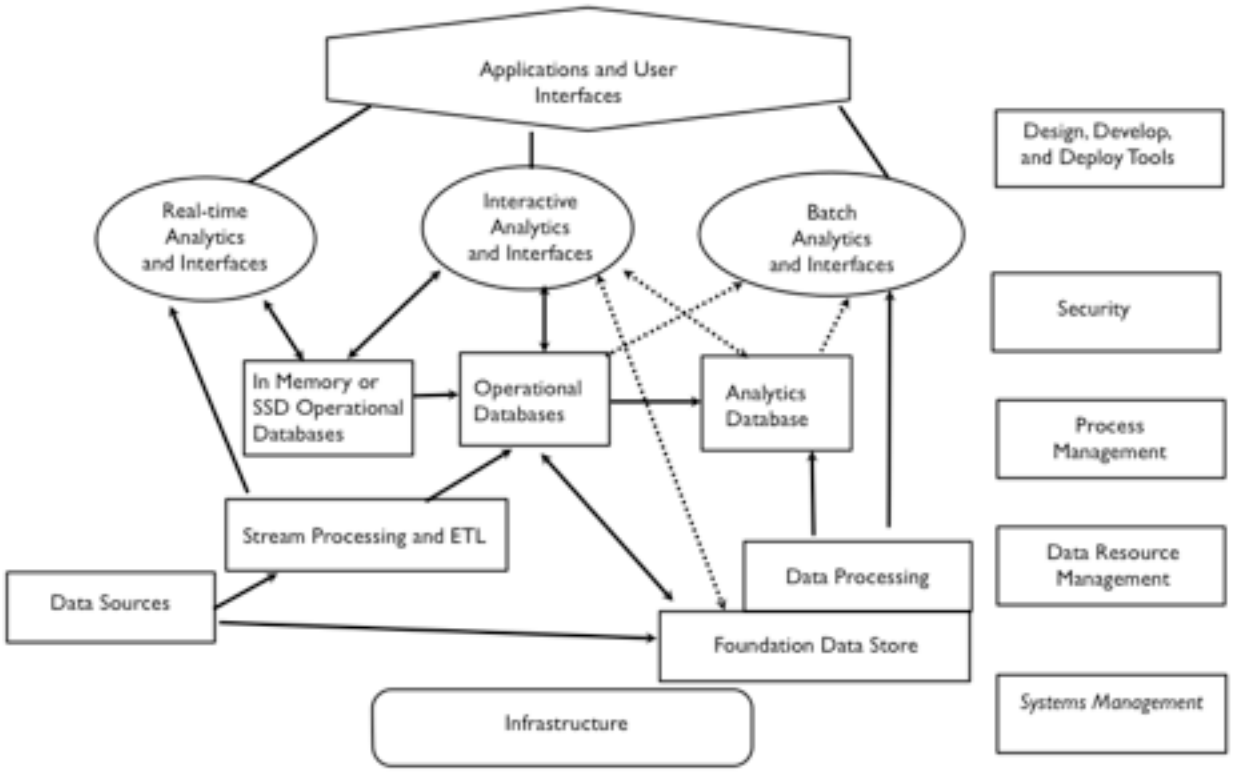
Reference Architecture, Requirements, Gaps, Roles

The contents of this document are an excerpt from the brainstorming document M0014 . The purpose is to show how a detailed Big Data Reference Architecture can be used to link requirements, gap analysis, and roles. A less detailed Reference Architecture (e.g. merging all data stores into a single data storage element) could be used as a high-level view of the detailed Reference Architecture. However a high-level view will not be able to differentiate Big Data specific components or map into detailed requirements and gap analysis. As a brainstorming first cut, the Reference Architecture can be refined and modified. However the level of detail and the ability to map to subgroup deliverables should probably be preserved.

1. Reference Architecture

The Reference Architecture below will help focus the discussion of other deliverables.



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:

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.

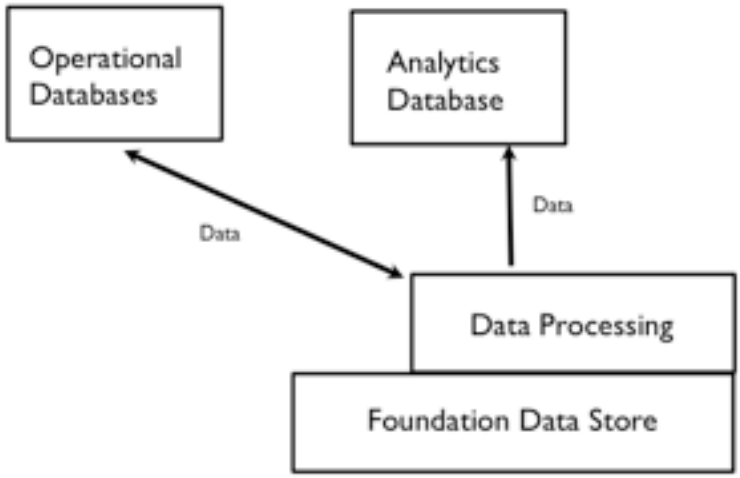
Possible Requirements: The data integration middleware should be able to do high performance extraction, transformation and load operations for diverse data models and formats.

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.

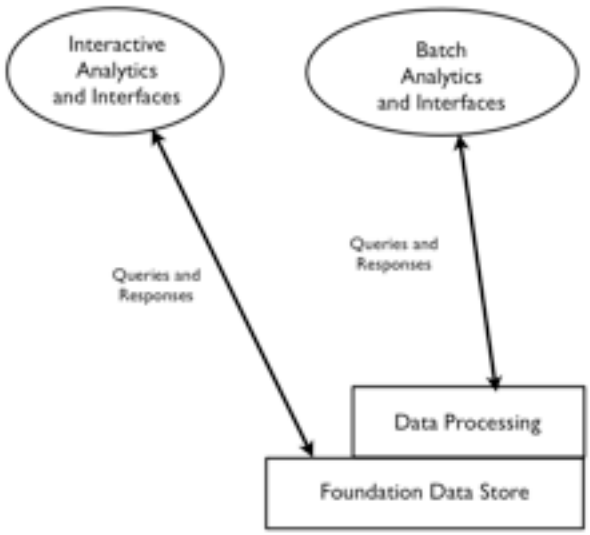
Possible Requirements: For good through-put performance on very large data sets, the data processing system will require multi-stage parallel processing

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:



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.

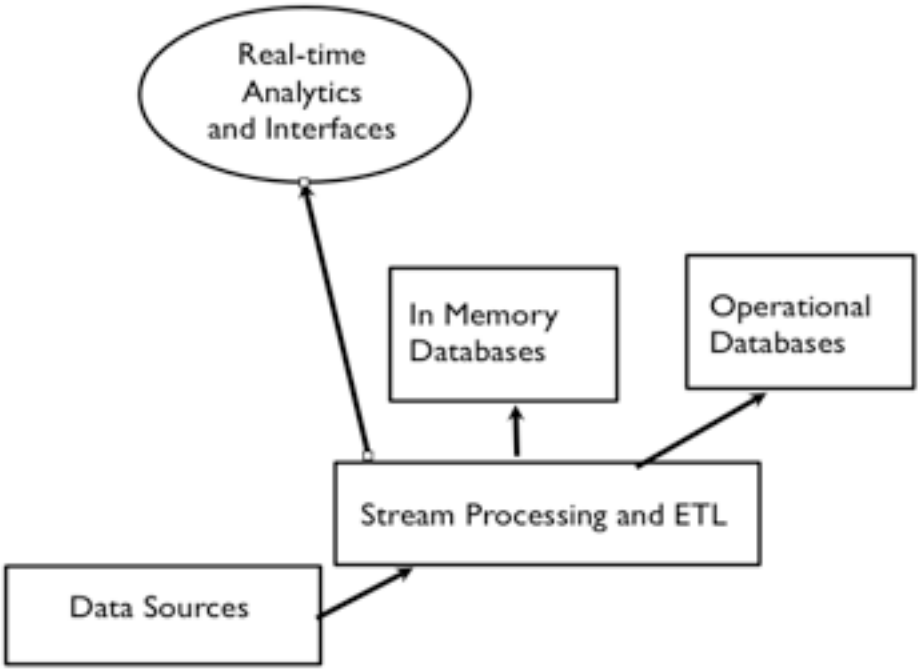
Possible Requirements: High performance data processing is needed for efficient analytics.

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:



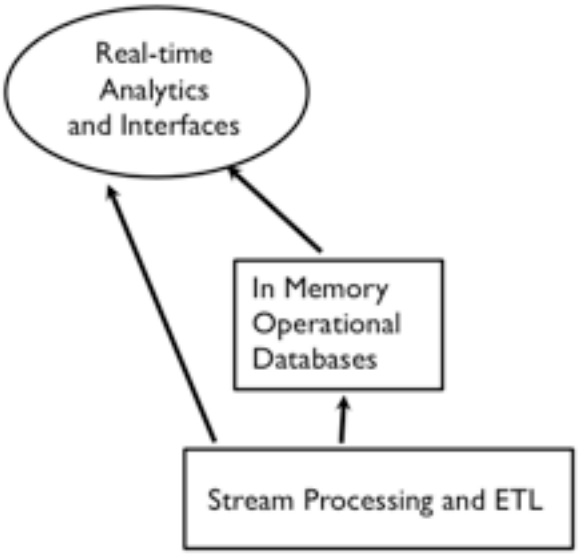
Description: Stream processing software can transform, process, and route data to databases and real time analytics

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.

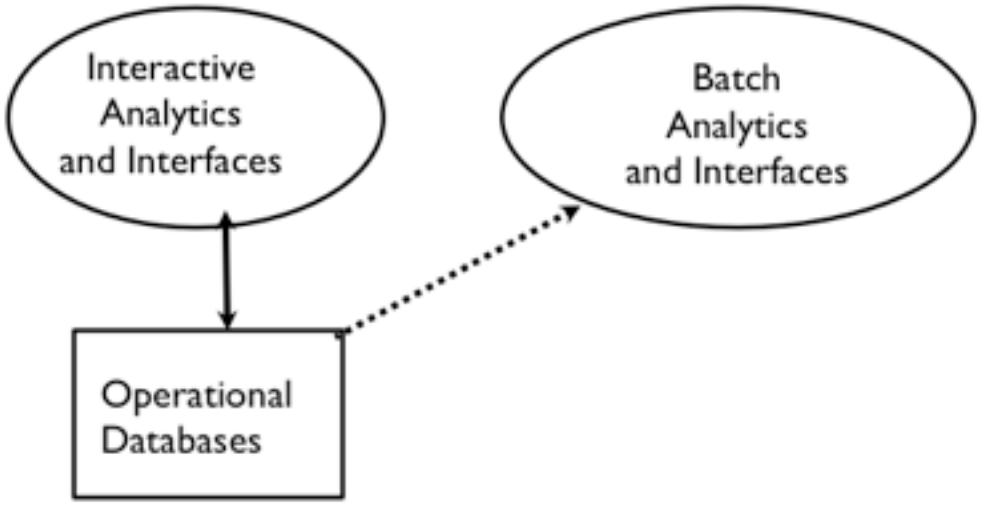
Possible Requirements: Efficient algorithms for pattern matching and/or machine learning are necessary.

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.

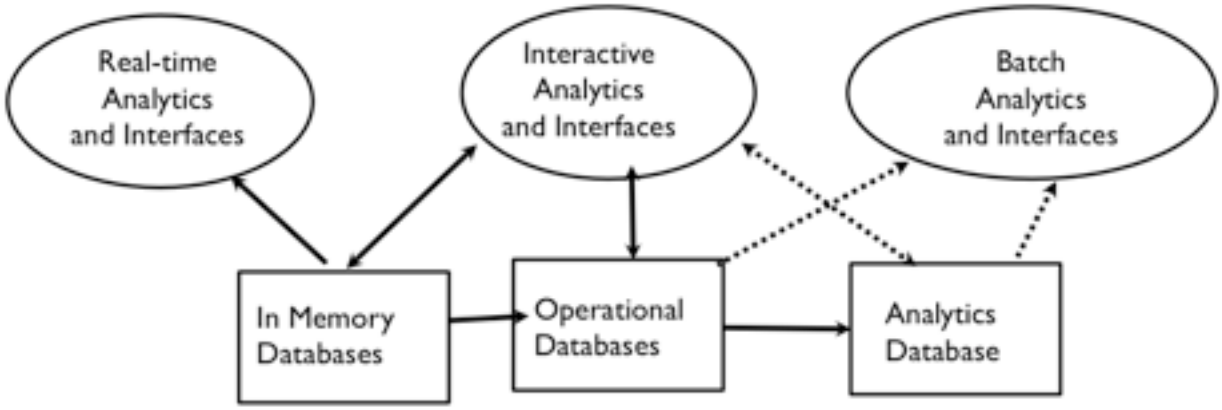
Possible Requirements: It is necessary to decide on the level of consistency vs. availability is needed since the CAP theorem demonstrates that both can not be achieved in horizontally scaled systems.

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.

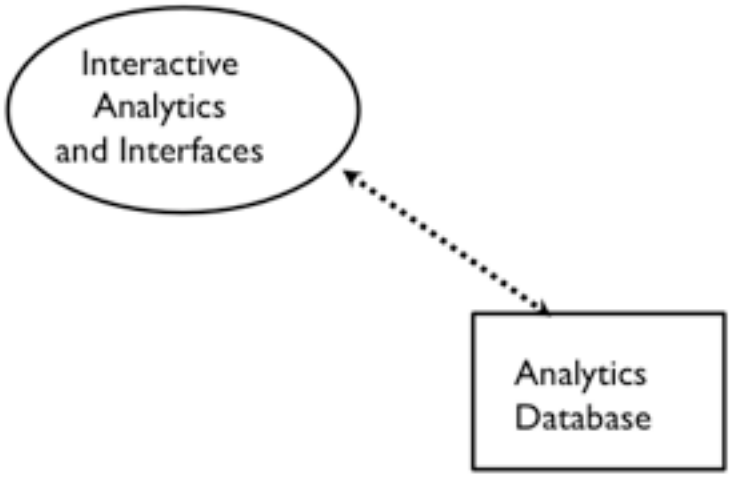
Possible Requirements: The data types to be stored depend on application data usage requirements and query patterns.

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

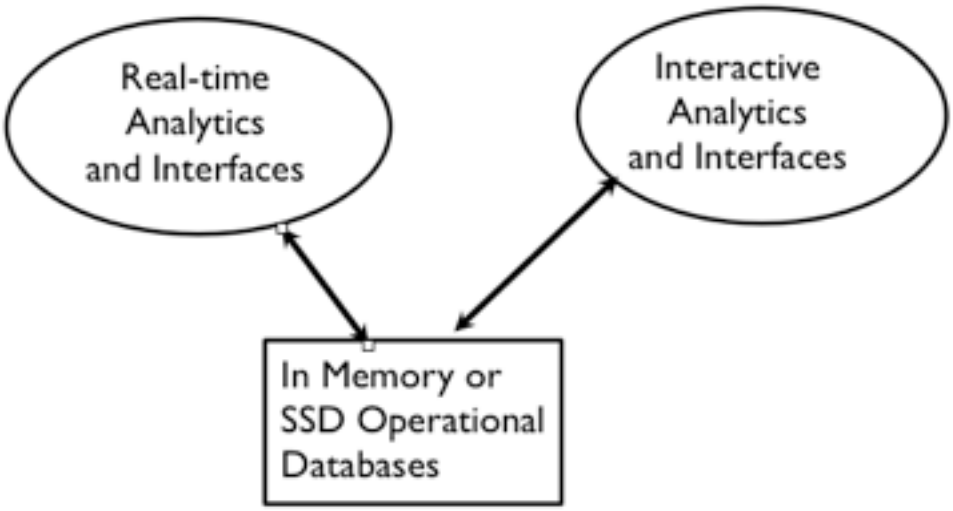
Possible Requirements: Analytic databases need high performance on complex queries which require optimizations such as columnar storage, in memory caches, and star schema data models.

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.

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 Resource Management)

7. **Data virtualization/cloud specialists:** Databases themselves are no longer as unique as they use 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)