Graph Storage and Analyzation in Big Data Scenarios

Bernd
Institute for Clarity in Documentation
webmaster@marysville-ohio.com

Florian Mihola
TU Vienna
e0304850@student.tuwien.ac.at

Lin Brookhaven Laboratories
Brookhaven National Lab
P.O. Box 5000
lleipuner@researchlabs.org

Patrick Saeuerl
TU Vienna
Bacc Studium
e1125492@student.tuwien.ac.at

ABSTRACT

The Abstract that we are going to write.

1. INTRODUCTION

The last years showed big changes for data management systems. The term "Big Data" has formed and relates to data that is too big to be processed and handled without the use of multiple machines. One of the driving factors for Big Data is social media. The amount of data produced by sites like Facebook and Twitter could not be handled anymore by traditional data storage systems and traditional data analysis methods.

We will explore why there has been a need for new data storage systems, leading to the "NoSQL" family of data storage systems and how data can be analysed in these systems. We therefore introduce the Map/Reduce model and newer distributed computing frameworks specialised for graphs. A last look will be on a project combining distributed graph storage and distributed computing.

2. BIG DATA

More and more data is produced every year. There are mainly two trends that bring up those problems: (need ref):

- The exponential growth of the volume of data generated by users, systems and sensors, further accelerated by the concentration of large parts of this volume on big distributed systems like Amazon, Google and other cloud services.
- The increasing interdependency and complexity of data accelerated by the Internet, Web2.0, social networks and open and standardized access to data sources from a large number of different systems.

The data generated by this systems is huge and is mainly characterized by the 5-Vs:

- Variety - describing the different data and data structures
- Velocity - the speed of data creation
- Volume - the amount of data
- Veracity - the reliability and trust in the data, also associated with quality of the data
- Value - the worth derived from the data

Each one of those V-s introduces challenges that traditional relational database management systems can not handle anymore. New databases were developed, broadly categorized as "NoSQL" systems that solve different issues for Big Data Management.

3. NOSQL OVERVIEW

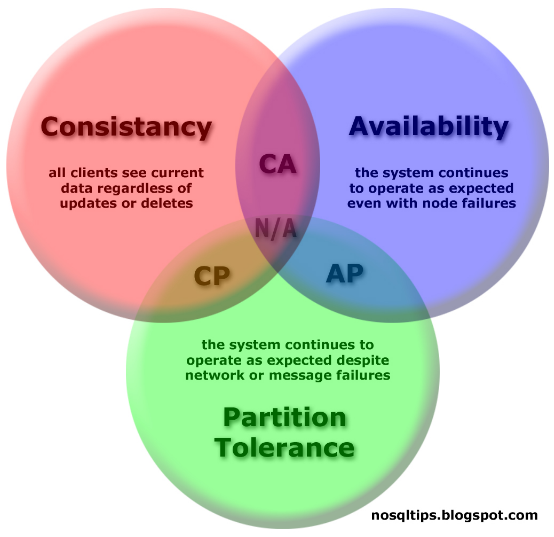
"NoSQL" stands for "Not only SQL" and describes a broad family of database management systems. A dominant feature of Big Data is the volume of data. It is too much for single systems and also too much for small clusters, therefore horizontal scaling is needed. Classical relational databases have problems with this because they focus on consistency and availability. The CAP Theorem [ref - j pokorny] basically says that you can choose two out of the following three: Consistency, Availablity, Partition Tolerance. Figure 1 illustrates this.

There are several different classifications of "NoSQL" databases. We will focus on and describe three groups: Document-Stores, Column-Stores and Graph Databases, as they have specific relevance to Big Data. Other database types exists like: Key-Value stores and XML-Databases but we will not focus on them here. ....[insert ref] has a good description of these types too.

3.1 Column-Stores

In column stores, the data is basically structured in columns which represent a single unit of data identified by a key and value. Those columns are part of super columns that build together a unit of information. Super columns are then part of a column-family. A column family is similar to a table in the relational schema. Due to their storage system, they fit well to distributed computing. Googles BigTable and HBase, the open source implementation of BigTable were the first of this kind of stores. They got to fame due to their usage with Map/Reduce algorithmns.

Map/Reduce is a system for expressing parallel batch systems. It consists of a series of map and reduce cycles. The map function



Consistancy
all clients see current data regardless of updates or deletes
CA
Availability
the system continues to operate as expected even with node failures
N/A
CP
AP
the system continues to operate as expected despite network or message failures
Partition Tolerance
nosqltips.blogspot.com

Figure 1: Figure 1: CAP Theorem, Source: blog.nosqltips.com

structures the data into a collection of key-value pairs. Those collections are then aggregated by their key into bins where each bin is processed with the reduce function. The popular open-source implementation of this approach is Apache Hadoop.

A lot of the scenarios in Big Data analysis feature graph analysis. The Map/Reduce paradigmn is not very suitable for this kind of analysis and is out-performend by second generation frameworks like Google Pregel or Giraph [An Evaluation Study of BigData Frameworks for Graph Processing]. We will come back to those frameworks in the graph analysis section.

3.2 Document Stores

This kind of NoSQL Databases consider documents as their unit of information. A document can be seen as a record in an RDBMS, but they are schemaless. This gives them more flexibility and they do not include null values. Therefore those systems are very well suited systems that have unstructured data, so they tackle the issue of variety in Big Data. The documents are normally stored in JSON or XML. Usually, document stores are fully searchable. A typical element of a document store looks like this:

```
{
        FirstName: "Jonathan",
        Address: "15 Wanamassa Point Road",
        Children: [
                {Name: "Michael", Age: 10},
                {Name: "Jennifer", Age: 8},
                {Name: "Samantha", Age: 5},
                {Name: "Elena", Age: 2}
        ]
}
```

Prominent members for document stores are MongoDB and CouchDB. MongoDB also offers support for Hadoop, therefore suits well for Map/Reduce applications. They both also provide good horizontal scaling.

An example of a document store in the Big Data scenario was done by[Insert name of Crime Paper]. They have utilized MongoDB to caputre unstructured data from various ressources likes blogs, social media and RSS feed. Afterwards, they classified and analyzed the data to predict areas which have high crime rate.

Because of the usage of document stores in Big Data scenarios, data mining algorithms start to adapt to those databases. For example [...] have created a service where topics and terms can be mined out of documents. This shows the wide adoption of this kind of storages.

3.3 Graph Databases

The dominant feature of graph databases is, that they represent their data in an actual graph. The importance is here on the relationship between the data. The nodes and edges can have properties, describing the node or the relationship between the nodes. By the way the data is stored internally, they perform much better than relational databases on graphs, although there has been research on how relational databases could handle graphs better [add ref].

When thinking of social media like Facebook, Xing and LinkedIn, one can see very easily how it would feel natural to model the data as graphs. Users in those systems represent nodes and the edges between them if they are friends, for example.

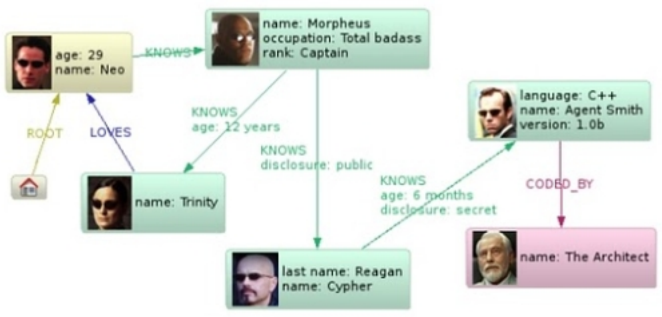


Figure 2: a social graph, Source: /www.infoq.com/articles/graph-nosql-neo4j

Besides representing social-graphs, they have also use in scientific computing. A request for databases optimized for graphs dates back to 1995 as they can represent genomes easier [reference to genome]. Another nice example for the use of graph databases is in the support of cloud management. [Soundararajan and Kakaraddi, needs ref] used a graph database to support system admins with managing their cloud environment by mapping their virtual infrastructure to a graph.

A popular example of graph databases is Neo4J. It can be queried with a variaty of query language such as: SPARQL, Gremlin, native Java API and CYPHER. A good example showing the expressiveness of CYPHER is from [Soundararajan and Kakaraddi, needs ref].

```
START a=node:indexName(name="A")
MATCH (a)-[*1..3]-(b)
WHERE b.node_type = "B"
RETURN distinct b;
```

This cypher query finds all connections between a and b with three hops or less. In SQL this would need at least three joins or other more complicated mechanisms.

4. GRAPH ANALYZATION IN BIG DATA

[Most from An Evaluation Study of BigData Frameworks for Graph Processing] We now know what Big Data is and what approaches we have for storing the data. We have now a closer look on frameworks that help us analyze data in Big Data scenarios and at last we have a closer look on a setup that utilizes a graph-database as storage and analysation frameworks.

4.1 Frameworks

The first framework for big data analization was Map-Reduce from Google back in 2004, which is publicly avaiable als Apache Hadoop. People started experimenting with it and applying it to various problems, like analyzing very large-scale graphs. Improvements have been made by projects like Stratosphere which basically introduces execution-plan improvement of Map-Reduce jobs. Still, Map-Reduce was initially designed for much simpler jobs and has problems with more complicated tasks. [ref 16 from apper] provides an overview of problems that Map-Reduce has with graph analyzation. This lead to the development of new frameworks tailored specifically for graphs based on the paradigma of BSP (bulk synchronization processing).

Pregel is a framework for BSP and is developed by Google. There exist two open source implementations of this by Apache named: Hama and Giraph. Giraphs purpose is the integration to other Apache products and Hama provides the processing API. They both are built on top of Hadoop.

GraphLab is another interesting framework based on the gather-apply-scatter pattern. The execution takes place on the nodes, where first the data from the neighbours is gathered. The calculation then takes place in the apply phase and the results can be scattered to the neighbours again.

[An Evaluation Paper] compared the above frameworks in performance. Except for using the Hadoop-File-System, they do not state what storage system they use in their test. This leaves open space for improvement which will be seen in the next section. Their result show an improvement of the speed in the new introduced frameworks Hama/Giraph and GraphLab. Due to ease of programming they recommend the frameworks based on Pregel, although GraphLab has extremly good results.

There is still room for improvement as shown by: [Cite Kylin Project]. There is currently an Apache Incubator project named Kylin [cite] which is not based on the referenced paper and should not be confused with. The Kylin proposed by ... is basically improving the storage system for a Pregel based processing system called GoldenOrb. As storage, they use HBase and improve it using the following techniques: vertex-weighted partitioning, Pull messaging and lazy vertex loading. Their results show that they outperform Hama and Giraph.

4.2 Graph Storage with BSP

The frameworks introduced do not utilize graph-databases as their storage, although they are tailored for graphes. With a special focus on social networks [Li yung ho] developed a system that utilzes a BSP framework backed up by a graph database to form a two-layer distributed graph database.

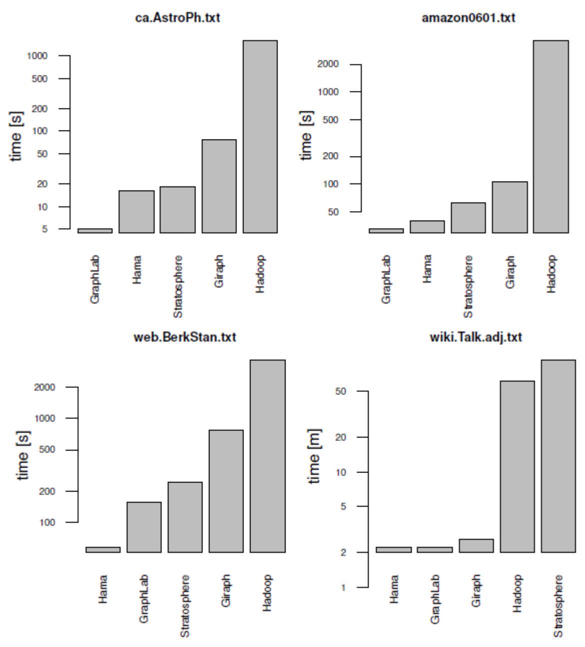


Figure 3: Results of evalution, Source: [REF: Cite Paper here]

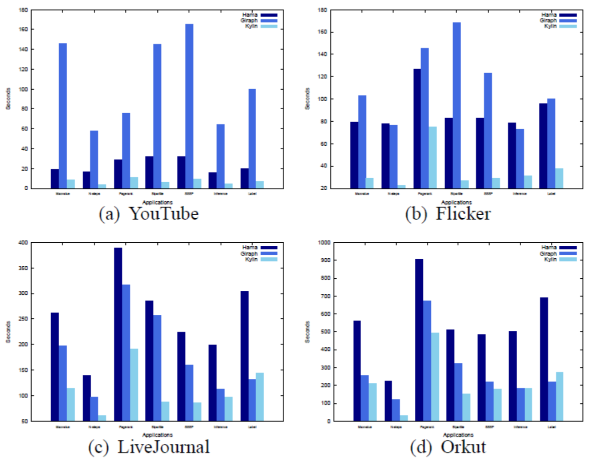


Figure 4: Results of evalution, Source: [REF: Cite Paper here]

The upper layer of their system consists of the graph processing engine. They have choosen GoldenOrb, an open-source implementation of Pregel which is normally based on the Hadoop File System. They studied which graph-system they can use as the storage for their application and have choosen Neo4J. They extended GoldenOrb so that it utilizes Neo4J as the storage system.

In their study, they show that Neo4J outperforms the Hadoop File System when extracting graph data, which is essential for every analyzation algorithm. They also compare different graph-partitioning algorithms and show their performance on social media subsets.

With the results of improved peformance at graph-extraction and the good results of BSP systems, this kind of architecture looks very promising. It would have been interesting to compare this impelmentation to Kylin as they are based on the same system with different storage. It is to note that this approach described here is older than Kylin.

5. CONCLUSIONS

Basically: Big Data is growing, new requirements - NoSQL helps, Engines(BSP) too. Space for specialization as seen by the GoldenOrb/Neo4J Architecture. More Frameworks - evaluation and specilization can be expected in the future.

6. REFERENCES

[1] M. Bowman, S. K. Debray, and L. L. Peterson. Reasoning about naming systems. *ACM Trans. Program. Lang. Syst.*, 15(5):795–825, November 1993.

[2] J. Braams. Babel, a multilingual style-option system for use with latex's standard document styles. *TUGboat*, 12(2):291–301, June 1991.

[3] M. Clark. Post congress tristesse. In *TeX90 Conference Proceedings*, pages 84–89. TeX Users Group, March 1991.

[4] M. Herlihy. A methodology for implementing highly concurrent data objects. *ACM Trans. Program. Lang. Syst.*, 15(5):745–770, November 1993.

[5] L. Lamport. *LaTeX User's Guide and Document Reference Manual*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1986.

[6] S. Salas and E. Hille. *Calculus: One and Several Variable*. John Wiley and Sons, New York, 1978.