Big data analytics poses new challenges to data quality and storage

Recent developments of information technology have enabled more information than ever to be collected by businesses. This abundance of data has provided businesses with tremendous opportunities to operate more efficiently and profitably through big data analytics. A report by McKinsey has projected that big data analytics can help U.S. Healthcare sector save over 300 billion dollars every year while increase operating margin in the retail sector by more than 60% (Manyika et al., 2011).

However, despite those potential benefits, firms are still facing challenges in converting big data into realizable results. According to a research by KPMG, around 85% of the CFOs and CIOs still have trouble conducting analysis (Capital, 2014). The report primarily attributed such failure to the fact that only 39% firms have trained their analysts with big data technologies (Capital, 2014).

In the world of big data analytics firms are required to construct data infrastructures that are able to sustain the volume, velocity, and variety of big data. In order for a database to satisfy those requirements, it not only needs to be able to partitioned efficiently to enable more storage capacity, but also a schema that is able to process those data with high speed and efficiency. Even though the traditional, relational approach to data management was effective in recording operational transactions, many schematic restrictions of the traditional database, such as a complex query system and partition intolerance have prevented it from becoming a efficient analytics schema for big data.

In this paper, we are going to explore technologies of big data analytics and offering managerial insights for firms regarding how to successfully incorporate analytics into their existing database infrastructures. We are going to accomplish this by first comparing and contrasting features offered by big data technologies compare with the traditional databases. Then we are going to introduce several different types of "analytical databases". Finally, we are going to offer some managerial insights for the adaptation of big data technologies.

ACID principles ensure maxium transactional data quality.

Before introducing traditional relational database, we believe it is necessary to describe the task those databases are designed for and the goal they are attemptign to achieve. As explained in the introduction, the traditional database is designed to measure each trasaction accruactly, and this acuracy is measured by the "ACID (Atominization, Consistency, Isolation, Durability) principle" (Haerder & Reuter, 1983).

ACID, first proposed by Theo Haerder and Andrea Reuter in 1983, describes a principle that has been accepted as golden standard for traditional relational database. Databases that upholds ACID are optimized for the operational setting where all data input and output must be accruate. All ACID features have been displayed in the table below. However, ACID principles are not optimized for tasks related to big data anlytics. For instance, because big data comes from a wide variety of sources that include video, audio, and texts of all formats, maintaining consistency of the database would be really difficult as all files of different types need to be converted to the

same type before entered into the database. Therefore, many present big data tehcnologies have forgone at least some elements of the ACID principle. Details of the ACID principle is presented in Table 1.

Table 1. ACID Principle Overview

Atomization	Atomization ensures that all of the elements stored in a traditional database should be at the lowest level of granularity. Atomization guarantees that all aspects of a database transaction must succeed in order for the change to the database to be completed. Otherwise, nothing regarding such transaction will be completed.
Consistency	Database consistency means that all entry into a database must be valid entry as defined by the database that includes the type requirement, space requirement, and other specific requirements for a specific column. It is to ensure that all elements extracted from the same role of a database behave predictably.
Isolation	Isolation ensures that all update and read request within a database are separate from each other. Isolation principle is created to avoid update and acquisition conflicts. Implementation wise, isolation principle only allows one user can access a specific data at any instances of time.
Durability	Durability principle ensures that after each transaction are complete; the change will persist forever in the database, surviving system failure and other extraneous circumstances. It is created to ward databases against crashes or other unforeseen disasters.

*All sources of the table can be contributed to Haerder, T., & Reuter, A. (1983). Principles of transaction-oriented database recovery. *ACM Computing Surveys (CSUR), 15*(4), 287-317.

Traditional database falls short in big data analytic tasks

This section is going to explore how traditional database structure satisfies each components of the ACID principle. While accomplishing this, we are also going to point out weaknesses of several traditional database components.

The relational database model is constructed based on relationships between entities that is relevant to the corporation. Each entity, along with its attributes, are stored in a separate table while their key values connect each tables in the form of foreign keys. Before entering into the system, each entity tables must be normalized to eliminate potential redundancy and inconsistent dependency (Microsoft, 2013).

Relational databases are queried with the SQL system. Due to the strict structure of the relational database model, users can conduct complex queries without much customization. Compare with other method of information retrieval, SQL provides a much more flexible and simple way to retrieve data for operational purposes (Marchus, 2013).

One of the biggest advantages of the relational schema is its simplicity. Data are stored in tables that consist of rows and columns, where each row represents an instance of the entity and each

column represents an attribute of the entity. A lock-in system is in place to prevent two operations from accessing and manipulating the database at the same time, therefore, the database is always in compliance with all four of the ACID principles (Oracle).

A weakness of the relational schema is the data upgrade lock-in system (Marchus, 2013). The existence of this lock-in system has prevented efficient processing of data as one transaction has to be completed before another one can be initiated. In big data analytics, data are usually stored multiple time across server to prevent data lost, greatly elongate the time systems needed to keep all copies of the same file up to date. Achievement of this kind of "strong consistency" almost inevitably means sacrifices in terms of processing speed, an attribute that is extremely important for big data analytics.

Even though SQL system is a convenient way to access data, it is relatively computational intensive. For analytics operations to occur, tables must be joined by their key values, which are usually very computationally intensive. In many complex SQL cases, all tables involved might need to be scanned completely (Marchus, 2013).

Another fundamental weakness of the relational database is that its structured data model prevents more complex data structured from being modeled. Unstructured data sources such as video, audio, and online images simply cannot be stored in a column-and-row database. Even if they were able to be stored in relational database, it would be really difficult to come up with SQL queries to extract information due to the complex, logical nature of the SQL system.

CAP poses a trilemma for big data analytics

One of the most pressing challenges for database designed for big data storage and analytics is to scale across multiple servers while preserving transactional accuracy provided by the traditional database. This dilemma was most accurately described by the CAP principle, which stands for Consistency, Availability, and Partition Tolerance. It was first proposed by Eric Brewer in 2000 describing three desirable web-service guarantees (Brewer, 2000).

Even though all three principles are important, Eric Brewer pointed out that those three guarantees couldn't exist simultaneously within a web service, or database system. For traditional database, partition tolerance was viewed as the least important feature as not enough data are collected for partition tolerance to be a pressing need.

In context of big data analytics, partition tolerance is considered the most essential part of a database. Without the ability to store the ocean of incoming data in the first place, analytics simply cannot take place inside a database.

As a result, currently big data technologies provide trade-offs between consistency and availability according to features it wants to provide and its target customers. For example, banks would prefer consistency to availability where the consumer goods sector would prefer availability to consistency. Big data technologies have also found ways to relax the CAP principles to make the existence of all three features possible. The relaxed CAP principle are

usually described as BASE, standing for Basically Available, Soft state, and Eventual Consistency. More detail of CAP and BASE principles are illustrated in the Table 2.

Table 2. CAP and BASE Principle Overview

CAP	Description	BASE	Description
Consistency	Database consistency means that all replica of a same dataset will be of the same content in the database. Relational databases are always consistent because only one dataset was written, read and modified other ways. To reach scalability, many big data databases are replicated with the same dataset represented in multiple times inside of the database (Marchus, 2013). Many NoSQL databases still enforces strong consistency, meaning that all changes on the dataset must be synced across all instances before the next operations may occur.	Eventual Consistency	In an eventually consistent database, he dataset could be accessed even though it is not consistent across all instances while hoping that the dataset will be eventually consistent in the future. Eventual consistency has been argued to greatly increase the scalability and processing speed of databases (Marchus, 2013).
Availability	Availability describes the accessibility of the data in all time spaces. Traditional databases at most times are available when processing small amount of data. However, as the size of the database increase and the frequency of operation request on a single data point increase, traditional database will become increasingly unavailable. For NoSQL databases that utilize eventual consistency, the availability is much higher than those who uses strong consistency due to relaxation of the data-locking mechanism (Marchus, 2013).	Basically Available	Even though a basically available database might not offer most-updated content of the query user have requested, it will always return a response regarding the state of the query. This response might be a “failure” notice or previous image of the data requested (Roe, 2013).
Partition Tolerance	Partition Tolerance describes the ability for the database to be divided up and distributed among multiple servers. Traditional	Soft State	When a database has a soft state, it means that it does not necessary store all of its data on the hard drive. Instead, most of

	databases are mostly partition intolerant due to strictly enforced schemas. Almost all big data databases are partition tolerance due to the need of scaling out. Partition Tolerance is considered the most important aspect of big data technologies (Marchus, 2013).		the data will be stored in-memory to increase the scalability of the dataset without purchasing newer servers (Chiappa, 2014).

Big data technologies take different approaches to optimzie for big data anlytics

Seeing the weaknesses of the relational schema in conducting big data analytics, many different types of databases were created in order to fill the void of big data analytics. In this section, we are going to introduce three big data systems.

The first system is the NoSQL systems, an abbreviation for "Not Only SQL". NoSQL databases sacrifice some attributes of ACID in exchange for scalability and processing speed big data analytics requires(Perret, 2014).

The second solution is the Hadoop, an Apache open-source ecosystem that solve the scalability challenges through a file distribution system (HDFS) and a parallel processing paradigm (MapReduce or YARN). Those two systems are further complemented by sub-projects such as an SQL query system (Hive), a NoSQL system (Hbase), and an in-memory storage system (Spark).

The third system is called "NewSQL." Created to challenge NoSQL databases, the goal of the NewSQL databases are to maintain the consistency and availability of the traditional database system while tackling the scalability challenge posed by big data.

NoSQL databases utilizes alternative schemas optimized for big data analytics.

NoSQL databases has recognized that even though transactional consistency is attractive, scalability and processing speed is more important for big data analytics. Therefore, in most NoSQL databases data are stored in a relatively less-structured format and the query engines are always optimized for big data analytics. Overall, NoSQL technologies are divided into key-value storage databases, key-document storage databases, column storage databases, and graphic storage databases. We are going to offer a general overview of each type of NoSQL technologies and analyze their strength and weaknesses. The similarity and difference between No-SQL systems are also illustrated in Table 3.

Key-value databases is similar to relational database but processes much more efficient

In key-value databases, each row is paired with a key that identifies the row. There are no relationships between row and rows and attributes of each row will be different from another. As described by the category leader Voldemort, "[key value storage database] is basically just a big, distributed, persistent, fault-tolerant hash table (Voldemort)." Due to this non-relational structure,

key-value storage databases can be easily distributed across different servers to achieve scalability. As compromise to scalability, only simple query operations can be conducted on key-value databases such as set, get, and delete (Marchus, 2013). One of the major drawbacks for this kind of database is that it is not able to conduct filter queries, a function that could be useful for many data analytics functions.

In order to improve the query system of key-value storage, other companies have introduced a variation of the key-value database called key-data structure database. Instead of only storing the key value of a row, values are identified as data types in each row so that data type specific atomic operations, such as appending a string, can be performed (Redis, 2015). The category leader of this type of database is Redis.

Key-storage databases are probably the most straightforward NoSQL databases. It provides a simple solution of scalability by completely forgoing ACID principle and relational schema. However, despite improvements by the key-data structure databases, the query functions of this type of database are extremely limited.

Key-value storages are optimized for structured data with fast processing speed

In general, key-value storages are designed to support business activities that require high read and write speed while requiring a relatively simple data model. More specifically, key-value storages are best fit for storing and analyzing log data, sensor data, and advertisement analysis. In this section, we are going to present a use case of key-value storage by Riot Gaming, maker of the popular video game *League of Legends* to conduct real-time.

Because *League of Legends* is one of the most popular online games, it has to collect data of over five million concurrent games and update it to its database many information statistic such as kill, death, and assists(Eardley, 2013). The legacy systems of the game were built upon a relational database stored in a single server. There are many major issues of the legacy system. First of all, it has a single point of failure, which will terminate the gaming service if it fails. Secondly, the database has low concurrency while providing transactional consistency the game does not really need(Eardley, 2013).

In order to solve this issue, Riot Gaming deployed Riak, one of the leading key-value storage databases, to replace the legacy system(Eardley, 2013). The new key-value database enabled horizontal integration of multiple servers and enlarged the cache capability of the database. The new database is also able to support more clients from different region due to its flexibility. Overall, key-value databases enabled Riot to more rapidly read and write in its databases and perform real-time compilation of simple statistics. This use cases has exemplified key-value databases' processing efficiency.

Bigtable Storage enables extreme scalability of high-volume data storage

The concept of "Bigtable" storage was first came up by Google in 2006 as a way of storing structured data using a distributed system in order to scale better. According to the original BigTable Document,

> "A Bigtable is a sparse, distributed, persistent multidimensional sorted map. The map is indexed by a row key, column key, and a timestamp; each value in the map is an uninterpreted array of bytes (Chang et al., 2008)."

The features of Bigtable are extremely similar to that of key-value databases. Compare with key-value, Bigtable databases provides less availabilities due to its column storage schema while offering better ability to scale out as a tradeoff. The category leader for BigTable database is Apache Cassandra, implemented by the firm DataStax.

Big table storages are generally used during the operating stage of big data, providing massive data scale-out capabilities. More specifically, Bigtable storages can apply to business functions such as personalized recommendation, fraud detection, and product catalog and playlist storage(Datastax, 2015). In this section, we are going to present Bigtable's use case at Netflix to further illustrate its features.

Netflix is the leading online video streaming provider. One of the key business functions of Netflix is to collect viewing data about its subscribers and offer personalized recommendation to them (Datastax, 2014b). As of now, millions of subscribers' watches over two billion hours of contents on Netflix everyday, which over capacitated the traffic processing capability of its traditional oracle database(Netflix, 2015). In addition to processing limit, the legacy database had a single point of failure and provides poor streaming quality for its customers, both features that will drive massive customer loss if not improved (Datastax, 2014b).

Netflix utilizes Cassandra to replace its traditional Oracle Databases after the expansion of its multi-device platform in 2010. The bigtable storage reduced the failure rate of Netflix's database by providing a distributive system and flexibility of schema changes. Currently, Netflix stores about 95% of the non-programing data in Cassandra, ranging from customer account information, viewing logs, to movie ratings (Datastax, 2014b). This example has provided illustration to bigtable databases' scalability.

Key-document databases provide simplified storage schema for storage of unstructured data

Key-document database is one of the most popular NoSQL databases on the market. Lead by MongoDB and CouchDB, Key-document databases store data inside a structured HTML or JSON file, replacing the traditional column and row tables. Instead of indexing each of the values, key-document databases only index the documents. Inside those JSON or HTML files, all data were stored utilizing a multi-layer dictionary system and can be embedded recursively inside one-another (Marchus, 2013).

In order to enable convenience retrieval of data from those documents, different companies uses slightly different indexing methods to support their own query. MongoDB, for example, utilizes B-tree data structure and generate their indexes at collection level. Indexing MongoDB supports

include but not limited to Default id, User-defined single field id, multi-key ID, and compound ID (MongoDB).

In an application perspective, Key-Document databases' advantage is its ability to store data from all sources and of high variety. More specifically, document-storage databases excel at supporting data integration, service personalization, and collecting data from multiple sensors from Internet of Things devices. In this section we are going to use case study from MongoDB to illustrate how this advantage is applied for MetLife Wall, an integrated customer service platform.

Metlife's data challenge originated from its legacy data instead of its legacy databases. First of all, due to regulatory changes throughout history, many of MetLife's data are missing fields in terms of policy records, which will provide a huge resource drain to its relational database. Secondly, in order to provide a unified view of a customer's profile, unstructured data such as death certificate or health records need to be stored in the integrated system, a task that is impossible for relational databases. Finally, in order to construct an integrated system, Metlife need to draw data from a variety of existing database systems with different schema – a task difficult to accomplish with relational database(Henschen, 2013).

Eventually, MetLife has chosen constructed its integrated system using a document-storage database. The MetLife Wall integrate all information of MetLife's 100 million+ customers from a variety of sources to provide a 360 degree, one-stop view of the customer's need and wants for customer service representatives. It improved information access time, reduced call times and increased expense saving for MetLife. It also enabled MetLife to provide real-time analysis of a customer and provide personalized promotion accordingly(MongoDB, 2015). This example illustrated key-document databases' ability to store unstructured data.

Graph storage provide unique insight through its alternative schema

Graph databases store data fundamentally different from other kinds of database systems. In other database systems, data were either stored in a hierarchical structure (Key-… storage) or in a relational manner. In graph storage, however, there is no intrinsic importance between object relations, and all relationships were arbitrary. The goal of Graph Storage is to connect entity and relationships on the go, instead of putting all data into a pre-defined relational schema. In doing this, Graph Storage companies like Neo4J attempt to help its customers to discover insights that could not have been discovered in other forms of databases.

Graph Storage database thrives in accurately tracking customer behaviors and drawing unique insights that other databases are unable to detect. More concretely, Graph storage is frequently used in tasks such as real-time recommendation, social network identification, and fraud detection. In this section, we are going to present an example of how Walmart utilizes Graph Storage to offer real-time recommendations to its customers.

Graph storages are often implemented in place to offer fast read and write operations on large volume of data. Walmart's business need was to provide its customers with high-quality recommendations on Walmart.com. However, being a firm that has to deal with almost 250

million customers across 27 countries, it is essential for Walmart.com to acquire a customer's past purchases and conduct analysis quickly to perform real-time recommendation. It has implemented Neo4j, a leading Graph storage database in 2013 to fulfill this need. Compare with other forms of databases, Neo4j conduct those operations significantly faster due to the graph storage schema and can optimize cross-selling major product lines in core markets (neo4j, 2015).

Table 3. No SQL Technologies and Vendor Profiles.

	Key-Value	Big Table	Key-Document	Graph Storage
Advantage	Fast Processing of Relatively Structured Data for operation	Store and processing of massive amount of log data for services like recommender system.	Integrate data sources with high varieties for better customer service experiences	Draw insights from its flexible, unique data schema and more accurately mimic real customer experience
Major Vendors	Basho(Riak)	Datastax(Cassandra)	MongoDB, CouchDB	Neo4J, OrientDB
Most Relevant Industry	Consumer, Gaming, Web Security	Universal	Customer service, Insurance services	Retail, E-Commerce
Consistency vs Availability	Availability	Consistency	Availability	Availability

Hadoop ecosystem offers a mature framework for scalable storage

The Hadoop Ecosystem describes an open-source implementation of the MapReduce Parallel processing paradigm. It has provided corporations with a solution to the big data storage and analytics challenges by proposing another way to store and process big data files coming in from all sources. As mentioned in the introduction, the primary components of Hadoop are the Hadoop File Distribution System and the Map Reduce Parallel Processing Paradigm 2.0, or Yet Another Resource Negotiator (YARN). This subsection is going to dive into Hadoop by first examining HDFS and YARN. Then we are going to grant a brief overview of other components of the ecosystem.

Hadoop Distributive File System (HDFS)

One of the major problems of the traditional database system is that it is not designed to tolerate hardware failures. HDFS is created to solve this problem. HDFS is a file distribution platform where all files in a database were replicated multiple times, partitioned into pieces, and then stored across multiple different nodes called datanodes. These datanodes are in turn managed by a higher-grade server called namenodes, which are responsible for everything related to the namespace of the cluster, including renaming the files and indexing the files (150). When a user

application is trying to access the file through HDFS client, the client first performs a handshake with the relevant datanode to access information about the location of the file, and then access the file block by block from different datanodes according to instructions given by the namenode.

Because files are replicated multiple times in Hadoop before broken down and distributed, it is highly unlikely that all copies of a file fails simultaneously. This also reduces the need to use expensive servers. However, one of the major drawback of HDFS is that it will take a long time for clients to access the file compare with the traditional relational, or most of the NoSQL systems. Therefore, it is always recommended to utilize Hadoop as a warehouse-type analytic database in which scalability is the primary concern instead of processing time.

The MapReduce Parallel Processing Paradigm and YARN

One of the biggest challenges of conducting data analytics in the SQL system is that all processings have to be done locally in one machine. MapReduce was created to satisfy the need of processing queries on multiple servers to achieve the scalability big data needs. It breaks down ordinary data manipulation functions into two components: Mapper() and Reducer(). Each Mappers takes in a key value pair, perform operations on the pair, and then return an intermediate key value pair. Reducers then take in these intermediate key value pairs and combine them into the ultimate result of the query (Apache, 2013).

In MapReduce Framework 1.0, one master node, called JobTracker, is responsible for tracking the status of each node, monitor their tasks, and reassign their tasks if the node were to fail. To respond to JobTracker, in each cluster there is a TaskTracker that is responsible for executing the tasks assigned by the master JobTacker (Apache, 2015). A problem of this architecture is that it runs into resource negotiation problem as the node size increase to thousands or even tens of thousands nodes. At the same time, the system lacks flexibility of accepting other kind of processing frameworks to be implemented over Hadoop. Therefore, recently MapReduce 2.0 was introduced with addition of YARN, another resource negotiator to resolve the scalability issues of the previous MapReduce Model.

In MapReduce 2.0, resource assignment, a job JobTracker was responsible for, is now granted to the new Resource Manager, which is responsible for receiving and running MapReduce and other applications on the clusters. In each node, a NodeManager have replaced TaskTracker and is responsible for allocating resources for each applications and deployment of the application. JobTracker is now mitigated into the MapReduce Application Master, whose sole responsibility is to monitor each of the MapReduce jobs. JobHistory Server, a node that is independent of the architecture, is now responsible for storing information about completed jobs instead of JobTracker (Apache, 2015).

Compare with MapReduce 1.0, the new YARN schema is not only more scalable, it also enable the execution of schemas other than MapReduce on Hadoop clusters, greatly increased the flexibility of the Hadoop Ecosystem (Apache, 2015).

Other components of the Hadoop Ecosystem provide additional analytics features

Even though satisfying the scalability challenge for storing and processing big data, HDFS and YARN by themselves are limited in their functionality. At the same time, HDFS and YARN requires at least several years of Java programming experiences to be property operated, a skill that most data analysts do not posses at the moment (This is according to you but I can't seem to find a good source). In order to further convenient data access, multiple platforms were created to complement HDFS and YARN. Instead of dedicating an entire section detailing each system in detail, we have instead prepared a table offering a brief overview of well-known components of the Hadoop Ecosystem. At the end, we are going to spend some time discussing Spark, a component in the ecosystem that has grown in importance and have the possibility of replacing MapReduce in the future.

Table 4. Hadoop Ecosystem Sub-Projects Overview

Component	Function	Overview
Pig	Script	Pig is a higher-level scripting system that abstracts Java-based MapReduce idioms. It converts complex java scripts into a SQL-like script called Pig Latin. Utilizing Pig Latin, data analyst can access data in each cluster with ease and perform ETL operations, explorative research, and iterative data processing operations without intensive java experiences (HortonWorks, 2015b).
Spark	In-Memory Processing	Apache Spark is an in-memory processing engine for Hadoop that enables significantly faster processing speed than MapReduce . It provides multiple libraries that is able to perform SQL-like queries, real-time graphing, and machine learning algorithms. Spark also supports multiple programming languages including Java, Scala, Python, and R (Spark).
HBase	No-SQL Storage	Hbase is a wide-column big table No-SQL Storage unit that is built on top of Hadoop to provide faster read and write access. More details about Hbase and similar No-SQL databases will be illustrated in the "No-SQL" section of this paper.
Hive	SQL-on-Hadoop	Apache Hive is a data warehouse software used to perform SQL-like query (HiveQL) on top of the Hadoop database. It expands the accessibility of HDFS by providing tools for data summarization, querying, and analysis capabilities (Hive, 2015).
Storm	Data Streaming	Apache Storm is a distributed real-time computation system that is geared towards performing real-time analysis on high-velocity data. It could be implemented with multiple programming languages and is able to achieve speed of one million 100bytes messages per second per node (HortonWorks, 2015a).
Flume	Data Ingestion	Apache Flume is a distributed service for "efficiently collecting, aggregating, and moving a large amount of streaming data [into or out of HDFS) (HortonWorks, 2015a)". It helps organizations to stream massive amount of data from social media, machine sensors,

		and application logs into HDFS.

Spark possess a great threat to MapReduce in the future, but not Hadoop

Even though in the previous section we have described Apache Spark as a sub-project of the Hadoop ecosystem, it has become increasingly an independent project. Due to the in-memory nature of Spark, it is able to achieve significantly faster processing speed over MapReduce when performing the tasks of Machine Learning and Cascading. At the same time, Spark could be implemented not only over Hadoop, but over NoSQL systems such as Cassandra as well. In fact, according to research by TypeSafe, only around 42% of the company implement Spark over YARN, closely trailed by Cassandra and Mesos (TypeSafe, 2015).

Due to the effectiveness of Spark, many have projected that Spark will eventually replace MapReduce and threaten the existence of Hadoop ultimately. However, we believe that this concern is misplaced. First of all, the introduction of YARN has deemphasized the importance of MapReduce in the Hadoop Ecosystem. In fact, the primary demand for Spark right now is to act as a replacement for MapReduce for Hadoop, over the YARN framework. Secondly, in storing a large dataset for Spark to analyze, HDFS provides unparalleled advantage over most of the NoSQL technologies in the market. As long as HDFS's advantage remains, it is likely the existence of Spark will not only threaten the existence of Hadoop, but also increase its attractiveness.

Table 5. Big Data Technologies Vendor Landscape

	Key-Document NoSQL	Key-Value NoSQL	BigTable	Graph Stroage	Hadoop
Leaders	MongoDB, MarkLogic	AWS, Oracle, Aerospike	Datastax(Cassandra), HBase	Neo4j	AWS, IBM, Hortonworks, MapR, Cloudera, Teradata
Strong Performers	CouchBase	Basho(Riak), IBM Cloudant	IBM BigTable	OrientDB	Intel, Microsoft
Contenders	N/A	Berkley DB	N/A	N/A	N/A
Overall Category Popularity	High	Very High	Low	Low	High

*Information based on Mike Gualtieri, N. Y. (2014). The Forrester Wave™: Big Data Hadoop Solutions, Q1 2014, Forrester Research and Yuhanna, N. (2014). The Forrester Wave™: NoSQL Key-Value Databases, Q3 2014, Forrester Research

Hadoop's advantage lies in storing and processing high volume of data from a variety of sources. Practically, Hadoop's function is similar to that of data warehouse but with capability of process and extracting business intelligence from a much larger volume of dataset. Usage cases of Hadoop can be seen from all industries ranging from financial services to healthcare. In this section, we are going to show how Hadoop is utilized by Yahoo to improve many aspects of its business.

Yahoo is the largest user and contributor of Hadoop, with over 42,000 clusters currently in deployment(Baldeschwieler, 2009). As one of the biggest internet company in the world, it takes in over 500 million unique user requests per month and processes billion of transactions per day, all generating petabytes of data per day. Hadoop satisfies Yahoo's needs to process these data with both computing and cost efficiency. Yahoo uses Hadoop to perform many machine-learning functions such as spam filtering and other tasks such as content, advertisement, and search optimization. Utilization of Hadoop and 20 steps of MapReduce have helped Yahoo to shorten processing time of its log data from 26 days to about 20 minutes, illustrating Hadoop's power in high-scale processing (Baldeschwieler, 2009).

NewSQL databases attempt to provide scalability for the relational schema

NewSQL databases were invented in order to preserve the relational schema while offering the scalability NoSQL is able to provide. Compare with the previous two technologies, NewSQL databases are relatively new and less popular. However, NoSQL do shine in fields that require high transactional ACIDity and consistency. NewSQL databases accomplish its goals in two primary ways: modifying the existing relational architecture, or inventing more advanced SQL engine.

New Architecture

Many NewSQL firms attempt to increase scalability of the traditional database by coming up with new database architectures. These databases were usually created with distributional storage and processing in time while geared to maintain the relational nature. In many cases, this is achieved by emphasizing memory-centric processing instead of storage centric processing (NuoDB, 2015). Databases in this category include NuoDB, SAP HANA, and Volt DB.

One of the examples in this category is NuoDB. NuoDB has implemented a new, cloud-based database architecture called the Durable Distributed Cache system. It is a system that has converted the entire database system into sets of in-memory container objects that overflows to disk if it is necessary. The entire system is composed of Transaction Engine, which is in charge of reading and write data in memory, and the Storage managers, which is in charge of putting data that is unused by Transaction Engine into a durable storage.

With this architecture change, NuoDB is able to process at scale significantly faster than the traditional database. As users add new Transaction Engine servers, the processing speed will get faster and faster as all data are in-memory.

Advanced SQL Engine

Another approach NewSQL have taken is to improve the current SQL engine by either coming up with a new query system or improving the storage engine for existing system to increase query speed at scale. Technologies in this category include TokuDB, InnoDB, and InfiniDB. Generally, this type of NewSQL technology acts as drop-in storage engines for corporation's existing systems with almost no need of modification. This makes this type of NewSQL technology particularly attractive to firms that wants to conduct big data analytics but unwilling to make drastic infrastructure investments.

One example in this category is TokuDB, a MySQL storage engine that is created to accelerate querying in SQL by "accelerating replication speed, enabling unparalleled compressing and live schema modification (Tokutek, 2014)". It enables faster insertion and deletions, eliminating the slave lag, and allow the creation of additional indexes during processing. With these technologies, TokuDB is able to query 20x faster without tuning, and is more flexible compare to the traditional SQL databases. Finally, due to better compression technology, TokuDB is able to compress the file up to 90%, reducing the cost for hardware and increase scalability (Tokutek, 2014).

Big data technologies offer distinct advantages in processing various datatypes at various stages

After examining three big data database systems and the traditional relational system, we are now going to put them all together to compare and contrast. A comparison table has been established as Table 1. As illustrated in the table, in regard to ACID compliance and consistency, the traditional database and NewSQL databases poses clear advantage over both Hadoop and NoSQL due to their relationship data models. In regards to scalability, relational database falls off significantly compare with other three "big data" solutions. This makes NoSQL and Hadoop more optimized for large-scale big data analytics.

In regards to implementation, most NoSQL and NewSQL technologies are used for their high processing speed, while Hadoop’s advantage lies in the ability to store data and process them in a large volume. Therefore, it is usually recommended to implement Hadoop alongside with a NoSQL database in order for the YARN parallel computing paradigm to make up for the lack of large-scale analytic power of the NoSQL databases.

Table 6. Big Data Technologies Feature Comparison

Database	Relational	NoSQL	Hadoop	NewSQL
Schema	Relational	Various Schemas	HDFS	Relational

Scalability	Low	High	High	High
ACID Compliance	High	Medium - Low	Medium - Low	High
Adaptation (%)	High	Medium	Medium	Low
Query Format	SQL	Customized depending on providers	Hive	Advanced SQL/SQL
Implementation	Operational Databases	Mostly Operational Databases, Some Data Warehouses	Mostly Data Warehouses With NoSQL Components	Operational Databases
Leading Vendors	Oracle DBMS	MongoDB, CouchDB, Redis	Hortonwork, Cloudrea	NuoDB, VoltDB
Reliability	Medium (Single point of failure)	High	High	High
Optimized Task	Transaction Processing	High Speed Processing	High Volume Storage	High Speed Processing with Relational Schema
Examples (More in passage)	Most traditional retailing firms	MetLife, Ebay, Netflix, Walmart, BestBuy	Obama 2012, T-Mobile, Yahoo	HP, Ericson

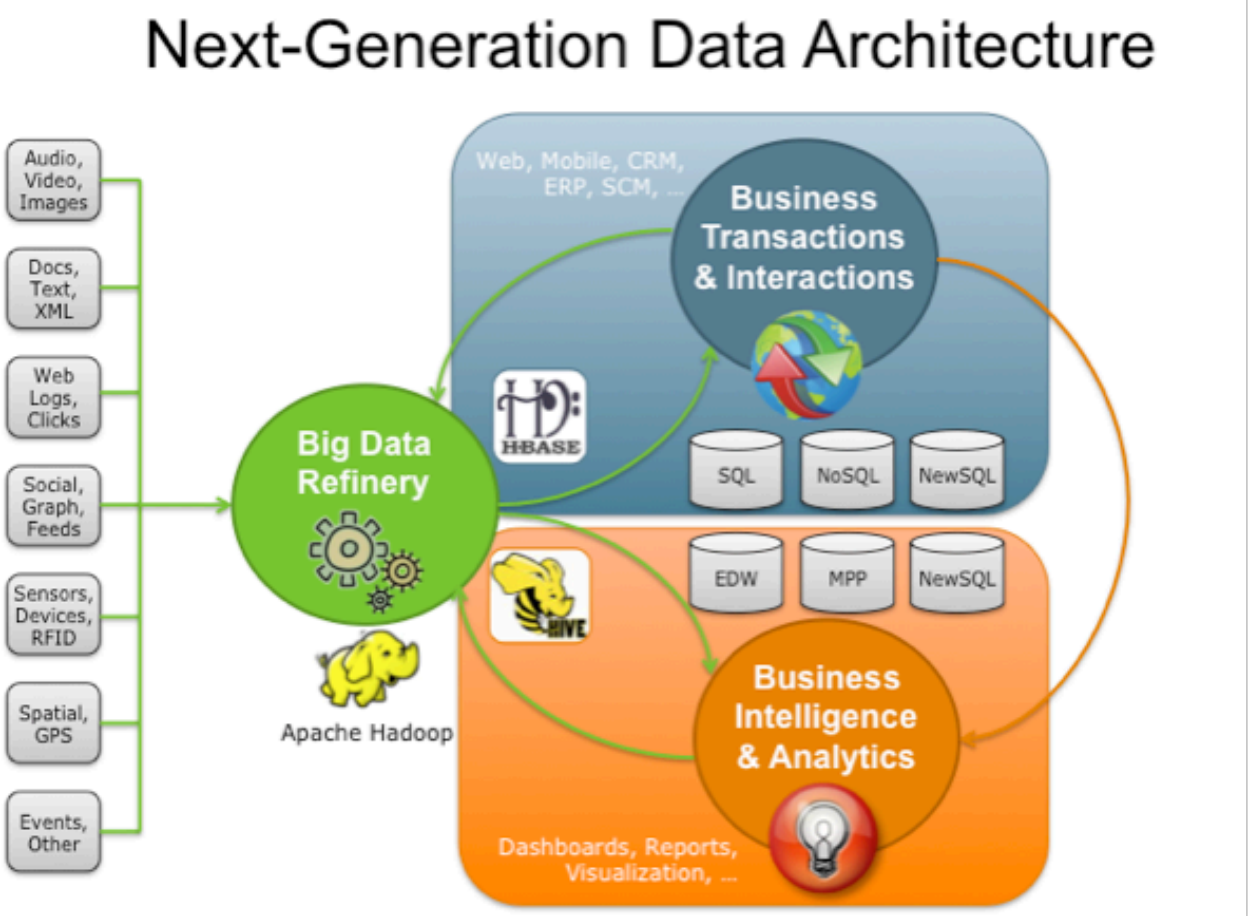
Big Data Technology framework varies but shares several similarities

One aspect exciting for the field of big data is that it is consistently evolving. For this exact same reason, there is not a unique framework for big data technologies within a firm. However, different big data implementation frameworks do share multiple similarities in the function of each component. This section is going to examine those components and illustrate a sample framework for business application.

In most cases, the Hadoop ecosystem serves as a storage and processor of data from all sources. Hadoop not only processes those high volume data and conduct basic analytics on them via the MapReduce Framework, it also prepares and refines structured and unstructured data for NewSQL and NoSQL systems to analyze.

NoSQL systems, on the other hand, serves as a temporary repository of unstructured data while NewSQL systems serve as a temporary repository for high-volume structured data along with relational databases. Those two databases also interacts with the Hadoop ecosystem by either pushing their data to the Hadoop or receiving processed data from Hadoop.

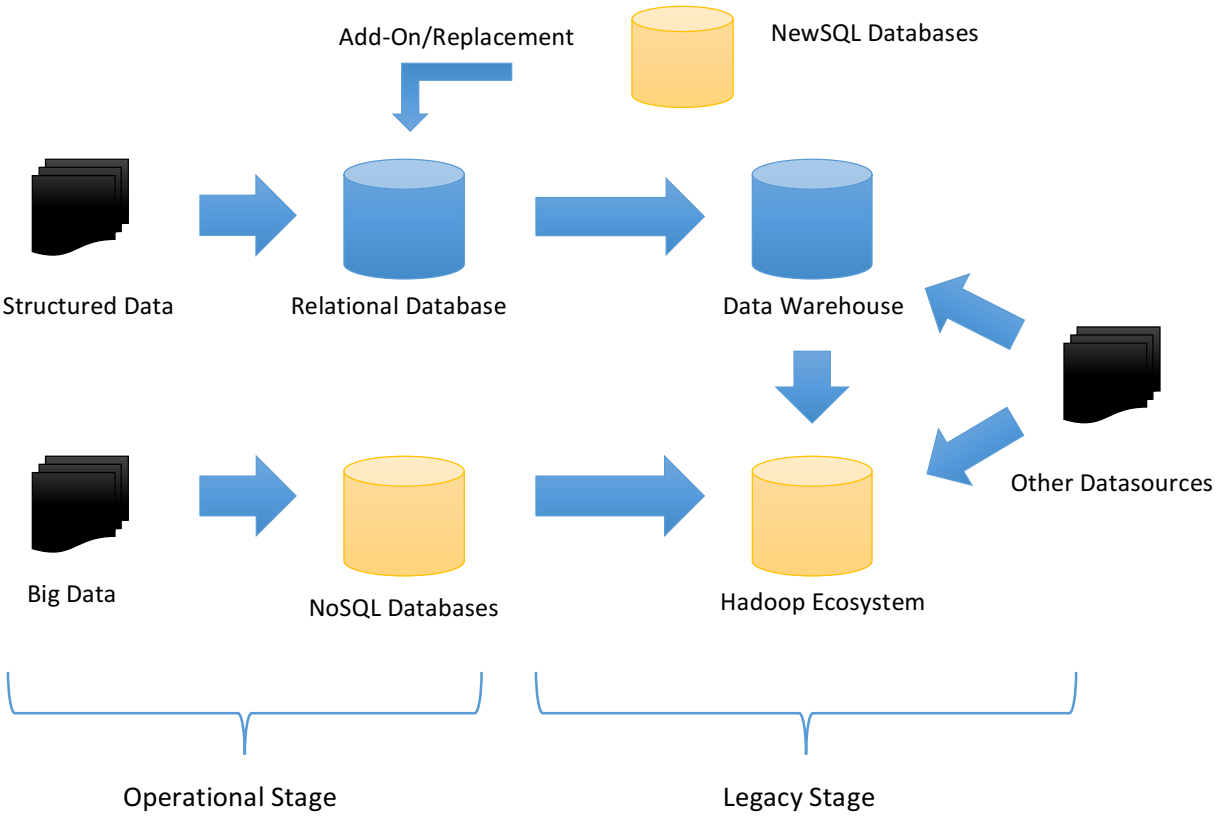
Alt1: Figure 1 illustrates the big data technology framework proposed by Hortonworks. In this case, data from Hadoop are not only processed for transaction and interactions, but also used for business analytics products such as a dashboard and report (Connoly, 2012).



* Connoly, S. (2012). Big Data Refinery Fuels Next-Generation Data Architecture. Retrieved from: http://hortonworks.com/blog/big-data-refinery-fuels-next-generation-data-architecture/

Alt 2: Figure 1 illustrates one framework of how big data technologies are able to assist the existing data lifecycle to analyze and store big data. Overall, NewSQL replaces or serves as an add-on to a corporation's preexisting relational database. It provides scalable analysis of structured operational data while offering hyper-efficient processing speed on top of maintaining ACIDity for data such as financial transactions. NoSQL technologies prevail at collecting and analyzing big data during the operational stage. Through various storage schemas, NoSQL technologies satisfy the storage and processing requirement for unstructured data. Hadoop or similar services serve the company best during the analytic stage. HDFS's distributive schema enables reliable and scalable storage of data from all sources, including the relational database, NoSQL databases, and external data sources. All three systems interact with each other to maximize analytics performance within a firm.

Figure 1. Database Technology Infrastructure



*Graph adapted based on Edlund, J. (2011). *CME Industry Directions: Aligning solutions to Megatrends*.

Cloud-Based Analytics Platforms Provide Big data Analytics for Lower Cost

Our discussion of big data analytics technology will not be considered complete if we do not mention cloud-based analytics platforms. Instead of purchasing their own servers and construct in-house analytics power, increasing amount of companies have started to use platforms already constructed by established internet and service firms such as Amazon and IBM. Outsourcing physical data infrastructures will not only enable companies to cut cost, but also avoid the hassle of maintaining and managing their own physical data center.

Overall, the cloud contributes to big data analytics in two primary ways. First of all, the many cloud-based companies provide Infrastructure-as-service platforms to help organizations manage and storage their data more easily. Instead of building their database systems in-house and purchase expensive servers, companies now are able to use instead cloud-based storage system and process servers such as Amazon S3 and Elastic Computing Cloud 2. Those cloud platforms not only offer the traditional relational data storage services, but also enable NoSQL and Hadoop for companies with more intensive big data storage needs. Moving databases to the cloud will enable companies to integrate their data across organization with more ease, access their data

more rapidly for various business functions, and avoid risks associated with building in-house databases such as server failure (Demirkan & Delen, 2013).

The second types of service offered in cloud are the Analytics-as-service platforms. Those analytics service platforms, such as IBM's Infosphere BigInsight, enable big-data analytics at a higher level of abstraction. With those platforms, companies can perform machine learning and sentiment analysis tasks without intensive acquisition of experts in those specific areas. At the same time, the agility of those analytics-as-services platforms enable companies to perform big data analytics on demand, without worrying about cost associated with idling their big data analytics talents (Demirkan & Delen, 2013).

Table 7 presents major cloud-based analytics platforms. AWS and Microsoft Azure are undeniably the leader for cloud analysis platform with Google and IBM trailing behind as possible future contenders.

Table 7. Cloud Analytics Platform Vendor Landscape

	Market Position	Service Overview	Use Cases	Big Data Technology
AWS	Leader	Amazon Web Service is considered the pioneer in the industry of cloud computing and analytics. It provide a wide range of data storage services and offer multiple analytic services such as Elastic MapReduce and Machine Learning.	Obama 2012 utilizes AWS to perform voter donation prediction and election result predictions. Utilization of AWS contributed to President Obama's victory in 2012 (Lynch, 2012).	Hadoop and Mapreduce via Elastic MapReduce and E2 Stroage. Document-Oriented Storage via DynamoDB
Microsoft Azure	Leader	Microsoft Azure is a close second in regards to cloud computing market share behind AWS. It provides mature data management and storage services and enable its customer to manage SQL databases and conduct machine learning in the cloud.	Heniken used Microsoft Azure to reach over 10.5 million customers during its big campaign in 2012. It managed over 2 million gameplays per hour with latency of 200-300 ms during the campaign (Microsoft, 2015).	Hadoop vis HDInsight Document-Oriented Storage via DocumentDB
Google Cloud	Visionaries	Google cloud platform currently	Google cloud platform supported the 2014 FIFA	MongoDB

Platform		offer three major services: bigquery, dataflow, and pubsub. However, compare to industry leaders, Google cloud platform still lacks comprehensive features and are releasing new features at a slower velocity.	world cup campaign for Coca Cola. It help Coca Cola store and handle image requests during the campaign, and performing tasks such as twitter integration and image transformations(Viveiros, 2014).	Redis Hadoop
IBM	Visionaries	IBM offers a comprehensive cloud analytics and storage solutions ranging from the Infosphere systems to SoftLayer IaaS service. However, due to the fact that SoftLayer is an IBM acquired company, IBM is currently facing integration problems between SoftLayer and other of its data management offerings.	University of Montana utilized IBM infosphere to analyze over 50,00 hours of video electrophalogram recordings to perform real-time traumatic brain injury identification and reduced need to crunch historical data(IBM, 2015).	IBM Hadoop Document-Storage via DB2 NoSQL Mongo DB

*Classification of market position based on Lydia Leong, D. T., Bob Gill (2015). Magic Quadrant for Cloud Infrastructure as a Service, Worldwide.

Big Data Technology Framework Comparison

One interesting fact we have found during our research is that there has not been a unified conceptual framework displaying firms a clear roadmap in implementing big data technology. In this section, we are going to briefly display two prominent big data technology frameworks so corporations can select the one best fit their needs.

This article examined three types of prominent big data technologies and how each technology can contribute to a firm's current data infrastructure to cope with the challenges posed by big data. Overall, NoSQL technologies provided great capacity of storing operational big data while the Hadoop ecosystem excels at analyzing large volume of data. NewSQL, on the other hand,

can serve as a replacement or add-on to the existing relational structure to increase scalability while maintaining transactional ACIDity. Finally, by presenting the fast-growing cloud computing technologies, we have offered readers an alternative to creating their own big data infrastructure from scratch.

Proper use of big data technology can convert big data into invaluable asset for firms of various industries. It is to our sincere wish that this article may serve as a guideline for big data infrastructure applications in firms to provide the best result for businesses.

Apache. (2013). MapReduce Tutorial *Hadoop 1.2.1 Documentation*: Apache.

Apache. (2015). Apache Hadoop NetGen MapReduce (YARN) *Apache Hadoop 2.7.1*: Apache.

Baldeschwieler, E. (2009). *Hadoop at Yahoo!*

Brewer, E. (2000). *Towards Robust Distributed Systems.* Paper presented at the PODC, Portland,Oregon.

Capital, K. (2014). Going Beyond Data: Achieving actionable insights with data and analytics. Retrieved from

Chang, F., Dean, J., Ghemawat, S., Hsieh, W. C., Wallach, D. A., Burrows, M., . . . Gruber, R. E. (2008). Bigtable: A distributed storage system for structured data. *ACM Transactions on Computer Systems (TOCS), 26*(2), 4.

Chiappa, J. N. (2014). "Soft" and "Hard" State. Retrieved from: http://mercury.lcs.mit.edu/~jnc/tech/hard_soft.html

Connoly, S. (2012). Big Data Refinery Fuels Next-Generation Data Architecture. Retrieved from: http://hortonworks.com/blog/big-data-refinery-fuels-next-generation-data-architecture/

Datastax. (2014a). Ebay Engage Customers with Personalized Recommendations. Retrieved from

Datastax. (2014b). Netflix Personalizes Viewing for Over 50 Million Customers with Datastax. Retrieved from

Datastax. (2015). NoSQL Use Cases. from http://www.planetcassandra.org/blog/functional_use_cases/messaging/

Demirkan, H., & Delen, D. (2013). Leveraging the capabilities of service-oriented decision support systems: Putting analytics and big data in cloud. *Decision Support Systems, 55*(1), 412-421.

Eardley, G. (2013). *Tracking MMs of Ganks In Near Real Time.* Paper presented at the StrangLoop 2013.

Edlund, J. (2011). *CME Industry Directions: Aligning solutions to Megatrends*.

Haerder, T., & Reuter, A. (1983). Principles of transaction-oriented database recovery. *ACM Computing Surveys (CSUR), 15*(4), 287-317.

Henschen, D. (2013). MetLife Uses NoSQL For Customer Service Breakthrough. Retrieved from: http://www.informationweek.com/software/information-management/metlife-uses-nosql-for-customer-service-breakthrough/d/d-id/1109919?

Hive, A. (2015). Getting Started. from https://cwiki.apache.org/confluence/display/Hive/GettingStarted

HortonWorks. (2015a). Apache Flume. from http://hortonworks.com/hadoop/flume/

HortonWorks. (2015b). Apache Pig. from http://hortonworks.com/hadoop/pig/

IBM. (2015). University of Montana: Gain lifesaving insights using real-time analytics across high-volume, fast-moving medical data Retrieved from: http://www-01.ibm.com/common/ssi/cgi-bin/ssialias?subtype=AB&infotype=PM&htmlfid=IMC14945USEN&attachment=IMC14945USEN.PDF

Lynch, M. (2012). Barack Obama's Big Data won the US election. Retrieved from: http://www.computerworld.com/article/2492877/government-it/barack-obama-s-big-data-won-the-us-election.html

Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., & Byers, A. H. (2011). Big data: The next frontier for innovation, competition, and productivity.

Marchus, A. (2013). The NoSQL Ecosystem. *I*. Retrieved from The Architecture of Open Source Applications website

Microsoft. (2013). Description of the database normalization basics. 2015

Microsoft. (2015). HEINEKEN Uses the Cloud to Reach 10.5 Million Consumers in Global Marketing Campaign. Retrieved from: https://customers.microsoft.com/Pages/CustomerStory.aspx?recid=17735

MongoDB. (2015). Rethinking The Customer Experience at Metlife: From Stalled to Success in Three Months. Retrieved from: https://www.mongodb.com/customers/metlife

neo4j. (2015). Gamesys bets its next-gen worth on Neo4j, adding advanced social integration. Retrieved from

Netflix. (2015). Netflix's Viewing Data: How We Know Where You Are In House of Cards.

NuoDB. (2015). Durable Distributed Cache Architecture. from http://www.nuodb.com/explore/newsql-cloud-database-ddc-architecture

Oracle. A Relational Database Overview. Java Documentation: Oracle.

Perret, R. (2014). 3 key IT infrastructure requirements for big data and analytics.

Redis. (2015). Introduction to Redis. from http://redis.io/topics/introduction

Roe, C. (2013). ACID vs. BASE: The Shifting pH of Database Transaction Processing. 18. Retrieved from

Spark. Apache Spark. from https://spark.apache.org/

Tokutek. (2014). TokuDB Documentation. from http://docs.tokutek.com/tokudb/

TypeSafe. (2015). Apache Spark: Preparing for the next wave of reactive big data (pp. 24). TypeSafe.

Viveiros, D. (2014). CI&T uses Google Cloud Platform to power the Coca-Cola “Happiness Flag” unveiled on the pitch at the opening match of the 2014 FIFA World Cup™. Retrieved from: http://googlecloudplatform.blogspot.com/2014/06/cit-uses-google-cloud-platform-to-power-the-coca-cola-happiness-flag-unveiled-on-the-pitch-at-the-opening-match-of-the-2014-fifa-world-cup.html

Voldemort. Project Voldemort. from http://www.project-voldemort.com/voldemort/