Programming Paradigms for Big Data

Surajnath Sidh* & Saumya Gupta#

Introduction

We have entered the Era of Big Data. Big data basically refers to huge volume of data that cannot be processed and stored using the traditional approach within the given time frame. We constantly produce a lot of data. For example, via social media, public transport and GPS. But it goes way beyond than that. It has been reported on some of the popular Tech blogs that daily we upload around 55 million pictures, 340 million tweets and 1 billion documents. In total we produce approximately 2.5 quintillion bytes a day! [9] This data is gathered from our devices, computers and smart phones that collect and transmits information on what we do. That's a lot of data if we talk about what we have produced in previous years. In fact, according to Bernard Marr, a best-selling author and consultant in big data, if we gather all the data from the beginning of time until the year 2000, it would be less than what we now create in a minute. [9]

Therefore, it becomes a challenging task to handle that is, to manage and process this huge volume of data quickly and efficiently using traditional approach. Hence, we need programming paradigms that combine the scalable use of hardware with data processing and analysis. No single programming paradigm can serve all problem fields, so we can expect many alternatives. [11] To process Big Data, we don't need huge computers. People work with the cloud and endless network of normal servers and powerful algorithms. This way they can analyse over a million pieces of data in minutes.

Businesses are already using Big Data to better understand and predict customer behaviour and optimise and improve business processes. [10] Your business needs to consider Big Data handling too or risk being left behind. For example video streaming website like Netflix analyse the Big Data of their viewers like popular shows and watching series hence producing successful series with the perfect combination of actors, directors and storyline. [9] The Big Data of traffic is being analysed to develop a car that drive completely accident free all by itself. [9] These are just one of the few applications.

In this paper, we'll discuss various techniques that manages very large data flows with efficiency and opportunities and challenges related to it. We will also present you with modern emerging paradigm and frameworks for Big Data handling. MapReduce, Google's computing model, motivated the scientists to think again how they should handle the data of large sizes. Today Google's BigQuery, DataFlow and MapReduce, Apache Software foundation Hadoop, Microsoft's Dryad, are examples of powerful computing tools.

Challenges to solve the problem

We are facing an imminent torrent of machine generated data, creating volumes that will break the back of conventional hardware and software architectures. It is no longer be feasible to move the data to the compute process – the compute process has to be moved to the data.
-Mike Hoskins, Actian Chief Technology Officer. [1]

Processing Challenge-

With the rapid growth of emerging applications like social network, semantic web, sensor networks and LBS (Location Based Service) applications, the variety of data to be processed is increasing quickly. The existence of connections between data on the other hand leads to increased complexity of datasets. Effective management and processing of large-scale data thus, poses an interesting but critical challenge. Hence, extensive parallel processing is essential to perform on a massive volume of data in a timely manner. In contrast, the use of distributed techniques and algorithms is the key to achieve better scalability and performance in processing big data. [2] At present, there are a lot of popular parallel and distributed processing models, including MPI, General Purpose GPU (GPGPU), MapReduce and MapReduce-like. [2] We will focus on the MapReduce and its challenges in the next paragraph. Many researchers have suggested that present commercial Database Management Systems (DBMSs) are not suitable for processing extremely big data. So, there is no perfect big data management solution yet.

MapReduce proposed by Google, is a very popular big data processing model that has rapidly been studied and applied by both industry and academia. [3] MapReduce has two major advantages: it hide details related to the data storage, distribution, replication, load balancing and so on.[6] Furthermore, it is so simple that programmers only specify two functions, which are map function and reduce function. But at the same time, it is also criticized as a "major step backwards" compared with DBMS.[4] MapReduce tasks must be written as acyclic dataflow programs, i.e. a stateless mapper followed by a stateless reducer, that are executed by a batch job scheduler.[7] This paradigm makes repeated querying of datasets difficult and imposes limitations that are felt in fields such as machine learning, where iterative algorithms that revisit a single working set multiple times are the norm.[7] Therefore, MapReduce systems are inefficient in utilizing computing resources.

Storage Challenge-

Another big problem of its own — where do we put it all? Data is increasing exponentially, but storage technologies aren't necessarily keeping up.

Hard drive capacity isn't keeping pace with the current level data increases. Predictions says that global data will increase by a factor of 50 by 2020, but hard drives are only predicted to grow 15-fold in that same timeframe.[5] Using Solid-state drives (SSD) is both impractical and cost-prohibitive. Data deduplication and otherwise deleting unnecessary data might work — but first it will require better algorithms for calculating which data is "useful" and which is not. Outsourcing some or all of their data to the cloud is also an option, but cloud provider data centres still need to purchase and maintain sufficient storage resources to handle the increase in data computing needs. [5]

Companies need to rethink their static and rigid business intelligence and analytic software architectures in order to continue working at the speed of business. [1] It's clear that time has become the new gold standard – you can't produce more of it; you can only increase the speed at which things happen.

Motivation

The world is becoming an increasingly digital space. We share almost all of our lives online. This is transforming our understanding of the world and our place in it. Big data analytics helps organizations bind their data and use it to identify new opportunities. That, in turn, leads to smarter business moves, more efficient operations, higher profits and happier customers. So, it is invaluable for business to understand the processing and learn and study the various programming paradigms and develop more. It could provide a window into the lives of customers that we have never previously imagined. In his report Big Data in Big Companies, IIA Director of Research Tom Davenport interviewed more than 50 businesses to understand how they used big data. [8] He found they got value in the following ways:

Cost reduction- Big data technologies such as Hadoop and cloud-based analytics bring significant cost advantages when it comes to storing large amounts of data – plus they can identify more efficient ways of doing business.

Faster, better decision making- With the speed of Hadoop and in-memory analytics, combined with the ability to analyse new sources of data, businesses are able to analyse information immediately – and make decisions based on what they've learned.

New products and services- With the ability to measure customer needs and satisfaction through analytics comes the power to give customers what they want. Davenport points out that with big data analytics, more companies are creating new products to meet customers' needs. [8] Therefore, no way a business can ignore analysing their Big Data. For analysing the data we must know how to process it and how to store it at a very large level. Hence programming paradigms are necessary to be studied upon in order to progress in the field.

Related work

Google had a deep understanding of unstructured data and became one of the earliest companies who started working on big data. So, in order to handle big data, in Sixth Symposium on Operating System Design and Implementation 2004[OSDI'04], Google published a paper on MapReduce [12], a programming model and also introduced an associated implementation for processing and generating large data sets. Google had already been using MapReduce for their in house implementation to utilize commodity hardware with multiple machines to process big data. This quite worked for them but was not completely satisfactory. After that, Apache Software Foundation released the project Hadoop, an Open-Source Implementation of MapReduce with other required tools for processing data on multiple commodity hardware computing cluster like Hadoop Distributed File System [HDFS] [13]. It is a distributed file system which is necessary to execute MapReduce on a cluster.

Although Hadoop is the most popular implementation of MapReduce out there right now for batch data processing but there are also other options for Big Data processing. If your data does not quite fit in Batch Model of MapReduce then there is Graph Model for processing data and if you have an endless stream of data and your requirement doesn't stop there too, then you may find Streaming Model for data processing useful.

Batch, Streaming and Graph are the most common data models for processing and analysing data but we can't really fit real life data into these simple data models. So generally we try to break real life data or reduce it directly into these data models or some mixed form of these data models. We have a lots of tools to process different types and variations of data because each type of data can't be analysed and processed through the same model. Let's talk about some of the most common tools.

For Batch Model processing we have Hadoop [16], HPCC [17], DISCO [18], and Dryad [26] (Abandoned). For Graph model we have Kafka [21] and Giraph. [24] For streaming model we have Storm [20] and Samza. [22] Samza can be used for Async computation messaging in streaming model and can also push message in queue and hence, works out of the box with streaming processing infrastructure. [23]

DataFlow [25], introduced by Google researchers is a data processing model that can handle both data models Batch and Streaming. All sort of mixed data models that can be broken into Batch and Streaming can be handled by DataFlow elegantly.

Flink [19] [38] can handle batch and streaming model, in fact, you can run your DataFlow processing over Flink too.

As we said, since all data processing problems are different, people developed different solutions for problems like Microsoft developed Dryad and Apache Tez [27], for DAG (Directed Acyclic Graph) of Jobs because Hadoop was not suitable for repetitive Jobs, so for that, people developed Tez on top of Hadoop. Other than that we see all kinds of Distributed File Systems, extension and tools for processing data on the top of the standard implementation of basic data model. For example Mahoot [28] for Machine learning, Cassandra [29], Pig [30], Spark [31], Hive [32], HBase [33], Chukwa [34], Ceph [35], Kylin [36], Drill [37], MongoDB[39], CouchDB[40], Infinispan [41],Riak [42] and DynamoDB[43]. DynamoDB and Riak are built upon Amazon's Dynamo Storage System [44] (NoSQL Distributed storage system with key-value pair and DHT (distributed hash table) support). Some of these are Open Source and some of these tools were developed in house by companies for their processing needs and then released in Open Source like Kylin, first developed by eBay and CouchDB, developed by IBM etc.

Problems

Problem with MapReduce is that if we take a look at hardware side of distributed computation then limited bandwidth and congestion at the master node in MapReduce is an issue [45] because when our multiple helper nodes send data back to master node after calculation and if you are running it on a massive scale then your network hardware link between your master node and helper node can be a bottleneck.

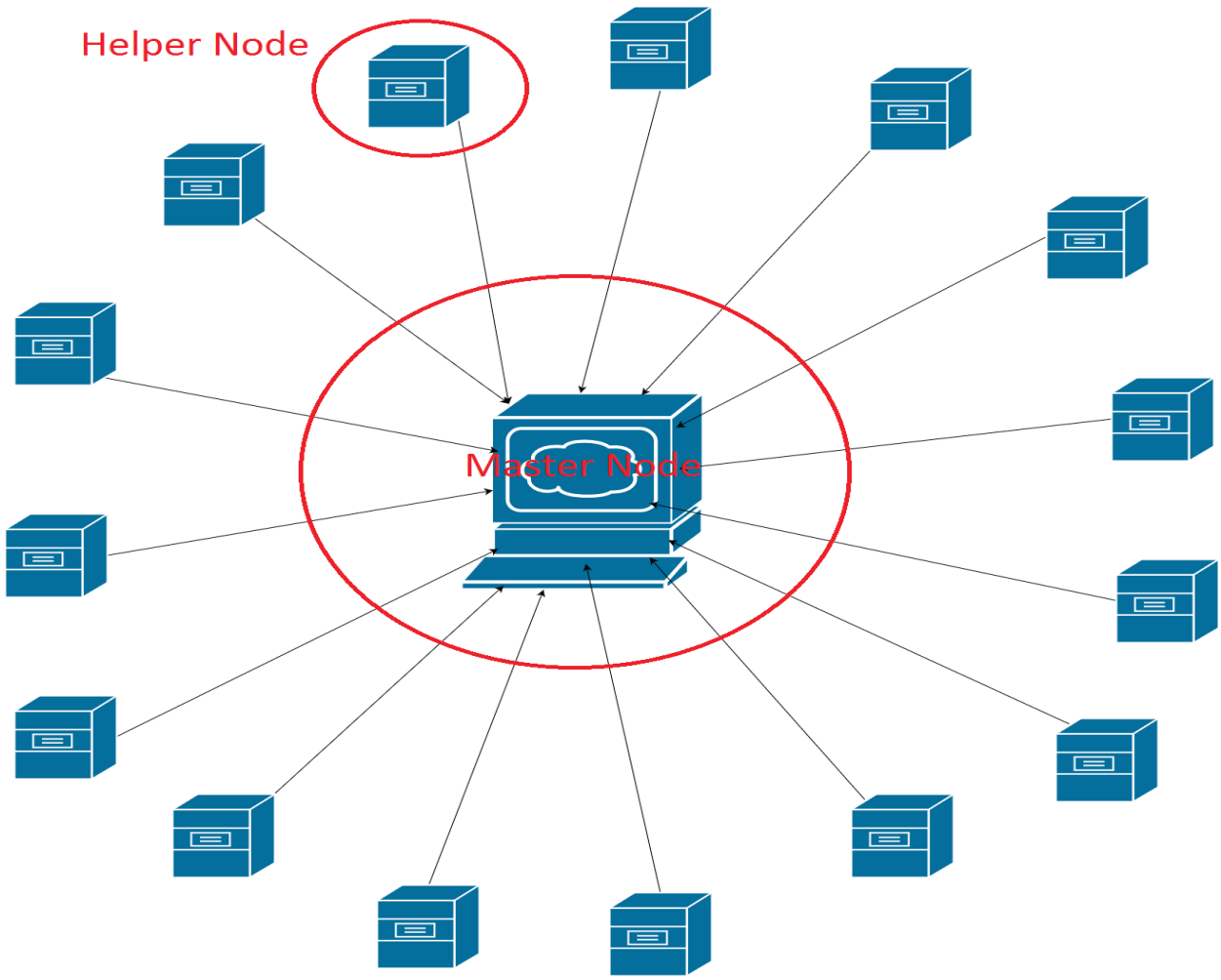

Fig 1: Bandwidth bottleneck in MapReduce while computing on large scale.

Some suggested approaches to solve to problem

Hardware bottleneck can be solved using special networking hardware with wider bandwidth support on the master node but that can cost us much more money than commodity hardware. So to resolve bandwidth bottleneck we can introduce special helper nodes which aggregates data from other normal helper nodes and then submit that data to master node. But another drawback with this is that we are introducing another point of failure in the system. So to prevent failure of special helper nodes we should make these helper nodes fault tolerant since if a special helper node fails we lose work of all the helper nodes working under that special helper node. So with the help of special helper node we removed that networking bottleneck that occurred at master node but we have to trade-off this with another point of failure.

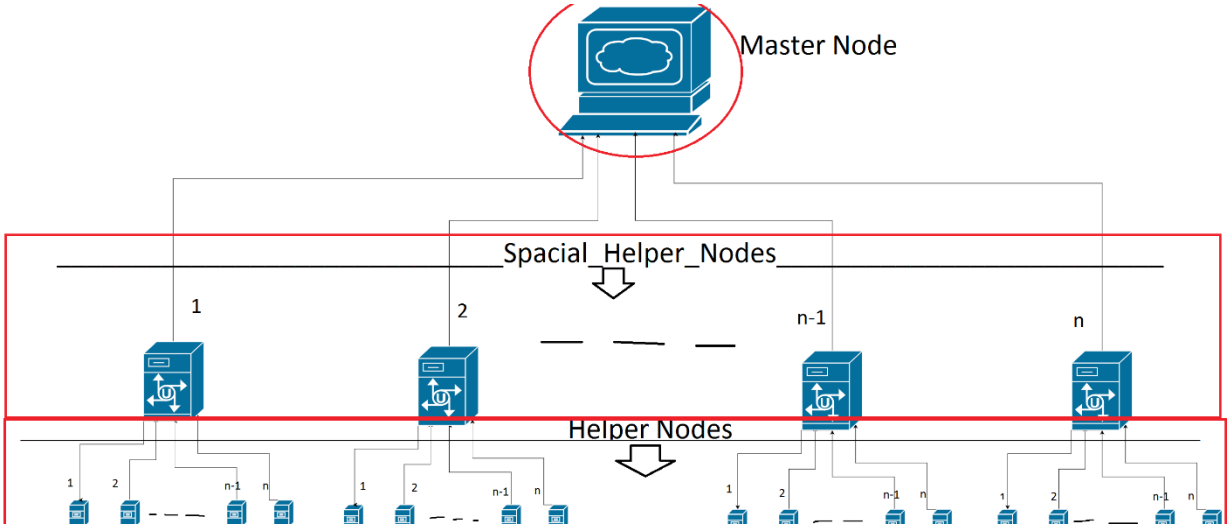

Fig 2: Proposed solution with introduction of special helper nodes.

Conclusion

We have seen that not every real life data can be modelled into predefined data models because each type of data is special in its own way. Hence, we can't use the same tools for processing in order to achieve our desired goals. This has lead us in developing more and more standard and complex data models that can be used to express and analyse the kind of real life data that we come across today. But still today we don't have enough tools and models to analyse every possible real life data. Another factor is time taken for processing data especially in critical systems. For example, in a big data analysis system developed to predict stock market behaviour based on previous data, if we fail to deliver processed results in time then those results are useless and have no real value and serve no purpose. That's where time constraints comes into play but right now we don't have strict deadline based systems for analysis of big data that guarantees us to deliver results under certain time constraint but that's fine until we have time critical mission derived by result of that data. To ensure that our time critical mission is not affected by time taken by analysis of data we must develop Hard Deadline Data processing systems.

References

1. http://www.odbms.org/blog/2013/12/challenges-and-opportunities-for-big-data-interview-with-mike-hoskins/
2. http://pubs.sciepub.com/jcsa/3/6/13/
3. Felten, E. 2010. "Needle in a Haystack Problems",https://freedom-to-tinker.com/blog/felten/needle-haystackproblems/
4. http://craig-henderson.blogspot.in/2009/11/dewitt-and-stonebrakers-mapreduce-major.html
5. http://www.contegix.com/big-data-comes-with-big-problems/
6. https://www.researchgate.net/publication/236263585_Big_data_processing_Big_challenges
7. https://en.wikipedia.org/wiki/MapReduce#Restricted_programming_framework
8. http://www.sas.com/en_us/insights/analytics/big-data-analytics.html
9. https://www.youtube.com/watch?v=TzxmjbL-i4Y
10. https://www.youtube.com/watch?v=mhe5kX10CR4a
11. http://www.drdobbs.com/database/dataflow-programming-handling-huge-data/231400148
12. https://en.wikipedia.org/wiki/MapReduce http://static.googleusercontent.com/media/research.google.com/en//archive/mapreduce-osdi04.pdf
13. https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html

14. Hadoop vs Spark vs Storm https://www.dezyre.com/article/spark-vs-hadoop-vs-storm/145
15. Graph Data Model http://www.graphanalysis.org/SC12/02_Feo.pdf
16. https://en.wikipedia.org/wiki/Apache_Hadoop
17. https://en.wikipedia.org/wiki/HPCC
18. http://discoproject.org/
19. https://flink.apache.org/
20. http://storm.apache.org/
21. http://kafka.apache.org/
22. http://samza.apache.org/ [used for asynchronous computation messaging]
23. http://samza.apache.org/learn/documentation/latest/introduction/background.html
24. http://giraph.apache.org/
25. http://www.vldb.org/pvldb/vol8/p1792-Akidau.pdf
26. http://research.microsoft.com/en-us/projects/dryad/
27. https://tez.apache.org/
28. http://mahout.apache.org/
29. http://cassandra.apache.org/
30. https://pig.apache.org/
31. http://spark.apache.org/
32. http://hive.apache.org/
33. http://hbase.apache.org/
34. http://chukwa.apache.org/
35. http://ceph.com/
36. http://kylin.apache.org/
37. https://drill.apache.org/
38. [talk on Data Flow in @scale conference] https://www.youtube.com/watch?v=3UfZN59Nsk8
39. https://en.wikipedia.org/wiki/MongoDB (MongoDB)
40. https://en.wikipedia.org/wiki/CouchDB (IBM developed in house then Open Sourced.)
41. https://en.wikipedia.org/wiki/Infinispan (RedHat)
42. https://en.wikipedia.org/wiki/Riak (Basho)
43. https://en.wikipedia.org/wiki/Amazon_DynamoDB (Amazon)
44. https://en.wikipedia.org/wiki/Dynamo_(storage_system) (NoSQL Distributed storage system with key-value pair and DHT (distributed hash table) support), DynamoDB and Riak are built on this.
45. https://www.youtube.com/watch?v=n0PL74zwaoM

---

* Surajnath Sidh {surajnath.sidh@st.niituniversity.in / www.github.com/electron0zero}
# Saumya Gupta {saumya.gupta@st.niituniversity.in }