is generally read from a graph and written to a graph. In order to ensure the causal consistency in the data, the write operation must take into account previous write operations. The Causal Consistency model for distributed computing requires every node in the system to see causally related operations in the same order. This model ensures that the data can be

written to cores and the written data be read from read replicas. Fig.3 illustrates a Causal Cluster with causal consistency

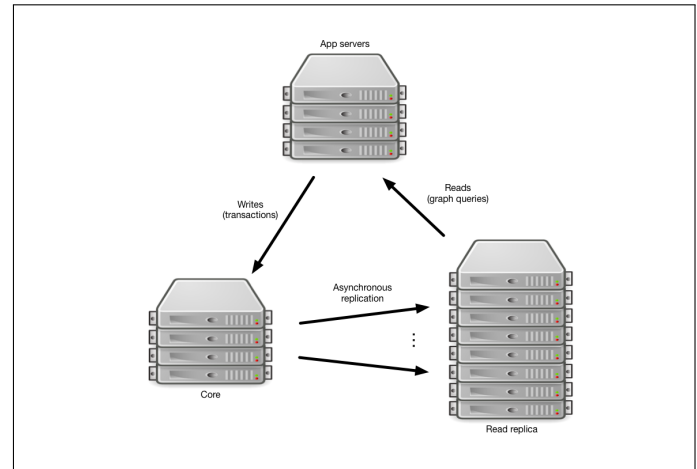


Fig. 3. Causal Cluster with causal consistency set via Neo4j drivers [6]

HIGHLY AVAILABLE CLUSTER

In this type of cluster each instance of the cluster contains full copy of the data in their local database. The cluster can be visualized as containing a single master with multiple slaves in which each instance is connected to every other instance (A 3 member cluster is shown in Fig.4.0)

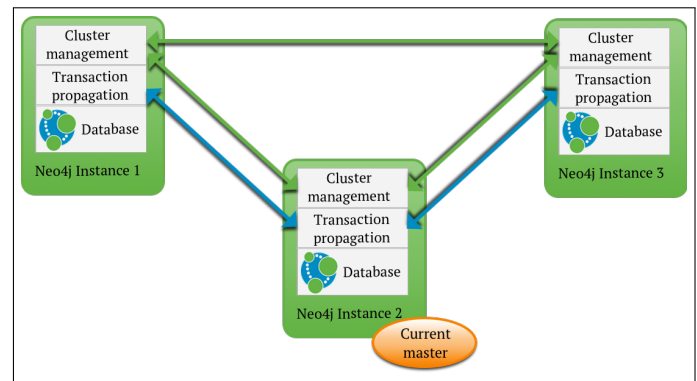


Fig. 4. A Highly Available cluster model [7]

Also, each instance contains the logic to perform read/write operations and election management [7]. Every slave, excluding the Arbiter instance periodically communicate with the master to keep databases up to date [7]. There is a special slave called the Arbiter explained in the following section.

Arbiter Instance

The Arbiter instance is a special slave that participates in cluster management activities but does not contain any replicated data. It simply contains a Neo4j software running in arbiter mode [7].

Transaction Propagation

Write Transactions performed directly on the Master will be pushed to slaves once the transaction is successful. When a write transaction is performed on a slave, the slave synchronizes

with the Master after each write operation. The write operation on slave is always performed after ensuring that the slave is synchronized with the Master [7].

Failover

When an instance becomes unavailable, it is marked as temporarily failed by other instances. If the Master fails then, another member in the cluster will be elected as the Master [7].

USE CASES

Some of the use case of Neo4j is given below [8].

- Fraud Detection
- Graph based search
- Network and IT operations
- Real-Time Recommendation system
- Social Network
- Identity and Access Management

Neo4j for Social Network Analysis

Social Networks are already graphs and several possible use cases for Social Networks are listed below [9].

- The Friends of Friends recommendation in social networks is one useful use case. The traversal capability makes this task simple and efficient.
- It can be used to discover previously unknown relationships in massive networks. People get connected through multiple channels. Neo4j may be of great help in studying these relationships
- Collaboration and Sharing become far more easier in the presence of graph databases. The clustering facility enables data being safe and secure yet highly available. Content visibility increases to a great extent.

CONCLUSION

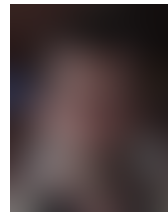
Neo4j being an open source, graph based and a highly scalable software, it is suitable for applications that deal with huge amounts of data. Also, Neo4j can be integrated with other tools and software such as Spark, Docker, Elastic Search, MongoDB etc. The versatility of Neo4j makes it a great software aid for data scientists trying to analyze relationships and networks in real time as well as in batch.

REFERENCES

- [1] Neo4j, "Chapter 1. introduction," Web Page, last Accessed: 03.24.2017. [Online]. Available: <https://neo4j.com/docs/operations-manual/current/introduction/>
- [2] S. Haines, "Introduction to neo4j," Web Page, last Accessed: 03.24.2017. [Online]. Available: <http://www.informit.com/articles/article.aspx?p=2415370>
- [3] M. project, "Social network project," Web Page, last Accessed: 03.24.2017. [Online]. Available: <https://meu-solutions.com/case-studies-social-network-project/>
- [4] R. Ostman, "Graphical database of citation network analysis," PDF Document, last Accessed: 03.24.2017. [Online]. Available: <http://webdocs.ischool.illinois.edu/crt/ostman.pdf>

- [5] Neo4j, "Chapter 4. clustering," Web page, last Accessed: 2017.02.24. [Online]. Available: <https://neo4j.com/docs/operations-manual/current/clustering/>
- [6] —, "Causal cluster," Web page, last Accessed: 2017.03.24. [Online]. Available: <https://neo4j.com/docs/operations-manual/current/clustering/causal-clustering/introduction/>
- [7] —, "Highly available cluster," Web page, last Accessed: 2017.03.24. [Online]. Available: <https://neo4j.com/docs/operations-manual/current/clustering/high-availability/architecture/>
- [8] —, "Graph database use cases," Web page, last Accessed: 2017.03.24. [Online]. Available: <https://neo4j.com/use-cases/>
- [9] —, "Solutions: Social network," Web page, last Accessed: 2017.03.24. [Online]. Available: <https://neo4j.com/use-cases/social-network/>

AUTHOR BIOGRAPHIES



Sowmya Ravi pursuing Masters in Data Science from Indiana University. Her research interests include Machine Learning, Data Mining and Big Data Analytics