

Big Data and Saffron Associative Memory BAse

a brief overview

By Paul Hofmann phofmann@saffrontech.com

Intended reader: Anyone who wants to understand big data storage and analytics, and why associative is different – from non-tech executives to analysts

# EXECUTIVE SUMMARY

The term Big Data commonly refers to the management and processing capabilities of huge amounts of data, from hundreds of terabytes above. In the enterprise space such data is usually collected from the underlying databases of business applications and the information storage and stream (e.g. tweets) of web applications (social media, etc.). Recently, big data has been widely addressed by parallel computing techniques, such as key value stores and analytics run by MapReduce.

The challenge for businesses is to leverage Big Data for real time situation sensing and decision-making that anticipates change and risk on the front lines. When data is big, it is very difficult to gain a-priori a clear picture of what “matters” in the data. The three V’s of Big Data– Volume, Variety and Velocity – are challenging to exploit at the same time. Volume and speed are natural opposites while data from diverse sources – structured and unstructured (text, audio and video) – is challenging to analyse with traditional statistical or semantic modelling techniques.

Big Data analytics is entrenched between semantics and statistics, which have been, until now, separate technologies. SaffronMemoryBase ([SMB](http://www.saffrontech.com)) uses an associative memory approach, which mimics the way in which the human brains work, to analyse structured and unstructured data from varied, federated data sources. SMB’s associative memory base finds without knowing, by connecting and correlating all the people, places, things and events in the data creating an elegant knowledge base complete with statistics (counts) and semantics (context). There is no need for data engineers to discover and model the data in advance of its usability for analytics.

# Big Data

The size of Big Data makes it ineffective to work with traditional database management tools. Difficulties include the capture of data from disparate sources, fast changing schemas, storage,search, sharing, analytics, and visualization. The business value of working with larger and larger datasets include the ability to analyse business trends, predict customer behaviour from social (sentiment, motivation and network), predict financial risk, prevent diseases, fight crime and much more.

The trend of Big Data continues because data grows by Moore’s Law (i.e. technology capacity roughly doubles every 18 month). In 2011 we have generated ~ 1.5 zettabytes. In a business context we usually refer to hundreds of terabytes up to petabytes when talking about Big Data. Corporate business data warehouses are considerably smaller, in the order of hundred terabytes rarely more than 200 terabytes (TB). The library of congress is ~ 250 TB including web documents. Big Data is considerably driven by text analytics; text grows 1.5 times faster than the rest. Many Big Data tools have been created to analyse the huge amounts of text found in the World Wide Web (e.g. [MapReduce](file://localhost/Users/Paul/Dropbox/Public/Big%20Data/Big%20Data%20and%20Saffron.docx#_MapReduce) on distributed file systems).

Though a moving target, the current limits of Big Data are on the order of exabytes, mainly for scientific problems like meteorology, genomics, connectomics, complex physics simulations, biological, environmental research, particle physics, etc.

# Associative Memory and Saffron Technology, Inc.

Associative Memory (AM) stores have been around about 20 years, similar to distributed file systems (DFS). Traditional memories store data at a unique address and can then recall that data when presented with the unique address. Associative memories are a form of [content addressable memories](http://en.wikipedia.org/wiki/Content-addressable_memory) in which the object (content) also forms the address used to read and write; similar to hash keys.

AM can be divided into auto-associative memory and hetero-associative memories. Auto-associative memories typically complete patterns and can answer the question, “who is similar to whom?” when presented only with the cities a person has been in, or what languages a person speaks, or which schools he attended, etc. Auto-AM can thus recreate the whole from its parts. Hetero-associative Memories on the other hand can recall an associated piece of data from one category when presented with data from another category. Thus they can be used for classification, scoring, and workflow (best practices). For example lets look at the problem to learn and predict the success of movies. A typical input would be people (director, actor), budget, title (genre), and season of release, while the predicted revenue would be the obvious output. Associating the input vectors (movie data) to the output vector (revenue) is typically solved by a hetero-AM.

AM has been used for a variety of problems like text analytics, risk mitigation, finding best practices (knowledge representation), complex event processing (trending and flows), as well as robot and process control. It’s ability to store knowledge and adapt as new data arrives makes AM a perfect candidate for out of the box machine learning.

Saffron is the best in class Associative Memory. The Saffron memory base - SMB - is a scalable and transactional Big Data store. SMB stores knowledge and patterns. The analytics engine retrieves similar and related data from its memories. SMB learns from and unifies structured and unstructured data (text, numbers, events, etc..). It is schema less and non-functional. It can model combinations of functions and non-linear functions. Saffron memory store deploys fast (in days to weeks) and is very lean; no need to wait for the IT department to set up data cubes and knowledge engineers to define a model.

# Differentiation between Saffron Associative Memory Base (SMB) and other Big Data Techniques

SMB has some properties that make it uniquely suited for analyzing Big Data. SMB is schema less and scales like [MapReduce](file://localhost/Users/Paul/Dropbox/Public/Big%20Data/Big%20Data%20and%20Saffron.docx#_MapReduce) on [DFS](file://localhost/Users/Paul/Dropbox/Public/Big%20Data/Big%20Data%20and%20Saffron.docx#Distributed File Systems) and other distributed key value stores ([Dynamo](http://www.allthingsdistributed.com/files/amazon-dynamo-sosp2007.pdf), [HBase](http://hbase.apache.org/), [BigTable](http://research.google.com/archive/bigtable.html), etc.). SMB is a fully distributed triple store that stores the values of the same category (e.g. person) in a row for faster access during query time. Scalability and schema-less design makes SMB the champion for the unified analytics of huge amounts of structured and unstructured data. SMB is transactional which makes it the ideal tool for real time advanced analytics such as pattern recognition (fraud detection, call center efficiency, etc.) and risk prediction (predictive maintenance, credit risk, etc.). For example, SMB can compare insurance claims or tax returns with known patterns of fraud. But we know that new fraud patterns are constantly emerging, as a new tax return may be the seed of a new fraud pattern. SMB will learn this new fraud pattern instantaneously and suggest that a new fraud pattern has emerged. The batch process nature of MapReduce does not allow real time learning. In principle one can program simple machine learning algorithms with MapReduce, but its batch nature introduces a time lag between the arrival of new data and the learning of new patterns that include the new data.

SMB offers analytical capabilities that are neither found with RDBMS, MapReduce, nor graph databases. Table 1 and 2 highlight some of these differences.

|  |  |
| --- | --- |
| **RDBMS (Tables)** | **Saffron (Matrices)** |
| Table joins for semantics | Pre-joined matrices |
| Predefined schema | Schema-less |
| Limited keys & sorting joins | Everything is a key, globally sorted |
| No natural partitioning | Shared-nothing parallelism |
| Structured data is fact based | Knowledge is more exploitable |
| Nearest-neighbor is infeasible | Nearest-neighbor is instant |

***Table 1: Some Differences between RDBMS and Saffron Associative Memories***

RDBMS is a storage technique specialized for transactional systems. They solve only basic analytics like counts and averages. They cannot be used for advanced analytics techniques like machine learning. More importantly they are hard to distribute over many nodes; thus they don’t scale to Big Data problems. Vendors like Teradata, Netezza, Aster and Greenplum offer RDBMS specialized for parallel queries; none of them scale to Pbytes. RDMBS is well suited for structured data but text and semantic add-ons to RDBMS like [RDF](http://www.w3.org/RDF/), [SPARQL](http://www.w3.org/TR/rdf-sparql-query/), [XQuery](http://www.w3.org/TR/xquery/), etc. don’t perform well.

|  |  |
| --- | --- |
| **MapReduce** | **Saffron Memory Base** |
| Distributed batch processing | Distributed real time transactions |
| New attributes ->code change | Real time update – no manual effort |
| Low level assembler like API | High level, declarative API –> no programming |
| Only very basic machine learning | Advanced instance based machine learning |
| Generic framework | Optimized solution for advanced analytics |

***Table 2: Some Differences between MapReduce and Saffron Associative Memories***

SMB combines statistics and semantics. Statistics and semantics are basically two opposite ways to analyze data and represent the knowledge hidden within the data. Semantics attach meaning to entities by putting them in (a hierarchical) context (i.e. taxonomies or ontologies). Statistics make sense of data by finding inherent correlations between data as well as a given input and output. The combination of semantics and statistics (connections and counts) is at the core of the SMB knowledge store.

Statistics work well with lots of data for finding correlation and co-occurrence between the data. Semantics are very successful if the knowledge represented is concise. Statistics lose significance if the underlying data is very diverse and heterogeneous; if for example the sample size per problem space becomes too small.

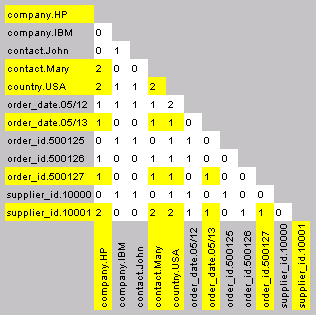
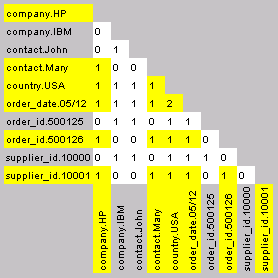
With Big Data the members of a class may be in the thousands. Wouldn’t it be very helpful to have an additional metric to rank the attributes/members within a classification? Saffron’s memory counts deliver a pre-joint rank order for free – no calculation at query time – for thousands of attributes and hundreds of classes. Even better, co-occurrence between several classes is available in real time; besides other measures of relevance like entropy.

For very expressive schemas the traditional semantic approach to text analytics by manually defining the ontologies becomes prohibitively expensive. IBM-Watson needs this kind of authoring while SMB finds the natural context hidden in a text body like the human brain without manual definition of ontologies.

SMB maps data into matrices. Relational schemas as well as semantic graphs derived from text analytics can be stored naturally and very efficiently in matrices for joint retrieval. Graphic 1 and graphic 2 show how SMB maps structured data and text data into matrices thus generating billions of very sparse matrices.

[Gartner called out Saffron as new vendor for advanced analytics on hybrid data](http://www.saffrontech.com/2012/04/06/gartner-big-value-analyzing-diverse-data/) during the Gartner BI Summit 2012 in Los Angeles. Hybrid data comes from structured sources like enterprise applications and the unstructured web sources.

The underlying knowledge in today’s business and mass-market environments is diverse and hetero-geneous. These environments include risk analysis, due diligence, market intelligence, personalized media, e-commerce, advertising, social networking, and vertical domains such as health care. In these environments, the heterogeneity of the underlying schema compromises the quality of the results generated through purely statistical approaches, while the overall scale of the environments makes ontological approaches too expensive.



supplier\_id

date

country

company

contact

500125

10000

05/12

USA

IBM

John

500126

10001

05/12

USA

HP

Mary

500127

10001

05/13

USA

HP

Mary

Order Supplier

1

2

3

Incremental,

non-parametric,

non-functional

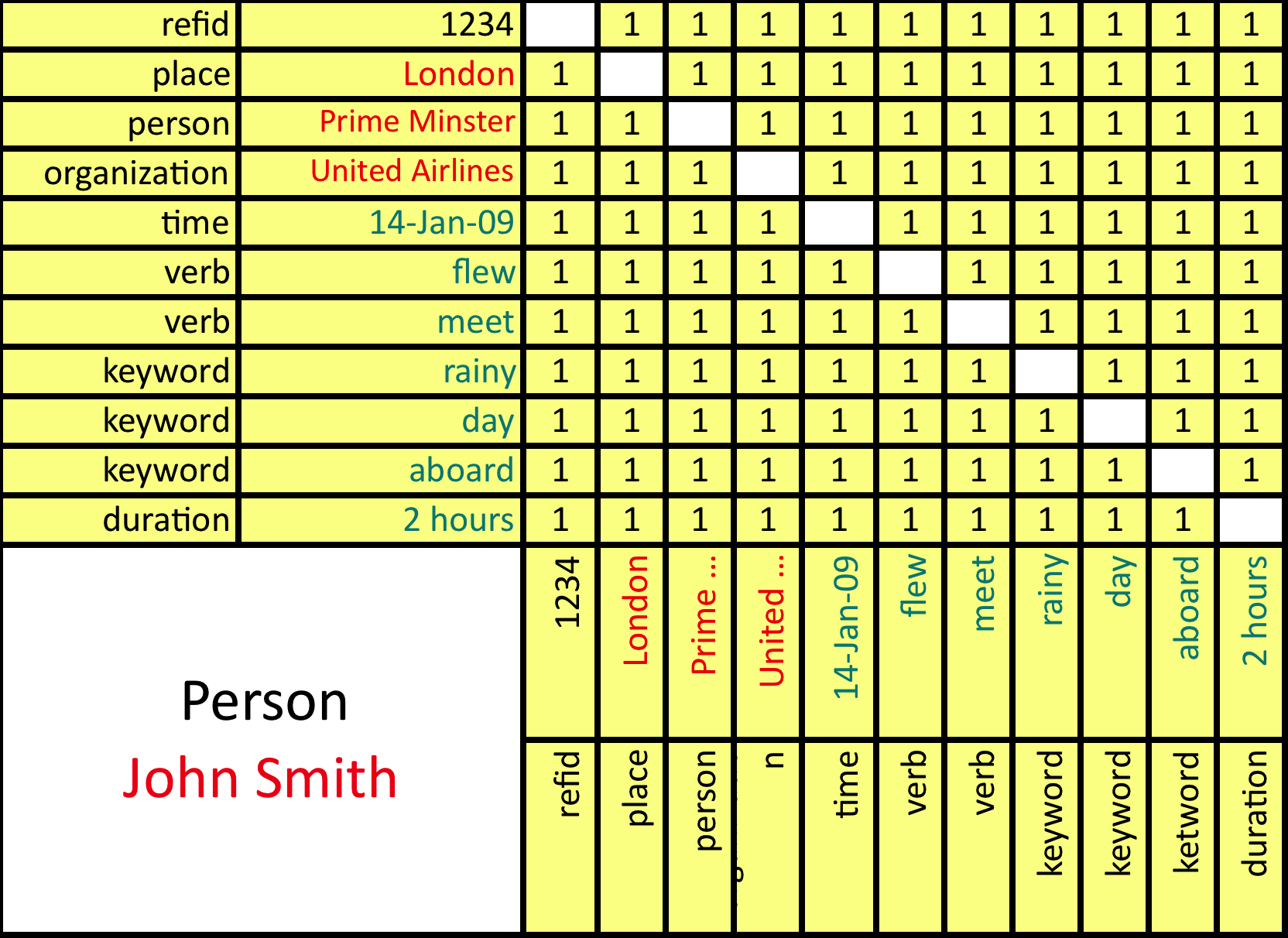
learning

and

novelty-detection

***Graphic 1: Relational Tables can be stored naturally in matrices***

John Smith flew to London on 14 Jan 2009 aboard United Airlines to meet with Prime Minister for 2 hours on a rainy day.



***Graphic 2: Snippet with RefID 1234 - Semantic graphs can be stored naturally in matrices too***

SMB is very well suited to succeed in these markets. SMB combine the best from statistics and semantics to analyze diverse Big Data from structured and unstructured (e.g. text, image and video) sources. It is capable of expressive and complex semantics maintaining scalability at the same time. SMB is schema-less and non-functional. Its model changes in real time allowing for fast changing information and avoiding stale models. Every analyst can use the schema-less and non-functional SMB without having to rely on (and wait for) knowledge modeling by PhDs. Feedback from the end user changes the knowledge representation of SMB in real time. The knowledge of one is now available to many.

Supporting the comprehensiveness of SMB Forrester Research reported in the June 2011 Research Report,*[Dawning of the New Age of the BI DBMS](http://www.saffrontech.com/2011/08/02/the-dawning-of-the-age-of-bi-dbms/)****,* “Use Associative When You Can’t Predict The Future But Have To Prepare For Anything”**. “The future is truly unpredictable. Forrester often hears from DW planners that their business users want “everything” stored and tracked in their DW, just in case. That’s precisely what associative DBMS attempts to do. Imagine a DW that can store all-to-all relationships — associations or vectors — between every entity and every attribute in your domain, with counters, aggregates, and indexes for every intersection!”

# Some SaffronMemoryBase Use Cases

**Our customers are in the DoD, National Security Community, Consumer and Telecommunications and Financial Services.**

**Predicting Threats For The Gates Foundation -- Protecting People, Investment, Reputation and Infrastructure**

The Bill and Melinda Gates Foundation exploits advanced analytics and risk scoring to ensure that its global mission is advanced safely and securely despite the opposition that such high profile organizations attract.

This analytical capability provides a 360 view of known risks and identifies the emerging risks that live within weak signals and noisy data. Our threat scoring system collects and analyzes unsolicited inbound correspondence to provide real-time diagnostic intelligence.

We synthesize a factor and motive based conceptual model from the Rand Corporation, a global risk-analysis think-tank, with behavioral modeling by Saffron Technology's Associative Memory capabilities to operationalize threat prediction. If a correspondence is classified as high threat -- it is automatically routed to The Foundation's protective intelligence specialists. They utilize Saffron to determine root cause and other patterns in order to plan resources and courses of action appropriately.

**Maintenance Repair and Operations at Boeing**

Saffron classifies hundreds of millions of maintenance records from different RDBMS and tribal sources (logs, comments, notes, etc.) within Boeing (~40 data sources) to find replacement parts for legacy aircrafts. Saffron’s associative knowledge base answers questions like, “what are similar parts”, or, “what has been done in a similar situation”. The project costs have been retrieved 10 times within one year due to increased efficiency.

Boeing uses Saffron Technology for predictive analytics too. The pilots’ intuition, sensory recall, complete maintenance records and the mechanics’ knowledge and experience are unified and stored by the associative memory base. Saffron then identifies signals and patterns within the data surfacing unusual maintenance events to the user for further root cause analysis. Its capability to learn from “one” and apply to others based on similarity allows to predict when parts should be replaced before they would break. Helicopters that fly through sandy terrain for example have shorter maintenance cycles than those that are exposed to less abrasive conditions. Saffron finds by similarity analysis the choppers that are on a “non-sandy” maintenance cycle but have been exposed already too much to sand for their parts to make it till the next scheduled maintenance. Before using Saffron Memory Base for predictive maintenance the prediction accuracy has been 63% with 18% false positives, i.e. 18% of parts have been predicted to break though they were still ok. Saffron has achieved 100% prediction accuracy reducing the false positives to 2% at the same time.

**Financial Risk** - Saffron searches and analyzes in real time large volumes of structured and unstructured data from disparate databases, including sources from the web, to allow analysts and information workers to determine how a proposed investment or acquisition compares to similar past investments. Saffron’s similarity analysis continuously adjusts its model to include changing market conditions and trends. Saffron’s knowledge representation, free of aging rules and models, automatically adapts as new information arrives in real time.

**Customer Loyalty -** Saffron can be used for customer loyalty analysis in any industry where there are large numbers of customers yet customer personalization matters. For example, Telco churn analysis which includes classifying customers of Telco products into customers who are loyal and customers who churn provided some event or combination of events or causes. Saffron learns on the fly and develops sophisticated customer loyalty scenarios. SMB can identify discrete customer classes, their social interaction and much more.

**Insurance and Financial Fraud Detection and Prevention** - Saffron identifies fraud in real time by comparing incoming insurance claims to prior examples of fraud in its constantly evolving knowledge base. Saffron can discern new emerging fraud patterns in real time. These new patterns are added to Saffron’s knowledge base without the need to change any models or rules.

**Complex Event Processing** – Saffron’s index is consistent over time. SMB has versioning and time slicing built in. It discerns trends and episodes – when has that or a similar pattern happened before. SMB can also predict patterns.

# Business Summary of Saffron Memory Base

SaffronAdvantage, Saffron’s All Source Intelligence application, delivers deep data exploitation without the need for data scientists, making it the perfect tool for knowledge workers whether they are business development teams, strategic sales teams, financial market analysts, fraud investigators or compliance managers in insurance or financial companies. Information and Knowledge workers quickly make sense of new and interesting connections occurring in their world of data and can further explore these connections using Saffron’s analytic reasoning methods for connections, similarities, episodic patterns, temporal trends and geo associations. Real-time data exploitation of unified structured and unstructured data to see the connections between every person, place, thing, event, and outcome in the context of time, space, and networks helps you quickly see the affect of new relationships, similarities and time based patterns in real time as data arrives.

# SHORT EXPLANATION OF SOME BIG DATA TERMINOLOGY AND BUSINESS INTELLIGENCE

# Distributed File Systems

Commercially available distributed file systems (DFS) go back to the mid 70s (DEC), mid 80s (Sun) and mid 90s (IBM). Today, the most prominent DFS is the Hadoop Distributed File System ([HDFS](http://en.wikipedia.org/wiki/Apache_Hadoop)). Hadoop’s popularity is mainly based on the fact that is licensed for free by the Apache SW Foundation as opposed to its proprietary siblings – Google File System ([GFS](http://www.cs.brown.edu/courses/cs295-11/2006/gfs.pdf)) and IBM General Parallel File System ([GPFS](http://www-03.ibm.com/systems/software/gpfs/)). The Apache Foundation does not pay any of the HDFS programmers. It is important to note that the customer or user has no-influence on when bugs will be removed due to the meritocratic business model of Apache. Besides HDFS there are other DFS like [CEPH](http://en.wikipedia.org/wiki/Ceph) (from UCSC developed for scientific applications), Gopher, [ClusterFS](http://gluster.org/community/documentation/index.php/GlusterFS_Concepts) and many more.

HDFS is a distributed, scalable, and portable file system written in Java for the Hadoop framework. Each node in a Hadoop instance typically has a single data node. HDFS is cluster of data nodes. Each data node stores and delivers blocks of data over the network using a block protocol specific to HDFS. HDFS uses the TCP/IP layer for communication while the clients use remote procedure call (RPC) to communicate between each other. HDFS stores large files across multiple machines. Reliability comes from replicating the data across multiple nodes. When a limited number of nodes in a file system go offline, the system continues to work without any data loss. For example, the default replication value of 3 means that data is stored on three nodes: two on the same rack, and one on a different rack. Data nodes talk to each other to rebalance data, to move copies around, and to maintain a high data replication.

HDFS does not provide high availability because an HDFS file system instance requires one unique server, the name node. This is a single point of failure for an HDFS installation. That means data stored on HDFS can be lost (without recovery) which makes it less attractive to companies for business use. Google’s GFS, and IBM’s GPFS and other commercially available products provide high availability.

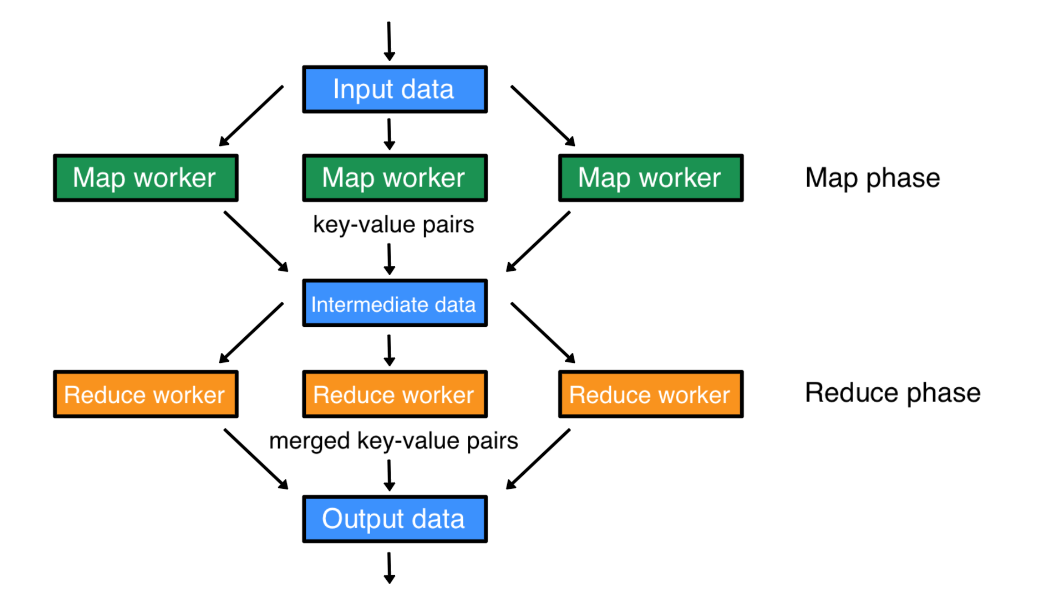
# MapReduce

MapReduce is a framework for processing highly distributable problems across huge datasets using a large number of computers.

**Map" step:** The master node takes the input (e.g. documents) typically from a DFS, partitions it up into smaller sub-problems and distributes them to worker nodes. The worker node processes the smaller problem (e.g. number of occurrence of each word in each document), and passes the answer back to its master node.

**"Reduce" step:** The master node then collects the answers (e.g. lists of words and their partial counts) to all the sub-problems and combines them in some way to form the output (e.g. list of words and their counts for all input documents).

MapReduce allows for distributed processing of the map and the reduce operations. Provided each mapping operation is independent of the others, all maps can be performed in parallel. Similarly, a set of 'reducers' can perform the data reduction phase. While this process can often appear inefficient compared to more sequential algorithms, MapReduce can be applied to significantly larger datasets than legacy data bases can handle. To this end, a large server farm can use MapReduce to sort a petabyte of data in a few hours. MapReduce’s parallelism also offers the possibility of recovering from partial failure of servers or storage during the operation: if one mapper or reducer fails, the work can be rescheduled – assuming the input data is still available.

***Graphic 3: Schematic MapReduce Steps***

MapReduce achieves reliability by sharding the number of operations on the data set to each node in the network. Each node is expected to report back periodically with completed work and status updates. If a node falls silent for longer than that interval, the master node records the given node as dead and sends out the node's assigned work to other nodes.

Implementations of MapReduce are not necessarily highly reliable. When using Hadoop as the DFS the name node becomes a single point of failure as we have seen above.

MapReduce has been applied successfully to a wide range of applications such as, distributed sort, web link-graph reversal, web access log stats, inverted index construction, document clustering, machine learning, statistical machine translation, and more.

MapReduce is a batch procedure for analysing huge amounts of data, however, it cannot be used for transactional systems like accounting, inventory management or fraud detection as Hadoop does not index nor provide relationship or transactional guarantees. MapReduce [has been criticized](http://craig-henderson.blogspot.com/2009/11/dewitt-and-stonebrakers-mapreduce-major.html) for the lack of breadth of problems to which it can be applied and for its low level interface as opposed to a high level and reusable SQL interface for RDBMS. MapReduce's use of input files and lack of schema support prevents the performance improvements enabled by common database system features such as B-trees and hash partitioning. DeWitt and Stonebraker have published a detailed benchmark study in 2009 that compares the performance of Hadoop's MapReduce to cluster RDBMS approaches in several specific problem areas.In this study, [MapReduce was slower than typical relational cluster databases](http://database.cs.brown.edu/projects/mapreduce-vs-dbms/) (e.g.Teradata, Greenplum, IBM BD2, Oracle Exadata, Aster Data, Vertica, etc.).

[Apache Hadoope](http://hadoop.apache.org/mapreduce/) is the open source implementation of MapReduce on Hadoop File System sponsored by the Apache SW Foundation.

# Distributed Key Value Stores

The term distributed data store is generally used for NoSQL databases that access huge amount of data on a large number of nodes. Many modern NoSQL databases run on distributed file systems in order to scale to many hundreds or even thousands of nodes. For example Google’s [Big Table](http://research.google.com/archive/bigtable.html) database runs on Google File System (GFS), while [HBase](http://hbase.apache.org/) - its open source cousin - runs on HDFS. Other examples for distributed key value stores are [Amazon](http://en.wikipedia.org/wiki/Amazon.com)'s [Dynamo](http://www.allthingsdistributed.com/files/amazon-dynamo-sosp2007.pdf), MS [Windows Azure Storage](http://en.wikipedia.org/wiki/Azure_Services_Platform), Apache’s [Cassandra](http://cassandra.apache.org/) used by Facebook for its non-relational data sore, or LinkedIn’s Voldemor. Dynamo for example is a highly available, distributed key-value data store. It has properties of both databases and distributed hash tables (DHTs). Dynamo delivers reliability; it always finds the shopping cart number even if some of the nodes that store this key value are down. Dynamo is scalable since it’s distributed. According to the CAP theorem it can be only eventually consistent. Therefore one would not use Dynamo for accounting or generally for inventory management. Taking for example 100 SKUs (stock keeping units) out of 1000 in inventory will show in a Dynamo implementation still 1000 SKUs instead of 900. Only after a certain time period depending on the parameters used (typically 1 to 2 hours with Amazon’s seller accounts) will the Dynamo store show 900 SKUs.

# Relational Data Base Systems

A relational database is a [database](http://en.wikipedia.org/wiki/Database) that conforms to the [relational model](http://en.wikipedia.org/wiki/Relational_model) theory. The software used in a relational database is called a [relational database management system](http://en.wikipedia.org/wiki/Relational_database#Relational_database_management_systems) (RDBMS). Colloquial use of the term "relational database" may refer to the RDBMS software, or the relational database itself. Nowadays, the relational database is by far the predominant choice for storing and analysing data, over other models like the [hierarchical database model](http://en.wikipedia.org/wiki/Hierarchical_database_model), the network model ([CODASYL](http://en.wikipedia.org/wiki/CODASYL)), or the graph model. Most business applications like ERP, CRM, or SCM run on top of an RDBMS. The total RDBMS and data integration market, today, is about $ 34 B.

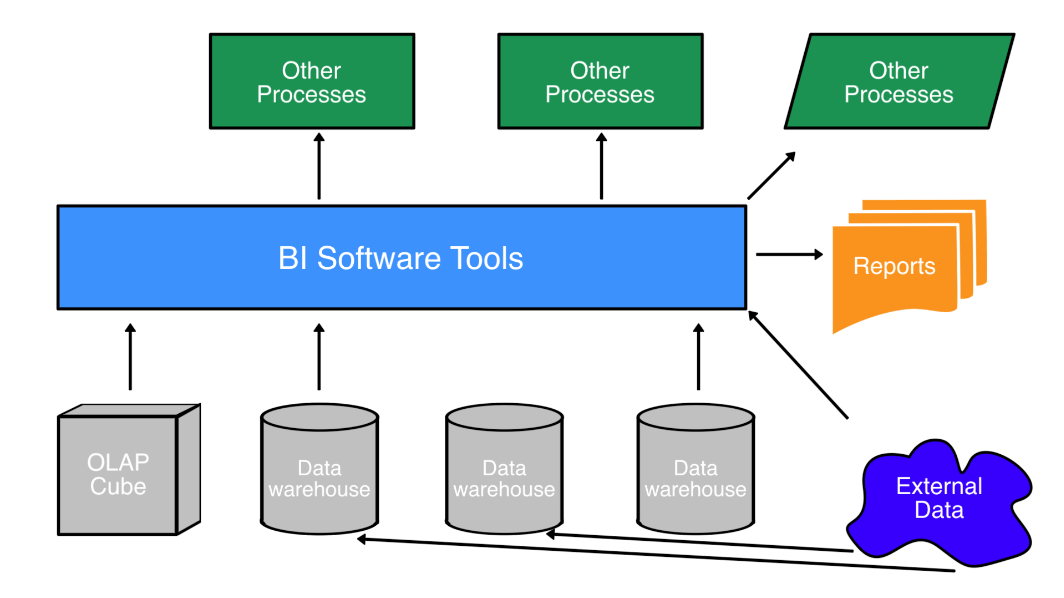
RDBMS offers referential integrity, reliability and consistency (ACID) making them ideal for accounting, inventory management systems, etc.. Their main draw backs are: they can’t be distributed on many thousand nodes and thus don’t scale for Big Data like DFS and their schema has to be predefined and is hard to change. RDBMS were not very successful for storing unstructured or semantic data. Many RDBMS add-ons for text (XML) and RDF (metadata data model for web resources) are too slow for widespread use.

There is an important trade off to be observed for data storage. The CAP Theorem states that it is not possible to have [consistency](http://en.wikipedia.org/wiki/Consistency_(database_systems)), [availability](http://en.wikipedia.org/wiki/Availability), and partition tolerance of the network, all at the same time. You have to choose two out of the three. Key value stores offer distributed high-speed read/write access and availability with reduced consistency. Whereas traditional relational data base systems (RDBMS) offer high availability and consistency but cannot be distributed (easily) for scalability. RDBMS limits the size of data storage and query size. When the data is updated frequently, traditional relational database engines are not well suited to answering complex queries within a very short timeframe. For example, Amadeus (the EU counterpart to SABRE) [gave up using RDBMS for its airline booking system](http://www.systems.ethz.ch/Talks/alonso/SwissBox-Humboldt-Dec-10.pdf) and replaced it with [Crescando](http://www.dbis.ethz.ch/research/publications/sigmod10-crescando.pdf), an in-memory data storage developed by ETH Zurich. Amadeus needs answers in real time to decision support queries like, “give me the number of first-class passengers in wheelchairs who will depart from Tokyo to any destination in the U.S. tomorrow” while the data storage used is updated several hundred times a second with new flight bookings.

Similarly, SABRE doesn’t use a RDBMS and is instead deploying the distributed graph DB [Versant](http://www.versant.com/index.aspx).

# Business Intelligence (BI)

BI mainly refers to techniques to identify, extract and analyse business data, such as sales revenue by products and/or departments, or by associated costs and incomes. BI provides historical, current and predictive views of business operations. Typically, BI applications use data gathered from a data warehouse (RDBMS) via an ETL (extract, transform and load) mechanism. Common functions of business intelligence technologies are reporting, online analytical processing (OLAP), data mining, process mining, complex event processing, business performance management, benchmarking, text mining, and predictive analytics. BI aims to support better business decision-making.

***Graphic 4: Business Intelligence Platform***

Traditional BI systems have been recently [criticized for being too slow to adapt to changing business requirements](http://www.saffrontech.com/wp-content/uploads/2011/08/dawning_of_age_of_bi_dbms_pagechange.pdf). BI designs require knowledge of “what’s in the data” and the use of pre-designed models that are subject to aging as the business advances or as the world changes around it.

In any given enterprise there are hundreds of applications (ERP, CRM, SCM, MRO, process control, sensors, electronic log books, email, blogs, text documents, spread sheets, etc.) generating data, content and communication records. Traditional BI approaches require controlled and cleaned datasets for analytics tools to work. This rigidity of BI is aggravated by the fact that the most valuable information about e.g. customers, their preferences and buying behaviour has to be discerned from the tribal knowledge hidden in social networks, Internet communities, tweets and other web sources. Only the combination (correlation, association, and co-occurrence) of in-house data with the context provided by social and interest networks will yield the return expected when billions of dollars had been spent for BI. “Information no longer “arrives” … it is always in use by the communal mind”, Gartner, BI Summit 2012, LA.

# Vocabulary/Abbreviations

ACID – (atomicity, consistency, isolation, durability) is a set of properties that guarantee that database transactions are processed reliably.

* Atomicity requires that each transaction is “all or nothing”: if one part of the transaction fails, the entire transaction fails, and the database state is left unchanged. This guarantees that a transaction cannot be left in an incomplete state.
* Consistency property ensures that any transaction will bring the database from one valid state to another. Any data written to the database must be valid according to all defined rules.
* Isolation refers to the requirement that no transaction should be able to interfere with another transaction, achieved by ensuring that no transaction that affect the same rows can run concurrently.
* Durability property indicates that once a transaction has been committed, it will remain so, meaning the results are stored permanently.

CAP Theorem – (aka Brewer’s Theorem) states that it is impossible for a distributed computer system to simultaneously provide all three of the following guarantees: consistency (all nodes see the same data at the same time), availability ( a guarantee that every request receives a response about whether it was successful or railed), Partition tolerance (the system continues to operate despite arbitrary message loss or failure of part of the system). However any two can satisfy these guarantees at the same time.

ETL – (extract, transform and load) is a process in database usage and especially in data warehousing that involves: extracting data from outside sources, transforming it to fit operational needs, and loading it into the end target.

NoSQL – a variety of non-relational databases that are used for handling huge amounts of data in the multi-terabyte and petabyte range. Rather than the strict table/row structure of the relational databases that are widely used in all enterprises, NoSQL databases are field oriented and more flexible for many applications. They are said to scale horizontally, which means that inconsistent amounts of data can be stored in an individual item/record (the equivalent of a relational row).

RDF – (resource description framework) is a family of World Wide Web Consortium specifications originally designed as a metadata data model. It is based upon the idea of making statements about resources (in particular Web resources) in the form of subject-predicate-object expressions.

Sharding – a “shared-nothing” partitioning scheme for large databases across a number of servers. Sharding breaks your database into smaller chunks “shards” and spreads these shards across a number of distributed servers.

XML – (extensible markup language) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable emphasizing simplicity, generality, and usability over the internet.