Outline: Combining Brainstorming Deliverables

Table of Contents

1. Introduction and Definition
2. Reference Architecture and Taxonomy
3. Requirements, Gap Analysis, and Suggested Best Practices
4. Future Directions and Roadmap
5. Security and Privacy - 10 Top Challenges
6. Conclusions and General Advice
Appendix A. Terminology Glossary
Appendix B. Solutions Glossary
Appendix C. Use Case Examples
Appendix D. Actors and Roles

1. Introduction and Definition

The purpose of this outline is to illustrate how some initial brainstorming documents might be pulled together into an integrated deliverable. The outline will follow the diagram below.

Section 1 introduces a definition of Big Data. An extended terminology Glossary is found in Appendix A. In section 2, a Reference Architecture diagram is presented followed by a taxonomy describing and extending the elements of the Reference Architecture. Section 3 maps requirements from use case building blocks to the Reference Architecture. A description of the requirement, a gap analysis, and suggested best practice is included with each mapping. In Section 4 future improvements in Big Data technology are mapped to the Reference Architecture. An initial Technology Roadmap is created on the requirements and gap analysis in Section 3 and the expected future improvements from Section 4. Section 5 is a placeholder for an extended discussion of Security and Privacy. Section 6 gives an example of some general advice. The Appendices provide Big Data terminology and solutions glossaries, Use Case Examples, and some possible Actors and Roles.
**Big Data Definition** - “Big Data refers to the new technologies and applications introduced to handle increasing Volumes of data while enhancing data utilization capabilities such as Variety, Velocity, Variability, Veracity, and Value.”

The key attribute is the large Volume of data available that forces horizontal scalability of storage and processing and has implications for all the other V-attributes. It should be noted that the other V-attributes were present before the introduction of “Big Data”. (For example, non-relational databases are not a new idea.) It is the combination of these attributes with required horizontal scalability that requires new technology.

**Some implications of the V-attributes implications are given below:**

- **Volume** - Key driving requirement for robust horizontal scalability of storage/processing
- **Variety** - Driving move to non-relational data models (e.g. key-value)
- **Variability** - Driving need for adaptive infrastructure
- **Value** - Driving need for new querying and analytics tools
- **Veracity** - Driving need for ensuring trust in the accuracy, relevance, and security of data
- **Velocity** - Driving many optimization such as key-based interfaces, high availability, in memory databases, columnar storage, SSD storage, and parallel stream processing

2. Reference Architecture and Taxonomy

The Reference Architecture below will help focus the discussion of other deliverables.
Requirements, capabilities, and gap analysis for Big Data technologies will be mapped to elements of the Reference Architecture. This will enable the development of a detailed Technology Roadmap across the entire Big Data space.

A Reference Architecture Taxonomy will provide descriptions and extensions for the elements of the Reference Architecture. Elements in the Reference Architecture Diagram are in bold.

1. Applications -

2. Design, Develop, Deploy Tools -
   http://gigaom.com/2012/12/18/a-programmers-guide-to-big-data-12-tools-to-know/

3. Security and Privacy -

4. Analytics and Interfaces -
   http://www.techrepublic.com/blog/big-data-analytics
   4.1 Complex Analytics
      4.1.1 Real-time
      http://en.wikipedia.org/wiki/Stream_processing
      4.1.1.1 Complex Event Processing
      http://en.wikipedia.org/wiki/Complex_event_processing
      4.1.2 Interactive
      4.1.3 Batch
      http://datatactics.blogspot.com/2013/02/batch-versus-streaming-differentiating.html
      4.1.3.1 Machine Learning
      http://stackoverflow.com/questions/13760967/machine-learning-big-data
   4.2 Interfaces
      4.2.1 Non-SQL
      http://www.dataversity.net/unql-a-standardized-query-language-for-nosql-databases/
      4.2.2 SQL
      4.2.2.1 To Filesystem
      4.2.2.2 To NoSQL Database
      http://databasesincloud.wordpress.com/2011/05/16/talking-sql-to-nosql-data-stores/
   4.3 Visualization
      http://gigaom.com/2013/05/13/visualization-is-the-future-6-startups-re-imagining-how-we-consume-data/
   4.4 Business Intelligence

5. System and Process Management -
   5.1 Systems Management
   http://incubator.apache.org/ambari/
   5.2 Process Management
   http://oozie.apache.org/
6. Data Processing within the Architecture Framework -
   6.1 ETL
   http://en.wikipedia.org/wiki/Extract,_transform,_load
   6.2 Data Serialization

7. Data Governance -

8. Data Stores -
   8.1 File Systems
   8.2 Databases
      8.2.1 Operational
         8.2.1.1 NoSQL Databases
         https://en.wikipedia.org/wiki/NoSQL
         8.2.1.1 Column-oriented
         http://en.wikipedia.org/wiki/Column-oriented_DBMS
         8.2.1.2 Document
         https://en.wikipedia.org/wiki/NoSQL#Document_store
         Examples: MongoDB, CouchDB
         8.2.1.1 Graphical
         https://en.wikipedia.org/wiki/NoSQL#Graph
         8.2.1.1 Key-Value
         https://en.wikipedia.org/wiki/NoSQL#Key.E2.80.93value_store
         8.2.1.2 NewSQL Databases
         http://en.wikipedia.org/wiki/NewSQL
     8.2.2 Analytic
     http://searchbusinessanalytics.techtarget.com/definition/analytic-database
     8.2.2.1 EDW
     http://en.wikipedia.org/wiki/Data_warehouse
     8.2.3 In Memory or Solid State Drive (SSD) resident data bases

9. IO External to Architecture Framework and Stream Processing -
   http://flume.apache.org/

10. Infrastructure
    10.1 Appliances
    10.2 Internal Server Farm
    http://en.wikipedia.org/wiki/Server_farm
    10.3 Data Grids and Fabrics
    10.4 Cloud-based
    http://readwrite.com/2013/06/07/how-cloud-computing-democratizes-big-data
3. Requirements, Gap Analysis, and Suggested Best Practices

In the Requirements discussion, building block components for use cases will be mapped to elements of the Reference. These components will occur in many use cases across multiple application domains. A short description, possible requirements, gap analysis, and suggested best practices is provided for each building block.

1. Data input and output to Big Data File System (ETL, ELT)

*Example Diagram:*

![Diagram](image)

*Description:* The Foundation Data Store can be used as a repository for very large amounts of data (structured, unstructured, semi-structured). This data can be imported and exported to external data sources using data integration middleware.

*Possible Requirements:* The data integration middleware should be able to do high performance extraction, transformation and load operations for diverse data models and formats.

*Gap Analysis:* The technology for fast ETL to external data sources (e.g. Apache Flume, Apache Sqoop) is available for most current data flows. There could be problems in the future as the size of data flows increases (e.g. LHC). This may require some filtering or summation to avoid overloading storage and processing capabilities.

*Suggested Best Practices:* Use packages that support data integration. Be aware of the possibilities for Extract-Load-Transform (ELT) where transformations can be done using data processing software after the raw data has been loaded into the data store e.g., Map-Reduce processing on top of HDFS.
2. Data exported to Databases from Big Data File System

Example Diagram:

Description: A data processing system can extract, transform, and transmit data to operational and analytic databases.

Possible Requirements: For good through-put performance on very large data sets, the data processing system will require multi-stage parallel processing.

Gap Analysis: Technology for ETL is available (e.g. Apache Sqoop for relational databases, MapReduce processing of files). However if high performance multiple passes through the data are necessary, it will be necessary to avoid rewriting intermediate results to files as is done by the original implementations of MapReduce.

Suggested Best Practices: Consider using data processing that does not need to write intermediate results to files e.g. Spark.
3 Big Data File Systems as a data resource for batch and interactive queries

**Example Diagram:**

![Diagram](image)

*Description:* The foundation data store can be queried through interfaces using batch data processing or direct foundation store access.

**Possible Requirements:** The interfaces should provide good throughput performance for batch queries and low latency performance for direct interactive queries.

**Gap Analysis:** Optimizations will be necessary in the internal format for file storage to provide high performance (e.g. Hortonworks ORC files, Cloudera Parquet)

**Suggested Best Practices:** If performance is required, use optimizations for file formats within the foundation data store. If multiple processing steps are required, data processing packages that retain intermediate values in memory.
4. Batch Analytics on Big Data File System using Big Data Parallel Processing

Example Diagram:

Description: A data processing system augmented by user defined functions can perform batch analytics on data sets stored in the foundation data store.

Possible Requirements: High performance data processing is needed for efficient analytics.

Gap Analysis: Analytics will often use multiple passes through the data. High performance will require the processing engine to avoid writing intermediate results to files as is done in the original version of MapReduce.

Suggested Best Practices: If possible, intermediate results of iterations should be kept in memory. Consider moving data to be analyzed into memory or an analytics optimized database.
5. Stream Processing and ETL

Example Diagram:

Description: Stream processing software can transform, process, and route data to databases and real time analytics.

Possible Requirements: The stream processing software should be capable of high performance processing of large high velocity data streams.

Gap Analysis: Many stream processing solutions are available. In the future, complex analytics will be necessary to enable stream process to perform accurate filtering and summation of very large data streams.

Suggested Best Practices: Parallel processing is necessary for good performance on large data streams.
6. Real Time Analytics (e.g. Complex Event Processing)

Description: Large high velocity data streams and notifications from in memory operational databases can be analyzed to detect pre-determined patterns, discover new relationships, and provide predictive analytics.

Possible Requirements: Efficient algorithms for pattern matching and/or machine learning are necessary.

Gap Analysis: There are many solutions available for complex event processing. It would be useful to have standards for describing event patterns to enable portability.

Suggested Best Practices: Evaluate commercial packages to determine the best fit for your application.
7. NoSQL (and NewSQL) DBs as operational databases for large-scale updates and queries

Example Diagram:

Description: Non-relational databases can be used for high performance for large data volumes (e.g. horizontally scaled). New SQL databases support horizontal scalability within the relational model.

Possible Requirements: It is necessary to decide on the level of consistency vs. availability is needed since the CAP theorem demonstrates that both can not be achieved in horizontally scaled systems.

Gap Analysis: The first generation of horizontal scaled databases emphasized availability over consistency. The current trend seems to be toward increasing the role of consistency. In some cases (e.g. Apache Cassandra), it is possible to adjust the balance between consistency and availability.

Suggested Best Practices: Horizontally scalable databases are experiencing rapid advances in performance and functionality. Choices should be based on application requirements and evaluation testing. Be very careful about choosing a cutting edge solution that has not been used in applications similar to your use case. SQL (or SQL-like) interfaces will better enable future portability until there are standards for NoSQL interfaces.
8. NoSQL DBs for storing diverse data types

Example Diagram:

Description: Non-relational databases can store diverse data types (e.g. documents, graphs, heterogeneous rows) that can be retrieved by key or queries.

Possible Requirements: The data types to be stored depend on application data usage requirements and query patterns.

Gap Analysis: In general, the NoSQL databases are not tuned for analytic applications.

Suggested Best Practices: There is a trade off when using non-relational databases. Usually some functionality is given up (e.g. joins, referential integrity) in exchange for some advantages (e.g. higher availability, better performance). Be sure that the trade-off meets application requirements.
9. Databases optimized for complex ad hoc queries

Example Diagram:

Description: Interactive ad hoc queries and analytics to specialized databases are key Big Data capabilities

Possible Requirements: Analytic databases need high performance on complex queries which require optimizations such as columnar storage, in memory caches, and star schema data models.

Gap Analysis: There is a need for embedded analytics and/or specialized databases for complex analytics applications.

Suggested Best Practices: Use databases that have been optimized for analytics and/or support embedded analytics. It will often be necessary to move data from operational databases and/or foundation data stores using ETL tools.
10. Databases optimized for rapid updates and retrieval (e.g. in memory or SSD)

Example Diagram:

Description: Very high performance operational databases are necessary for some large-scale applications.

Possible Requirements: Very high performance will often require in memory databases and/or solid state drive (SSD) storage.

Gap Analysis: Data retrieval from disk files is extremely slow compared in memory, cache, or SSD access. There will be increased need for these faster options as performance requirements increase.

Suggested Best Practices: In the future, disk drives will be used for archiving or for non-performance oriented applications. Evaluate the use of data stores that can reside in memory, caches, or on SSDs.

4. Future Directions and Roadmap

In the Big Data Technology Roadmap, the results of the gap analysis should be augmented with a list of future developments that will help close the gaps. Ideally some timelines should be included to aid in project planning. This sections lists ongoing improvements mapped to elements of Reference Architecture with links for more detail.
1. Processing Performance Improvements
(Reference Architecture: Data Processing)

Data in memory or stored on Solid State Drive (SSD)
http://www3.weforum.org/docs/GITR/2012/GITR_Chapter1.7_2012.pdf
http://www.datanami.com/datانami/2012-02-13/big_data_and_the_ssd_mystique.htm

Enhancements to first generation Map-Reduce
http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html
http://incubator.apache.org/mesos/

Use of GPUs

2. Application Development Improvements
(Reference Architecture: Development Tools)

Big Data PaaS and data grids
http://searchsoa.techtarget.com/feature/Look-out-Big-Data-In-memory-data-grids-start-to-go-mainstream
http://aws.amazon.com/elasticmapreduce/

Visual design, development, and deploy tools
http://www.pentahobigdata.com/overview

Unified interfaces using data virtualization

3. Complex Analytics Improvements
(Reference Architecture: Analytics)

Embedded analytics
http://www.slideshare.net/InsideAnalysis/embedded-analytics-the-next-megawave-of-innovation

Stream analytics, filtering, and complex event processing
http://www.sqlstream.com/

Integrated data ingestion, processing, storage, and analytics

4. Interoperability Improvements
(Reference Architecture: integration across components)

Data sharing among multiple Hadoop tools and external tools (e.g. using HCatalog)
http://hortonworks.com/hdp/hdp-hcatalog-metadata-services/

Queries across Hadoop and legacy databases (e.g. EDW)
http://hadoop.com/product/

Data exchanges and ETL among diverse data stores
http://sqoop.apache.org
5. Possible Alternative Deployment Improvements
(Reference Architecture: Infrastructure)

Cloud
http://www.cloudstandardscustomercouncil.org/031813/agenda.htm

HPC clusters

Appliances

6. Applications
(Reference Architecture: Applications)

Internet of Things

Big Data for Vertical Applications (e.g. science, healthcare)
http://jameskaskade.com/?p=2708

Big Data Society Applications and Issues

7. Interface Improvements
(Reference Architecture: Interfaces)

SQL interfaces to NoSQL databases
http://qconsf.com/sf2012/dl/qcon-sanfran-2012/slides/MaryHolstege_and_StephenBuxton_TheDesignOfASQLInterfaceForANoSQLDatabase.pdf

Performance optimizations for querying (e.g. columnar storage)
http://searchdatamanagement.techtarget.com/definition/columnar-database

Querying and analytics interfaces for end-user
http://www.tableausoftware.com/

5. Security and Privacy


1. Secure computations in distributed programming frameworks
2. Security best practices for non-relational data stores
3. Secure data storage and transactions logs
4. End-point input validation/filtering
5. Real-time security monitoring
6. Scalable and composable privacy-preserving data mining and analytics
7. Cryptographically enforced data centric security
8. Granular access control
9. Granular audits
10. Data provenance
6. Conclusions and General Advice

From Demystifying Big Data by TechAmerica

7. References

Demystifying Big Data by TechAmerica

Consumer Guide to Big Data from ODCA
Appendix A. Terminology Glossary

The description and links for terms are listed to help in understanding other sections.

**ACiD** - Atomicity, Consistency, Isolation, Durability are a group of properties that together guarantee that database transactions are processed reliably. 
http://en.wikipedia.org/wiki/ACID

**Analytics** - “The discovery and communication of meaningful patterns in data” 
http://en.wikipedia.org/wiki/Analytics

**Avatarnode** - Fault-tolerant extension to Namenode 

**BASE** - Basically Available, Soft state, Eventual consistency semantics 
http://en.wikipedia.org/wiki/Eventual_consistency

**Big Data** - “A collection of data set so large and complex that it is difficult to process using on-hand database management tools or traditional data processing applications.” 
http://en.wikipedia.org/wiki/Big_data

**BSON** - Binary coding of JSON 
http://bsonspec.org/

**BSP (Bulk Synchronous Parallel)** - A programming model for distributed computation that avoid writing intermediate results to files 
http://en.wikipedia.org/wiki/Bulk_synchronous_parallel

**Business Analytics** - “Refers to the skills, technologies, applications and practices for continuous iterative exploration and investigation of past business performance to gain insight and drive business planning” 
http://en.wikipedia.org/wiki/Business_analytics

**Cache** - Intermediate storage between files and memory used to improve performance 
http://en.wikipedia.org/wiki/Database_caching

**(CEP) Complex Event Processing** - “Event processing that combines data from multiple sources[2] to infer events or patterns that suggest more complicated circumstances”. 
http://en.wikipedia.org/wiki/Complex_event_processing

**Consistent Hashing** - A hashing algorithm that is resilient to dynamic changes 
http://en.wikipedia.org/wiki/Consistent_hashing

**Descriptive Analytics** - "The discipline of quantitatively describing the main features of a collection of data.”
Discovery Analytics - Data mining and related processes

ELT (Extract, Load, Transform) - “A process architecture where a bulk of the transformation work occurs after the data has been loaded into the target database in its raw format”
http://it.toolbox.com/wiki/index.php/ELT

ETL (Extract, Transform Load) - Extracting data from source databases, transforming it, and then loading it into target databases.
http://en.wikipedia.org/wiki/Extract,_transform,_load

In Memory Database - A database that primarily resides in computer main memory.
http://en.wikipedia.org/wiki/In-memory_database

JSON (Javascript Object Notation) - Hierarchical serialization format derived from Javascript.
http://www.json.org/

MapReduce - A programming model for processing large data sets. It consists of a mapping processing to distributed resources, followed by a sorting phase of intermediate results, and parallel reduction to a final result.
http://en.wikipedia.org/wiki/MapReduce

MPP (Massively Parallel Processing) - “Refers to the use of a large number of processors to perform a set of coordinated computations in parallel”
http://en.wikipedia.org/wiki/Massive_parallel_processing

NewSQL - Big Data databases supporting relational model and SQL
http://en.wikipedia.org/wiki/NewSQL

NoSQL - Big Data databases not supporting relational model
https://en.wikipedia.org/wiki/NoSQL

OLAP (Online Analytic Processing) - “OLAP tools enable users to analyze multidimensional data interactively from multiple perspective”
http://en.wikipedia.org/wiki/Online_analytical_processing

OLTP (Online Transactional Processing) - “A class of information systems that facilitate and manage transaction-oriented applications”
http://en.wikipedia.org/wiki/Online_transactional_processing

Paxos - A distributed coordination protocol
http://en.wikipedia.org/wiki/Paxos_%28computer_science%29
**Predictive Analytics** - “Encompasses a variety of techniques that analyze facts to make predictions about future, or otherwise unknown, events”
http://en.wikipedia.org/wiki/Predictive_analytics

**Prescriptive Analytics** - “Automatically synthesizes big data, multiple disciplines of mathematical sciences and computational sciences, and business rules, to make predictions and then suggests decision options to take advantage of the predictions”
http://en.wikipedia.org/wiki/Prescriptive_Analytics

**Semi-Structured Data** - Unstructured data combine with structured data (e.g. metadata)

**SSD (Solid State Drive)** - “A data storage device using integrated circuit assemblies as memory to store data persistently”

**Stream Processing** - “Given a set of data (a stream), a series of operations (kernel functions) is applied to each element in the stream”
http://en.wikipedia.org/wiki/Stream_processing

**Structured Data** - Schema can be in part of data store or within application

**Unstructured Data** - Data stored with no schema and at most Implicit structure.

**Vector Clocks** - An algorithm that generates partial ordering of events in distributed systems
http://en.wikipedia.org/wiki/Vector_clock

**Web Analytics** - “The measurement, collection, analysis and reporting of Internet data for purposes of understanding and optimizing web usage.”
http://en.wikipedia.org/wiki/Web_analytics
Appendix B. Solutions Glossary

Descriptions and links are listed here to provide references for technology capabilities.

**Accumulo** - (Database, NoSQL, Key-Value) from Apache  

**Acunu Analytics** - (Analytics Tool) on top of Aster Data Platform based on Cassandra  

**Aerospike** - (Database NoSQL Key-Value)  

**Alteryx** - (Analytics Tool)  

**Ambari** - (Hadoop Cluster Management) from Apache  

**Analytica** - (Analytics Tool) from Lumina  

**ArangoDB** - (Database, NoSQL, Multi-model) Open source from Europe  
[http://www.arangodb.org/2012/03/07/avocadodbs-design-objectives](http://www.arangodb.org/2012/03/07/avocadodbs-design-objectives)

**Aster** - (Analytics) Combines SQL and Hadoop on top of Aster MPP Database  

**Avro** - (Data Serialization) from Apache  

**Azkaban** - (Process Scheduler) for Hadoop  

**Azure Table Storage** - (Database, NoSQL, Columnar) from Microsoft  

**Berkeley DB** - (Database)  

**BigData Appliance** - (Integrated Hardware and Software Architecture) from Oracle includes Cloudera, Oracle NoSQL ,Oracle R and Sun Servers  

**BigML** - (Analytics tool) for business end-users  
[https://bigml.com/](https://bigml.com/)
**BigQuery**  - (Query Tool) on top of Google Storage
[https://cloud.google.com/products/big-query](https://cloud.google.com/products/big-query)

**BigSheets**  - (BI Tool) from IBM

**BigTable**  - (Database, NOSQL, Column oriented) from Google
[http://en.wikipedia.org/wiki/BigTable](http://en.wikipedia.org/wiki/BigTable)

**Caffeine**  - (Search Engine) from Google use BigTable Indexing

**Cascading**  - (Processing) SQL on top of Hadoop from Apache

**Cascalog**  - (Query Tool) on top of Hadoop

**Cassandra**  - (Database, NoSQL, Column oriented)

**Chukwa**  - (Monitoring Hadoop Clusters) from Apache

**Clojure**  - (Lisp-based Programming Language) compiles to JVM byte code

**Cloudant**  - (Distributed Database as a Service)
[https://cloudant.com/](https://cloudant.com/)

**Cloudera**  - (Hadoop Distribution) including real-time queries
[http://www.cloudera.com/content/cloudera/en/home.html](http://www.cloudera.com/content/cloudera/en/home.html)

**Clustrix**  - (NewSQL DB) runs on AWS

**Coherence**  - (Data Grid/Caching) from Oracle

**Colossus**  - (File System) Next Generation Google File System

**Continuity**  - (Data fabric layer) Interfaces to Hadoop Processing and data stores
Corona - (Hadoop Processing tool) used internally by Facebook and now open sourced

Couchbase - (Database, NoSQL, Document) with CouchDB and Membase capabilities
http://www.couchbase.com/

CouchDB - (Database, NoSQL, Document)
http://couchdb.apache.org/

Data Tamer - (Data integration and curation tools) from MIT

Datameer - (Analytics) built on top of Hadoop
http://www.datameer.com/

Datastax - (Integration) Built on Cassandra, Solr, Hadoop
http://www.datastax.com/

Dremel - (Query Tool) interactive for columnar DBs from Google
http://research.google.com/pubs/pub36632.html

Drill - (Query Tool) interactive for columnar DBs from Apache
http://en.wikipedia.org/wiki/Apache_Drill

Dynamo DB - (Database, NoSQL, Key-Value)
http://aws.amazon.com/dynamodb/

Elastic MapReduce - (Cloud-based MapReduce) from Amazon
http://aws.amazon.com/elasticmapreduce/

ElasticSearch - (Search Engine) on top of Apache Lucerne
http://www.elasticsearch.org/

Enterprise Control Language (ECL) - (Data Processing Language) from HPPC
http://hpccsystems.com/download/docs/ecl-language-reference

Erasure Codes - (Alternate to file replication for availability) Replicates fragments.

eXtreme Scale - (Data Grid/Caching) from IBM

F1 - (Combines relational and Hadoop processing) from Google built on Google Spanner
http://research.google.com/pubs/pub38125.html
**Falcon** - (Data processing and management platform) from Apache

**Flume** - (Data Collection, Aggregation, Movement)

**FlumeJava** - (Java Library) Supports development and running data parallel pipelines

**Fusion-io** - (SSD Storage Platform) can be used with HBase

**GemFire** - (Data Grid/Caching) from VMware

**Gensonix** - (NoSQL database) from Scientel
[http://scientel.com/platform.html](http://scientel.com/platform.html)

**Gephi** - (Visualization Tool) for Graphs
[https://gephi.org/features/](https://gephi.org/features/)

**Gigaspaces** - (Data Grid/Caching)

**Giraph** - (Graph Processing) from Apache

**Google Refine** - (Data Cleansing)
[http://code.google.com/p/google-refine/](http://code.google.com/p/google-refine/)

**Google Storage** - (Database, NoSQL, Key-Value)
[https://developers.google.com/storage/](https://developers.google.com/storage/)

**Graphbase** - (Database, NoSQL, Graphical)
[http://graphbase.net/](http://graphbase.net/)

**Greenplum** - (MPP Database. Analytic Tools, Hadoop)

**HBase** - (Database, NoSQL, Column oriented)

**Hadapt** - (Combined SQL Layer and Hadoop)
Hadoop Distributed File System - (Distributed File System) from Apache

Hama - (Processing Framework) Uses BSP model on top of HDFS
http://hama.apache.org/

Hana - (Database, NewSQL) from SAP
http://en.wikipedia.org/wiki/SAP_HANA

Haven - (Analytics Package) from HP

HAWQ - (SQL Interface to Hadoop) from Greenplum and Pivotal
http://www.greenplum.com/blog/dive-in/hawq-the-new-benchmark-for-sql-on-hadoop

HCatalog - (Table and Storage Management) for Hadoop data
http://incubator.apache.org/hcatalog/

HDF5 - (A data model, library, and file format for storing/managing large complex data)
http://www.hdfgroup.org/HDF5/

High Performance Computing Cluster (HPCC) - (Big Data Processing Platform)
http://hpccsystems.com/why-hpcc

Hive - (Data warehouse structure on top of Hadoop)
http://en.wikipedia.org/wiki/Apache_Hive

HiveQL - (SQL-like interface on Hadoop File System)
https://www.inkling.com/read/hadoop-definitive-guide-tom-white-3rd/chapter-12/hiveql

Hortonworks - (Extensions of Hadoop)
http://hortonworks.com/

HStreaming - (Real time analytics on top of Hadoop)
http://www.hstreaming.com/

Hue - (Open source UI for Hadoop) from Cloudera
http://cloudera.github.io/hue/

Hypertable - (Database, NoSQL, Key-Value) open source runs on multiple file systems
http://hypertable.org/

Impala - (Ad hoc query capability for Hadoop) from Cloudera
http://blog.cloudera.com/blog/2012/10/cloudera-impala-real-time-queries-in-apache-hadoop-for-real/
**InfiniDB** - (Scale-up analytic database)
http://infinidb.org/

**Infochimps** - (Big Data Storage and Analytics in the Cloud)
http://www.infochimps.com/

**Infosphere Big Insights** - (Analytic) from IBM
http://www-01.ibm.com/software/data/infosphere/biginsights/

**InnoDB** - (Default storage engine for MYSQL)
http://en.wikipedia.org/wiki/InnoDB

**Jaql** = (Query Language for Hadoop) from IBM
http://www-01.ibm.com/software/data/infosphere/hadoop/jaql/

**Kafka** - (Publish-and-subscribe for data) from Apache
http://kafka.apache.org/

**Karmasphere** - (Analytics)
http://www.karmasphere.com/

**Knox** - (Secure gateway to Hadoop) from Apache
http://knox.incubator.apache.org/

**Lucidworks** - (Search built on Solr/Lucene) and an associated Big Data Platform
http://www.lucidworks.com/

**Knowledge Graph** - (Graphical data store) from Google
http://en.wikipedia.org/wiki/Knowledge_Graph

**Mahout** - (Machine Learning Toolkit) built on Apache Hadoop
http://en.wikipedia.org/wiki/Knowledge_Graph

**MapD** - (Massive Parallel Database) Open Source on top of GPUs

**MapReduce** - (Processing algorithm)
http://en.wikipedia.org/wiki/MapReduce

**MapR** - (MapReduce extensions) built on NFS
http://en.wikipedia.org/wiki/Knowledge_Graph

**MarkLogic** - (Database, NoSQL, Document) interfaced with Hadoop
http://www.marklogic.com/
**Memcached** - (Data Grid/Caching)
http://en.wikipedia.org/wiki/Memcached

**MemSQL** - (In memory analytics database)
http://www.memsql.com/

**MongoDB** - (Database, NoSQL, Document) from 10gen
http://www.mongodb.org/

**mrjob** - (Workflow) for Hadoop from Yelp
http://bighadoop.wordpress.com/2012/04/15/yelps-mrjob-a-python-package-for-hadoop-jobs/

**MRQL** - (Query Language) supports Map-Reduce and BSP processing
http://code.google.com/p/mrql/

**Muppet** - (Stream Processing) MapUpdate implementation

**MySql** - (Database Relational)
http://www.mysql.com/

**Namenode** - Directory service for Hadoop
http://wiki.apache.org/hadoop/NameNode

**Neo4j** - (Database, NoSQL, Graphical)
http://www.neo4j.org/

**Netezza** - (Database Appliance)
http://www-01.ibm.com/software/data/netezza/

**NuoDB** - (MPP Database)
http://www.nuodb.com/

**Oozie** - (Workflow Scheduler for Hadoop) from Apache
http://oozie.apache.org/

**Oracle NoSQL** - (Database, Key-Value)

**ORC (Optimized Row Columnar) Files** - File Format for Hive data in HDFS
http://docs.hortonworks.com/HDPDocuments/HDP2/HDP-2.0.0.2/ds_Hive/orcfile.html

**Parquet** - (Columnar file format for Hadoop) from Cloudera
http://blog.cloudera.com/blog/2013/03/introducing-parquet-columnar-storage-for-apache-hadoop/
Pentaho  - (Analytic tools)  
http://www.pentaho.com/  

Percolater  - (Data Processing) from Google  
http://research.google.com/pubs/pub36726.html  

Pig  - (Procedural framework on top of Hadoop)  
http://pig.apache.org/  

Pig Latin  - (Interface language for Pig procedures)  
http://pig.apache.org/docs/r0.7.0/piglatin_ref1.html  

Pivotal  - (New company utilizing VMware and EMC technologies)  
http://www.gopivotal.com/  

Platfora  - (In memory caching for BI on top of Hadoop)  
http://www.platfora.com/  

Postgres  - (Database Relational)  
http://www.postgresql.org/  

Precog  - (Analytics Tool) for JSON data  
http://precog.com/  

Pregel  - (Graph Processing) used by Google  

Presto  - (SQL Query for HDFS) from Facebook  

Protocol Buffers  - (Serialization) from Google  

Protovis  - (Visualization)  
http://mbostock.github.io/protovis/  

PureData  - (Database Products) from IBM  
http://www-01.ibm.com/software/data/puredata/  

R  - (Data Analysis Tool)  
http://en.wikipedia.org/wiki/R_%28programming_language%29  

Rainstor  - (Combines Hadoop and Relational Processing)  
http://rainstor.com/
**RCFile** - (Record Columnar File) - File format optimized for HDFS data warehouses

**Redis** - (Database, NoSQL, Key-Value)
http://redis.io/

**Redshift** - (Database Relational) Amazon
http://aws.amazon.com/redshift/

**Resilient Distributed Datasets** - (Fault tolerant in memory data sharing)

**Riak** - (Database, NoSQL, Key-Value with built-in MapReduce) from Basho
http://basho.com/riak/

**Roxie** - (Query processing cluster) from HPCC
http://hpccsystems.com/FAQ/what-roxie

**RushAnalytics** - (Analytics) from Pervasive
http://bigdata.pervasive.com/Products/Big-Data-Analytics-RushAnalytics.aspx

**S3** - (Database, NoSQL, Key-Value)
http://en.wikipedia.org/wiki/R_%28programming_language%29

**S4** - (Stream Processing)
http://incubator.apache.org/s4/

**Sawzall** - (Query Language for Map-Reduce) from Google

**Scala** - (Programming Language) Combines functional and imperative programming
http://www.scala-lang.org/

**Scalebase** - (Scalable Front-end to distributed Relational Databases)
http://www.scalebase.com/

**SciDB** - (Database, NoSQL, Arrays)
http://www.scidb.org/

**scikit learn** - (Machine Learning Toolkit) in Python
http://scikit-learn.org/stable/

**Scribe** - (Server for Aggregating Log Data) originally from Facebook may be inactive
SequenceFiles - (File format) Binary key-value pairs
http://wiki.apache.org/hadoop/SequenceFile

Shark - (Complex Analytics Platform) on top of Spark
https://amplab.cs.berkeley.edu/projects/shark-making-apache-hive-run-at-interactive-speeds/

Simba - (ODBC SQL Driver for Hive)

SimpleDB - (Database, NoSQL, Document) from Amazon
http://aws.amazon.com/simpledb/

Skytree - (Analytics Server)
http://www.skytree.net/

Solr/Lucene - (Search Engine) from Apache
http://lucene.apache.org/solr/

Spotfire - (Stream processing tool) from TIBCO
http://spotfire.tibco.com/

Spanner - (Database, NewSQL) from Google

Spark - (In memory cluster computing system)
http://spark-project.org/

Splunk - (Machine Data Analytics)
http://www.splunk.com/

Spring Data - (Data access tool for Hadoop and NoSQL) in Spring Framework
http://www.springsource.org/spring-data

SQLite - (Software library supporting server-less relational database)
http://www.sqlite.org/

SQLstream - (Streaming data analysis products)
http://www.sqlstream.com/

Sqoop - (Data movement) from Apache
http://en.wikipedia.org/wiki/Sqoop

Sqrrl - (Security and Analytics on top of Apache Accumulo)
http://www.sqrrl.com/
Stinger - (Optimized Hive Queries) from Hortonworks  
http://hortonworks.com/blog/100x-faster-hive/

Storm - (Stream Processing)  

Sumo Logic - (Log Analytics)  
http://www.sumologic.com/

Tableau - (Visualization)  
http://www.tableausoftware.com/

Tachyon - (File system) from Berkeley  

Talend - (Data Integration Product)  
http://www.talend.com

TempoDB - (Database, NoSQL, Time Series)  
https://tempo-db.com/

Teradata Active EDW - (Database, Relational)  
http://www.teradatas.com/Active-Enterprise-Data-Warehouse/

Terracotta - (In memory data management)  
http://terracotta.org/

Terraswarm - (Data Acquisition) Sensor Integration  
http://www.terraswarm.org/

Thor - (Filesystem and Processing Cluster) from HPCC Systems  
http://hpccsystems.com/FAQ/what-thor

Thrift - (Framework for cross-language development)  
http://thrift.apache.org/

Tinkerpop - (Graph Database and Toolkit)  
http://thrift.apache.org/

Vertica - (Database Relational)  
http://www.vertica.com/

Voldemort - (Database, NoSQL, Key-Value)  
http://www.project-voldemort.com/voldemort/
VoltDB - (Database NewSQL)
http://voltdb.com/

Watson from IBM - (Analytic Framework)
http://www-03.ibm.com/innovation/us/watson/

WebHDFS - (REST API for Hadoop)
http://hadoop.apache.org/docs/r1.0.4/webhdfs.html

WEKA - (Machine Learning Toolkit) in Java
http://en.wikipedia.org/wiki/Weka_%28machine_learning%29

Wibidata - (Components for building Big Data applications)
http://www.wibidata.com/

YarcData - (Graph Analytics for Big Data)
http://www.yarcdata.com/

Yarn - (Next Generation Hadoop) from Apache
http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html

Yottamine - (Machine Learning Toolkit) Cloud-based
http://yottamine.com/

Zettaset Orchestrator - (Management and Security for Hadoop)
http://www.zettaset.com/platform.php

ZooKeeper - (Distributed Computing Management)
http://zookeeper.apache.org/
Appendix C. Application Use Case Examples

From http://thebigdatainstitute.wordpress.com/2013/05/23/our-favorite-40-big-data-use-cases-whats-your/

“While there are extensive industry-specific use cases, here are some for handy reference:

**EDW Use Cases**

- Augment EDW by offloading processing and storage
- Support as preprocessing hub before getting to EDW

**Retail/Consumer Use Cases**

- Merchandizing and [market basket analysis](#)
- [Campaign management](#) and customer [loyalty programs](#)
- [Supply-chain management](#) and analytics
- Event- and behavior-based targeting
- Market and consumer segmentations

**Financial Services Use Cases**

- Compliance and regulatory reporting
- Risk analysis and management
- [Fraud detection](#) and security analytics
- CRM and customer loyalty programs
- Credit risk, scoring and analysis
- High speed arbitrage trading
- Trade surveillance
- Abnormal trading pattern analysis

**Web & Digital Media Services Use Cases**

- Large-scale clickstream analytics
- Ad targeting, analysis, forecasting and optimization
- Abuse and click-fraud prevention
- Social graph analysis and profile segmentation
- Campaign management and loyalty programs

**Health & Life Sciences Use Cases**

- Clinical trials data analysis
- [Disease pattern](#) analysis
- Campaign and sales program optimization
- Patient care quality and program analysis
- Medical device and pharma supply-chain management
- Drug discovery and development analysis
Telecommunications Use Cases

- Revenue assurance and price optimization
- Customer churn prevention
- Campaign management and customer loyalty
- Call detail record (CDR) analysis
- Network performance and optimization
- Mobile user location analysis

Government Use Cases

- Fraud detection
  - Threat detection
- Cybersecurity
- Compliance and regulatory analysis
- http://www.whitehouse.gov/sites/default/files/microsites/ostp/big_data_press_release_final_2.pdf

New Application Use Cases

- Online dating
- Social gaming

Fraud Use-Cases

- Credit and debit payment card fraud
  - Deposit account fraud
- Technical fraud and bad debt
  - Healthcare fraud
- Medicaid and Medicare fraud
- Property and casualty (P&C) insurance fraud
- Workers’ compensation fraud

E-Commerce and Customer Service Use-Cases

- Cross-channel analytics
- Event analytics
- Recommendation engines using predictive analytics
- Right offer at the right time
- Next best offer or next best action”

http://www.theequitykicker.com/2012/03/12/looking-to-the-use-cases-of-big-data/discusses some Big Data Use Case examples.
Appendix D. Actors and Roles

From http://www.smartplanet.com/blog/bulletin/7-new-types-of-jobs-created-by-big-data/682

The job roles are mapped to elements of the Reference Architecture in red

“Here are 7 new types of jobs being created by Big Data:

1. **Data scientists:** This emerging role is taking the lead in processing raw data and determining what types of analysis would deliver the best results. Typical backgrounds, as cited by Harbert, include math and statistics, as well as artificial intelligence and natural language processing. (Analytics)
2. **Data architects**: Organizations managing Big Data need professionals who will be able to build a data model, and plan out a roadmap of how and when various data sources and analytical tools will come online, and how they will all fit together. (Design, Develop, Deploy Tools)

3. **Data visualizers**: These days, a lot of decision-makers rely on information that is presented to them in a highly visual format — either on dashboards with colorful alerts and “dials,” or in quick-to-understand charts and graphs. Organizations need professionals who can “harness the data and put it in context, in layman’s language, exploring what the data means and how it will impact the company,” says Harbert. (Applications)

4. **Data change agents**: Every forward-thinking organization needs “change agents” — usually an informal role — who can evangelize and marshal the necessary resources for new innovation and ways of doing business. Harbert predicts that “data change agents” may be more of a formal job title in the years to come, driving “changes in internal operations and processes based on data analytics.” They need to be good communicators, and a Six Sigma background — meaning they know how to apply statistics to improve quality on a continuous basis — also helps. (Not applicable to Reference Architecture)

5. **Data engineer/operators**: These are the people that make the Big Data infrastructure hum on a day-to-day basis. “They develop the architecture that helps analyze and supply data in the way the business needs, and make sure systems are performing smoothly,” says Harbert. (Data Processing and Data Stores)

6. **Data stewards**: Not mentioned in Harbert’s list, but essential to any analytics-driven organization, is the emerging role of data steward. Every bit and byte of data across the enterprise should be owned by someone — ideally, a line of business. Data stewards ensure that data sources are properly accounted for, and may also maintain a centralized repository as part of a Master Data Management approach, in which there is one “gold copy” of enterprise data to be referenced. (Data Governance)

7. **Data virtualization/cloud specialists**: Databases themselves are no longer as unique as they used to be. What matters now is the ability to build and maintain a virtualized data service layer that can draw data from any source and make it available across organizations in a consistent, easy-to-access manner. Sometimes, this is called “Database-as-a-Service.” No matter what it’s called, organizations need professionals that can also build and support these virtualized layers or clouds.” (Infrastructure)