# ♥ 01: Q1 – Q6 Hadoop overview & architecture interview questions & answers

Posted on April 15, 2016 by Arulkumaran Kumaraswamipillai

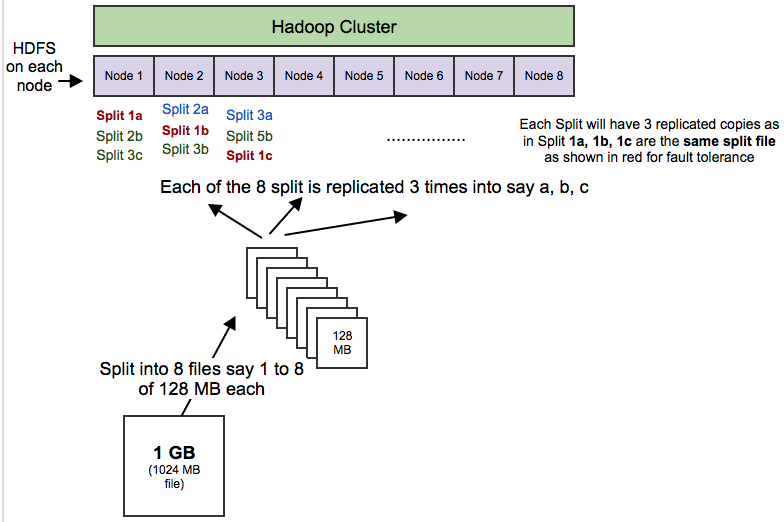


Q1. What is Hadoop?

A1. Hadoop is an open-source software framework for storing large amounts of data and processing/querying those data on a cluster with multiple nodes of **commodity hardware** (i.e. low cost hardware). In short, Hadoop consists of

**1.** HDFS (Hadoop Distributed File System): HDFS allows you to store huge amounts of data in a distributed and a redundant manner. For example, a **1 GB** (i.e 1024 MB) text file can be split into **16** * 128MB files and stored on 8 different

nodes in a Hadoop cluster. Each split can be replicated 3 times for fault tolerance so that if 1 node goes down, you have backups. HDFS is good for sequential **write-once-and-read-many times** type access.



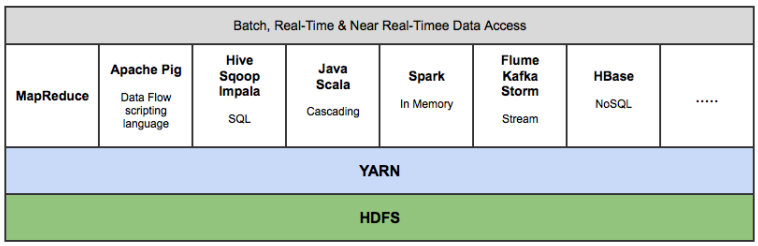

HDFS (Hadoop Distributed Files System)

**2.** MapReduce: A computational framework. This processes large amounts of data in a distributed and parallel manner. When you do a query on the above 1 GB file for all users with age > 18, there will be say "8 **map**" functions running in parallel to extract users with age > 18 within its 128MB split file, and then the "**reduce**" function will run to combine all the individual outputs into a single final result.

**3.** YARN (Yet Another Resource Nagotiator): A framework for job scheduling and cluster resource management.

**4.** Hadoop eco system, with 15+ frameworks & tools like Sqoop, Flume, Kafka, Pig, Hive, Spark, Impala, etc to ingest data into HDFS, to wrangle data (i.e. transform, enrich, aggregate, etc) within HDFS, and to query data from HDFS for business intelligence & analytics. Some tools like Pig & Hive are abstraction layers on top of MapReduce, whilst the other tools like Spark & Impala are improved architecture/design from MapReduce for much improved

latencies to support near real-time (i.e. NRT) & real-time processing.




Hadoop Overview

Q2. Why are organizations moving from traditional data warehouse tools to smarter data hubs based on Hadoop eco systems?

A2. Organizations are investing on enhancing their

## Existing data infrastructure:

- predominantly using "**structured data**" stored in high-end & expensive hardwares
- predominantly processed as ETL batch jobs for ingesting data into RDBMS and data warehouse systems for data mining, analysis & reporting to make key business decisions.
- predominantly handle data volumes in gigabytes to terabytes

To

## Smarter data infrastructure based on Hadoop where

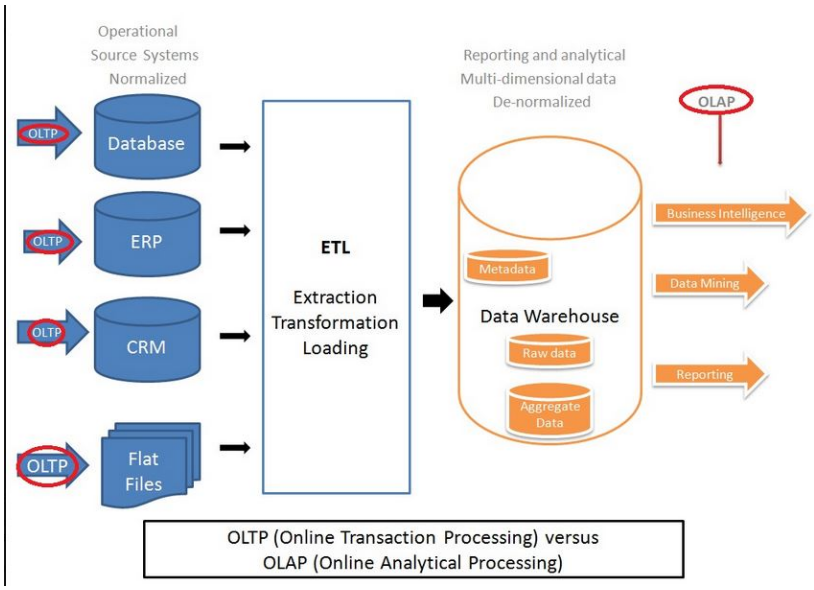
- structured (e.g. RDBMS), **unstructured** (e.g, images, PDFs, docs ), & **semi-structured** (e.g. logs, XMLs) data can be stored in cheaper commodity machines in a scalable and fault tolerant manner.
- data can be ingested via batch jobs and **near real time** (i.e. NRT, 200ms to 2 seconds) streaming (e.g. Flume & Kafka).

- data can be queried with low latency (i.e under 100ms) capabilities with tools like Spark & Impala.
- larger data volumes in terabytes to petabytes can be stored.

which empowers organizations to make better business decisions with smarter & bigger data with more powerful tools **to ingest data**, **to wrangle stored data** (e.g. aggregate, enrich, transform, etc), and **to query** the wrangled data with low-latency capabilities for reporting & business intelligence.

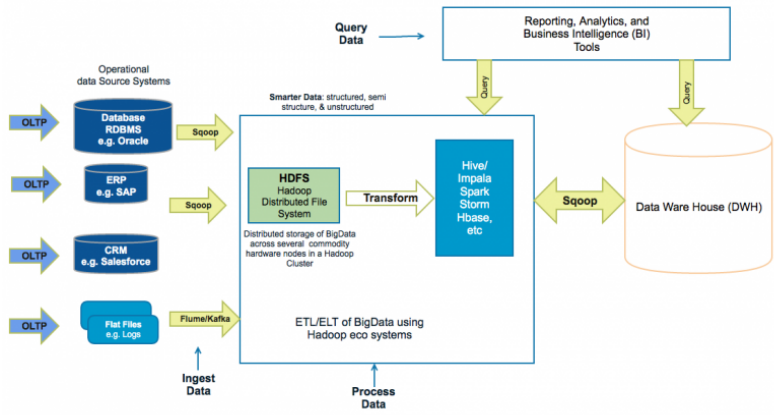
Q3. How does a smarter & bigger data hub architectures differ from a traditional data warehouse architectures?
A3.

## Traditional Enterprise Data Warehouse Architecture




OLAP

# Hadoop based Data Hub Architecture




Hadoop Architecture Overview

Q4. What are the benefits of Hadoop based Data Hubs?

A4.

**1) Improves the overall SLAs** (i.e. Service Level Agreements) as the data volume & complexity grows. E.g. “Shared Nothing” architecture, parallel processing, memory intensive processing frameworks like Spark & Impala, and resource preemption in YARN’s capacity scheduler.

**2) Scaling data warehouses can be expensive**. Adding additional **high-end hardware** capacities & **licensing** of data warehouse tools can cost significantly more. Hadoop based solutions can not only be cheaper with **commodity hardware** nodes & **open-source tools**, but also can compliment the data warehouse solution by offloading data transformations to Hadoop tools like Spark & Impala for more efficient parallel processing of BigData. This will also free up the data warehouse resources.

**3) Exploration of new avenues & leads**. Hadoop can provide an exploratory sandbox for the data scientists to discover potentially valuable data from social media, log files, emails, etc that are not normally available in data warehouses.

**4) Better flexibility**. Often business requirements change, and this requires changes to schema & reports. Hadoop based solutions are not only flexible to handle **evolving**

**schemas**, but also can handle semi-structured & unstructured data from disparate sources like social media, application log files, images, PDFs, and document files.

Q5. What are key steps in BigData solutions?
A5. Ingesting Data, Storing Data (i.e. Data Modelling), and processing data (i.e data wrangling, data transformations & querying data).

## 1) Ingesting Data:

Extracting data from various sources like

1. **RDBMs** Relational Database Management Systems like Oracle, MySQL, etc.

2. **ERP**s Enterprise Resource Planning (i.e. ERP) systems like SAP.

3. **CRM** Customer Relationships Management systems like Siebel, Salesforce, etc.

4. **Social Media feeds & log files**.

5. **Flat files, docos, and images**.

and storing them on data hub based on "Hadoop Distributed File System", which is abbreviated as **HDFS**. Data can be ingested via batch jobs (e.g. running every 15 minutes, once every night, etc), streaming near-real-time (i.e 100ms to 2 minutes) and streaming in real-time (i.e. under 100ms).

One common term used in Hadoop is "**Schema-On-Read**". This means unprocessed (aka raw) data can be loaded into HDFS with a structure applied at processing time based on the requirements of the processing application. This is different from "Schema-On-Write", which is used in RDBMs where schema need to be defined before the data can be loaded.

## 2) Storing Data:

Data can be stored on HDFS or NoSQL databases like HBase. HDFS is **optimized for sequential access** & the usage pattern of "Write-Once & Read-Many". HDFS has high read & write rates as it can parallelize I/O s to multiple drives. HBase sits on top of HDFS and stores data as key/value pairs in a columnar fashion. Columns are clubbed together as column families. HBase is suited for **random read/write access**. Before data can be stored into Hadoop, you need consider the following

**1) Data Storage Formats:** There are a number of file formats (e.g CSV, JSON, sequence, AVRO, Parquet, etc) & data compression algorithms (e.g snappy, LZO, gzip, bzip2, etc) that can be applied. Each has particular strengths. Compression algorithms like LZO and bzip2 are splittable.

**2) Data Modelling:** Despite the schema-less nature of Hadoop, schema design is an important consideration. This includes directory structures and schema of objects stored in HBase, Hive and Impala. Hadoop often serves as a data hub for the entire organization, and the data is intended to be shared. Hence, carefully structured & organized storage of your data is important.

**3) Metadata management:** Metadata related to stored data.

**4) Multitenancy:** As smarter data hubs host multiple users, groups, and applications. This often results in challemges relating to governance, standardization, and management.

## 3) Processing Data:

Hadoop's processing framework uses the HDFS. It uses the "**Shared Nothing**" architecture, which in distributed systems each node is completely independent of other nodes in the system. There are no shared resources like CPU, memory, and disk storage that can become a bottle-neck. Hadoop's processing frameworks like **Spark**, **Pig**, **Hive**, **Impala**, etc processes distinct subset of the data and there is no need to

manage access to the shared data. "**Sharing nothing**" architectures are very **1. scalable** as more nodes can be added without further contention & **2. fault tolerant** as each node is independent, and there are no single points of failure, and the system can quickly recover from a failure of an individual node.

Q6. How would you go about choosing among the different file formats for storing and processing data?
A6. One of the key design decisions is regarding file formats based on the

**1)** Usage patterns like accessing 5 columns out of 50 columns vs accessing most of the columns.
**2)** Splittability to be processed in parallel.
**3)** Block compression saving storage space vs read/write/transfer performance
**4)** Schema evolution to add fields, modify fields, and rename fields.

## 1. CSV Files:

CSV files are common for exchanging data between Hadoop & external systems. CSVs are readable & parsable. CSVs are handy for bulk loading from databases to Hadoop or into an analytic database. When using CSV files in Hadoop never include header or footer lines. Each line of the file should contain records. CSV files limited support for schema evaluations as new fields can only be appended to the end of a record and existing fields can never be limited. CSV files do not support block compression, hence compressing a CSV file comes at a significant read performance cost.

## 2. JSON Files:

JSON records are different from JSON files, and each line is its own JSON record, and as JSON stores both schema & data together for each record enabling full **schema evolution** & **splittability**. JSON files do not support block level compression.

## 3. Sequence Files:

Sequence files store data in binary format with a similar structure to CSV files. Like CSV, Sequence files do not store meta data, hence only schema evolution is appending new fields to the end of the record. Unlike CSV files, Sequence files do support **block compression**. Sequence files are also **splittable**. Sequence files can be used to solve "small files problem" by combining smaller XML files by storing the filename as the key and the file contents as the value. Due to complexity in reading sequence files, they are more suited for in-flight (i.e. intermediate) data storage.

**Note:** A SequenceFile is Java centric and cannot be used cross-platform.

## 4. Avro Files:

are suited for long term storage with schema. Avro files store meta data with data, but also allow specification of independent schema for reading the file. This enables full **schema evolution** support allowing you to rename, add, and delete fields and change data types of fields by defining a new independent schema. Avro file defines the schema in JSON format, and the data will be in binary JSON format. Avro files are also **splittable** & support **block compression**. More suited in **usage patterns** where row level access is required. This means all the columns in the row are queried. Not suited when a row has 50+ columns and the usage pattern requires only 10 or less columns to be accessed. Parquet file format is more suited for this columnar access usage pattern.

## 5. Parquet Files:

Parquet file is a columnar file like RC and ORC. Parquet files support block compression and optimized for query performance as 10 or less columns can be selected from 50+ columns records. Parquet file write performance is slower than non columnar file formats. Parquet also support limited

schema evolution by allowing new columns to be added at the end.

So, in summary favor Sequence, Avro, & Parquet file formats over the others. Sequence files for raw & intermediate storage. Avro & Parquet files for processing.

## More Hadoop eco system Interview Q&A

**1)** Q7 – Q13 Hadoop Overview & Architecture Interview Q&A.
**2)** Q16 – Q25 Hadoop MapReduce interview questions & answers.
**3)** Q27 – Q36 Apache Spark interview questions & answers
**4)** Q37 – Q50 Apache Flume interview questions & answers
**5)** Q51 – Q61 Hbase Interview Questions & Answers
**6)** Q62 – Q70 HDFS blocks Vs. splits & Spark partitions Interview Q&A

## Popular Posts

♦ 11 Spring boot interview questions & answers

**828 views**

♦ Q11-Q23: Top 50+ Core on Java OOP Interview Questions & Answers

**768 views**

18 Java scenarios based interview Questions and Answers

**400 views**

001A: ♦ 7+ Java integration styles & patterns interview questions & answers

**389 views**

01b: ♦ 13 Spring basics Q8 – Q13 interview questions & answers

**296 views**

♦ 7 Java debugging interview questions & answers

**293 views**

01: ♦ 15 Ice breaker questions asked 90% of the time in Java job interviews with hints

**286 views**

♦ 10 ERD (Entity-Relationship Diagrams) Interview Questions and Answers

**280 views**

♦ Q24-Q36: Top 50+ Core on Java classes, interfaces and generics interview questions & answers

**240 views**

001B: ♦ Java architecture & design concepts interview questions & answers

**202 views**

Bio | Latest Posts

## Arulkumaran Kumaraswamipillai

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via Amazon.com in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site.**945+** paid members. join my LinkedIn Group. **Reviews**

**About** Arulkumaran Kumaraswamipillai

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via Amazon.com in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site.**945+** paid members. join my LinkedIn Group. **Reviews**

‹ 06: Capture throughput & latencies with “Metrics Core” tutorial

02: Q7 – Q15 Hadoop overview & architecture interview questions & answers ›

**Posted in** Hadoop & BigData Interview Q&A

# Empowers you to open more doors, and fast-track

### Technical Know Hows

☀ Java generics in no time ☀ Top 6 tips to transforming your thinking from OOP to FP ☀ How does a HashMap internally work? What is a hashing function?
☀ 10+ Java String class interview Q&As ☀ Java auto un/boxing benefits & caveats ☀ Top 11 slacknesses that can come back and bite you as an experienced Java developer or architect

### Non-Technical Know Hows

☀ 6 Aspects that can motivate you to fast-track your career & go places ☀ Are you reinventing yourself as a Java developer? ☀ 8 tips to safeguard your Java career against offshoring ☀ My top 5 career mistakes

# Prepare to succeed

☀ Turn readers of your Java CV go from “Blah blah” to “Wow”? ☀ How to prepare for Java job interviews? ☀ 16 Technical Key Areas ☀ How to choose from multiple Java job offers?

Select Category

# © Disclaimer



 ↑ 