**http://www.sqlservercentral.com/articles/Big+Data/95647/**
Printed 2013/10/03 10:07AM

# Big Data for SQL folks: The Technologies (Part I)

**By Frank A. Banin, 2012/12/11**

## Introduction

Big Data has not been a source of much discussion in this forum even though Steve Jones has started the conversation a couple of times**.** It doesn't matter what you're currently doing, whether you're an Analyst, BI Professional, DBA or Developer, if you intend to stay in the data business then stay tuned because big data may cross your path very soon. This write-up presents a simplistic but comprehensive technical overview of the major Big Data technologies available today.

As trite as it may sound, I cannot ignore the fact that Big Data has been seriously hyped over a period of time now, but thankfully, there seems to be a general consensus on what it constitutes. With that said, let me mention that you will still find some experts with extreme opinions or vendors still hyping various aspects of Big Data to suit their needs.

# The Big Data Phenomenon

Big Data is now accepted to be large and complex data difficult to store, process, analyze and visualize using the traditional application architecture (conventional relational databases, desktop analytical and BI tools). In this context it is also safe to say that, what may be big data for a small to midsize company, may not be Big Data for a larger company. One often present component of Big Data is unstructured data which comes in the form of web-pages, web-logs, e-mails, memos, reviews, user groups, chats, sensor-data, image-files, audio video-files, and marketing material, news etc.

The definition of Big Data started out referring mostly to big data sizes, this was in an era when pundits would use various byte prefixes in the amount of data being collected and projections for the future. Also very common during this period were citations of extreme data collection and supercomputing efforts by mainframe and grid-based computer network infrastructure which are not reflective of the Big Data problems being addressed today.

With no consensus on the size (volume) or thresholds of whats constitute "Big", the definition of Big Data shifted to what is referred to as the three Vs, namely; Volume, Velocity and Variety of data, a definition which tried to incorporate the rate at which data is being accumulated and the variety of the data being collected, in addition to the initial size factor. A fourth V, Veracity, was also introduced in some circles. The problem with this definition was that it was still vague in many ways because, without thresholds and precise constituents, anybody could define his or her own V.

All the while, there were those who laid more emphasis on the value that can be derived from this data, since they believed the concept of data explosion was not really new. This group was also quick to attribute many data analysis related successes to Big Data, some of which were not on Big Data per se.

These two trends continued for a while often with some common grounds, but depending on the forum,

emphasis could be placed on any of the points outlined below, in no particular order:

· Size of Data

· Unstructured and semi-structured data

· The idea of gaining competitive edge in business through the use of available data.

· Nosql (or Not Only SQL)

· Advanced analytics (data mining and predictive analytics, Machine learning adaptive algorithms.)

· PhD's shacked-up in some corner offices applying complex algorithms to data not comprehensible by the average mind.

· New RDBM Appliances consisting of special and tailored hardware and software.

· Cloud computing

· Data Science

· Role and relevance of statistics in data analysis today.

Out of this a pattern finally started emerging. What is the point in collecting all this unstructured data if you don't intend to derive any value from it? Also, storing and analyzing the data and types requires some new hardware and software with the right storage and processing power for any type of analytical requirements thrown at it. All these factors undoubtedly require new software applications and various levels of new and improved skillsets.

The new consensus later embraced were the ideas that;

1. Whilst Big Data analytics used to be for mostly establishment name brands (Amazon, twitter, Google, Yahoo, facebook, Bing etc) and conglomerates with cash to burn, today Big Data technology can be mainstream and available to enterprises of all sizes.
2. Hardware, especially for storage is generally commodity hardware, i.e. computer hardware that is affordable and easy to obtain.
3. Big Data systems should fit into existing database-and-BI infrastructure and end-user applications.

# Technological Challenges

From the definitions and ideas above, two major points stick out in terms of technological challenges;

- Storing and processing large structured and unstructured data.
- Deriving value from the data through analysis.

Even though storage is becoming cheaper, storing Big Data could amount to managing multiple (tens to hundreds) servers. Secondly, deriving value from this huge amount of data (in some cases near real-time) also requires a lot of processing and analytical power. Finally, managing and presenting the back-end processing for analysis and integration into any new technologies and systems requires many presentation layers and protocol.

# Big Data Engines

From these challenges we can break down the major features of Big Data technology into two solutions:

1. A new engine with the storage and processing power to handle this new data and any kind of analytics thrown at it. The engine should be able to scale to storing petabytes of data and running large-scale parallel computations.
2. Management tools, APIs, connectors and front-end analytical tools that make data from the engine accessible for analysis and integration with existing systems.

For the rest of this discussion we will look at the major technological solutions that address these challenges. In this article we'll look at non-relational technologies that employ a distributed systems approach in the engine design. In the following articles we will look at relational technologies that employ Massive Parallel Processing in the engine design and lastly we'll look at the emerging Hybrid approach.

## The Hadoop Engine

Almost synonymous to Big Data nowadays is the word Hadoop. It's almost impossible to illustrate Big Data technology without mentioning Hadoop, Google, and a handful of other big names.

There are many non-relational Big Data solutions out there, but Hadoop is the most popular**.** The Apache Hadoop software library is an open-source framework that has found success in various market segments. Almost every major Big Data player has a project directly or indirectly designed around a Hadoop.

So what Big Data technology do both Hadoop and Google have in common?

1. They both implement special versions of the *Distributed File System* - a protocol used for storage and replication of data. Google has it's Google distributed file system (GFDS) and Apache has Hadoop distributed file system (HDFS).
2. They both also implement a special application that allows parallel processing using *MapReduce.*

The main selling points on the Hadoop engine are;

· HDFS (Hadoop Distributed File System) enables applications to scale to petabytes of data employing commodity hardware.

· Hadoop's MapReduce API enables work parallelization.

· Reliability enabled by fault-tolerant design. Hadoop engine's ability to replicate by MapReduce execution means it is able to detect task failure one node on the distributed system and restart programs on other healthy nodes.

### The Other Stuff

Note that the technologies we've discussed are mostly backend technologies that make storing data and running large-scale parallel computations possible - these technologies are not front end analytical platforms. All other implementations, like databases that utilize the engine to manage data (structured and unstructured) and any other front end analytical and monitoring tool forms part of what I call the "Other Stuff".

For instance, Google manages their structured data (over sixty of their products) through an in-house database-like application called *Bigtable*. Bigtable is a distributed storage system designed in GDFS (Google Distributed File System) that can be used with MapReduce for running large-scale parallel computations. Google claims Bigtable is not a database, it just resembles one and shares many implementation strategies with databases.

But the point is, Bigtable for instance modifies Google's Big Data equation as below.

Google Big Data Solution = Google Engine + *Bigtable* + $Stuff_2$ +……+ $Stuff_n$

So for these non-relational distributed Big Data technologies, even though their engine is necessary for big data solutions, equally important are the *other stuff* as we can see from this illustration.

## The Hangover

The major constituents of the "Other stuff" part of the hadoop Big Data Technology is what leads to a phenomenon some experts have termed the *Hadoop Hangover*.

The term stems from fact that many early adopters who opted for the Hadoop solution did not know they were getting only the base engine. The problem is, depending on the Big Data problem you are trying to address, the front-end part of your implementation could extremely significant.

## The Farm

The major hurdles discussed in the previous two sections led to a race to come up with solutions that will address most of the challenges in adopting the Hadoop solution. In the Apache Hadoop world this spun a lot of other projects. You've probably heard of projects, most of which sound like something from a farm (in keeping with the name of a toy elephant after which Hadoop was named).

***Hive:*** *A data warehouse infrastructure built on top of Hadoop . This interface allow users to issue queries in HiveQL, a SQL-like query language.*

***Pig:*** *An interface that allows developers to write data analysis programs in a high-level language, called Pig Latin, custom made for exploring very large datasets.*

***Beeswax:****The Beeswax application enables you to perform queries on Apache Hive.*

***HiveQL:*** *A SQL-like language for Hadoop which converts scripts (existing or new) into MapReduce jobs.*

***Hbase:*** *provides a non-relational database atop. One application that is particularly well suited for HBase is BLOB stores, which require large databases with rapid retrieval. BLOBS -- binary large objects -- are typically images, audio clips or other multimedia objects, and storing BLOBs in a database enables a variety of innovative applications.*

Applying some of these technologies to my equation, a typical Hadoop solution could be as below.

Hadoop Big Data Solution = Hadoop Engine + *Hbase* + *Hive* +……+ $Stuff_n$

## SQL on Hadoop Fever

As we discussed earlier the hadoop engine runs MapReduce jobs. The problem is, the average data professional does not know this application technique, but is very conversant with the good old SQL language. So just as the Hive and HiveQL are data warehouse-like infrastructure with SQL-like query language that converts scripts into MapReduce job, there is also a race amongst some of the major players in the industry with each trying to come up with similar infrastructure that provides faster and more extensive SQL querying capabilities atop of Hadoop. Besides the numerous start-ups, some of the major ones include; Impala from Cloudera, Stinger from Hortonworks, Apache Drill from MapR, Pivotal HD from EMC, Big SQL from IBM, SQL-H from Teradata. Some projects that started as separate efforts turned into marriages.

# Nosql Databases

These new breed of non-relational databases, like Hbase in the hadoop world, designed primarily with unstructured data in mind are the ones generally referred to as Nosql Databases. These databases are non-relational, distributed, open-source and horizontally scalable.

Even though many are being designed on top of Hadoop by various companies, many more are non-hadoop and are built on other open-source platforms or on proprietory engines. For instance Amazon has its own called Dynamo and Facebook calls its Cassandra.

However the general accepted characteristics of Nosql Databases are that: they are schema-free, they have easy replication support, they have simple API and have the ability to store huge amount of data. They are primarily BASE, not ACID like their relational counterparts.

They are also classified based on different storage and querying paradigms. A summary of some of the major breakdown and few database examples are outlined below.

- Wide Column Store/Column Families
  - Hbase (hadoop)
  - Cassandra (facebook)
- Document Store
  - MongoDB
  - CouchDB
- Key Value / Tuple Store
  - DynamoDB
  - LevelDB

It was some in this world that will initially argue that SQL and relational database were obsolete. But if you've been to any of their presentations they are quick to admit the existing hurdles**.** For instance, unlike RDBMS with a standardized query language, they all run proprietary query languages, some with relatively steep learning curves.

# NEXT

In this session we looked the evolution of Big Data as a phenomenon. We also looked at the non-relational Big Data technology world and some Big Data terminologies. That characteristics outlined for the Technology are general and that underlying and details implementation may vary. They scale well and have the ability to manage huge amounts of data.

In the next installment we will look at Relational Big Data technologies how they employ MPP engines and appliances to address Big Data challenges. We will also look at the hybrid approach, where adaptors are allowing the two sets of technologies to converge, using SQL Server as an example.

---