**Big data platforms: What's next?**

**Important pros of Hadoop compared with parallel SQL systems include:**

1.Open source availability versus expensive software licenses.

2.Multiple non-monolithic layers and components versus having only a top-level query API through which to access the data.

3. Support for access to file-based external data versus having to first design, load, and then index tables before being able to proceed.

4. Support for automatic and incremental forward recovery of jobs with failed tasks versus rolling long jobs back to their beginning to start all over again on failure.

5. Automatic data placement and rebalancing as data grows and machines come and go versus manual, DBA-driven data placement.

6. Support for replication and machine fail-over without operator intervention versus pager-carrying DBAs having to guide data recovery activities.

**Some of the cons are:**

1. Similar to early observations on why database systems’ needs were not met by traditional OSs and their file systems, layering a record-based abstraction on top of a very large byte sequential file abstraction leads to an impedance mismatch.

2. There is no imaginable reason, other than “because it is already there,” to layer a high-level data language on top of a two-unary-operator runtime like MapReduce, as it can be quite unnatural (e.g., for joins) and can lead to suboptimal performance.

3. With random data block partitioning, the only available parallel query processing strategy is to “spray-and-pray” every query to all blocks of the relevant data files.

4. A flexible, semi-structured, schema-less data model (based on keys and values) means that important information about the data being operated on is known only to the programs operating on it (so program maintenance troubles await).

5. Coupling front- and back-end big data platforms to cover the full big data lifecycle requires significant use of bubble gum, baling wire, and handwritten ETL-like scripts.

6. While Hadoop definitely scales, its computational model is quite heavy (e.g., always sorting the data flowing between Map and Reduce, always persisting temporary data to HDFS between jobs in a multi-job query plan, etc.)

**Beyond Hadoop**

* Came from Google file system and MapReduce from 2004
* Comprises two parts, the Hadoop Distributed File System (HDFS), similar to a file system on a single computer, disperses large datasets among hundreds or even thousands of commodity hardware. Hadoop MapReduce, takes advantage of data being slit up on individual computers with their own processing power, the task of analysing the data is split up.
* MapReduce not designed to analyse data sets threaded with connections such data from a social network
* data placement and communication issues
* Businesses want results in real time but hadoop is too slow therefore other tools have emerged such as HBase or Cloudant for example to augment Hadoop for this purpose

**Improving Network Scalability Using NoSql Database**

* Cassandra contains Column Families or Super Column Families instead of tables in a relational database. Column Families contains row key s where each row key contains one or more columns and each column is a name/value pair.
* In relational tables if we don't have value for a particular column, we use NULL where as in Cassandra instead of nulls allows a name/value pair can be omitted should there be no value.

**Oracle Releases NoSQL Database, Advances Big Data Plans**

* Cassandra is a transactional NoSQL database, this allows it to regularly make schema changes and add new attributes and features to profiles and social network interactions.
* Cassandra is open source and designed to scale out on commodity hardware therefore it is a low cost option.