Wrapping Your Head Around Hadoop

From its inception to this day, Hadoop has focused on providing a scalable, reliable platform for storage and data analysis that runs on commodity hardware and is fault tolerant. Storage is offered by HDFS (Hadoop Distributed File System) and the processing capabilities are offered by YARN (Yet Another Resource Negotiator). In a nutshell:

Hadoop = HDFS + YARN

Everything else is part of the Hadoop ecosystem. This is a radical simplification of a traditional database design.

Unlike databases, Hadoop does not know what kind of data will be stored as files in HDFS. It does not know whether that data has certain fields in it, or whether its structure is opaque. It is only during the processing step that some kind of a structure will be imposed on those raw files. This is known as the ***schema-on-read*** approach to data management: you just dump whatever raw data comes your way into files in HDFS and you do not think about it until the processing step. The repository of unstructured data is called a ***data lake*** (a term we have already seen when talking about Single View use cases).

Compare it with the traditional database design, where data is neatly put into tables with fixed columns. This is known as ***schema-on-write***: you actually have to structure your data upfront, before you can store

it. This allows traditional relational databases to implement things like indexing and upfront optimizations of SQL queries. With Hadoop, none of it comes for free, but you do get a much higher scalability and fault tolerance. In engineering, as in life: it's always about compromises.

To put it simply: we are trading the **extreme convenience of relational databases** for the **extreme scalability of Hadoop**.