Richard Taylor Follow

Mar 5, 2018 · 14 min read · Listen

Getting started with Data Engineering

- First I will provide a brief history of the big data sector and why it exists.
- I will then explain what a data engineer is and how and why the role came to be.
- Finally, I will give a big picture overview of the technologies data engineers have been using thus far — and what the future holds.

What Is Big Data 🌎

The term “Big Data” has been so heavily misconstrued by marketing over the years it makes the term “Agile” seem unambiguous and well practised.

It essentially boils down to – Data sets got so huge (Petabytes) that they couldn’t be served from a single machine (with commodity hardware) running an RDBMS (like mySQL, SQLServer, etc). This resulted in new technologies being invented (see noSQL, Hadoop) to solve this problem.

You can only get so far throwing more powerful hardware (CPU’s/RAM) into a single machine (this is called “scaling up” or “vertical scaling”). Therefore many of these new technologies focus on distributing processing/storage load

> **N.B** Before the advent of Hadoop attempts were made at storing massive data sets by running multiple RSBMS instances across multiple machines. These attempts by and large did not end well. The problem was that as the amount of data increased the latency (performance) of the system degraded. This is often referred to as the "Scalability" of the system (which, in the case of a multiple RDBMS system was terrible).
>
> **N.B.N.B** It could be argued that amazons AWS redshift service is an example of an RDBMS that can store large data sets with ease. While it is true that Redshift is based off *PostgreSQL it has been so heavily modified that I don't believe it can be compared to traditional* RDBMS's (It is in fact a columnar database with maximum storage capacity of *~1.6 PB (100 8XL node cluster), compared to Hadoop HDFS at ~100 PB*).

What is a Data Engineer 💻

We can distinguish data engineers from software developers/engineers In the same way we distinguish web/android developers from software developers. All have knowledge of building software; some just have more specialised




> **N.B.** A similar distinction is being made on the operations side of things as well. With job titles such as "*data infrastructure engineer" cropping up.*

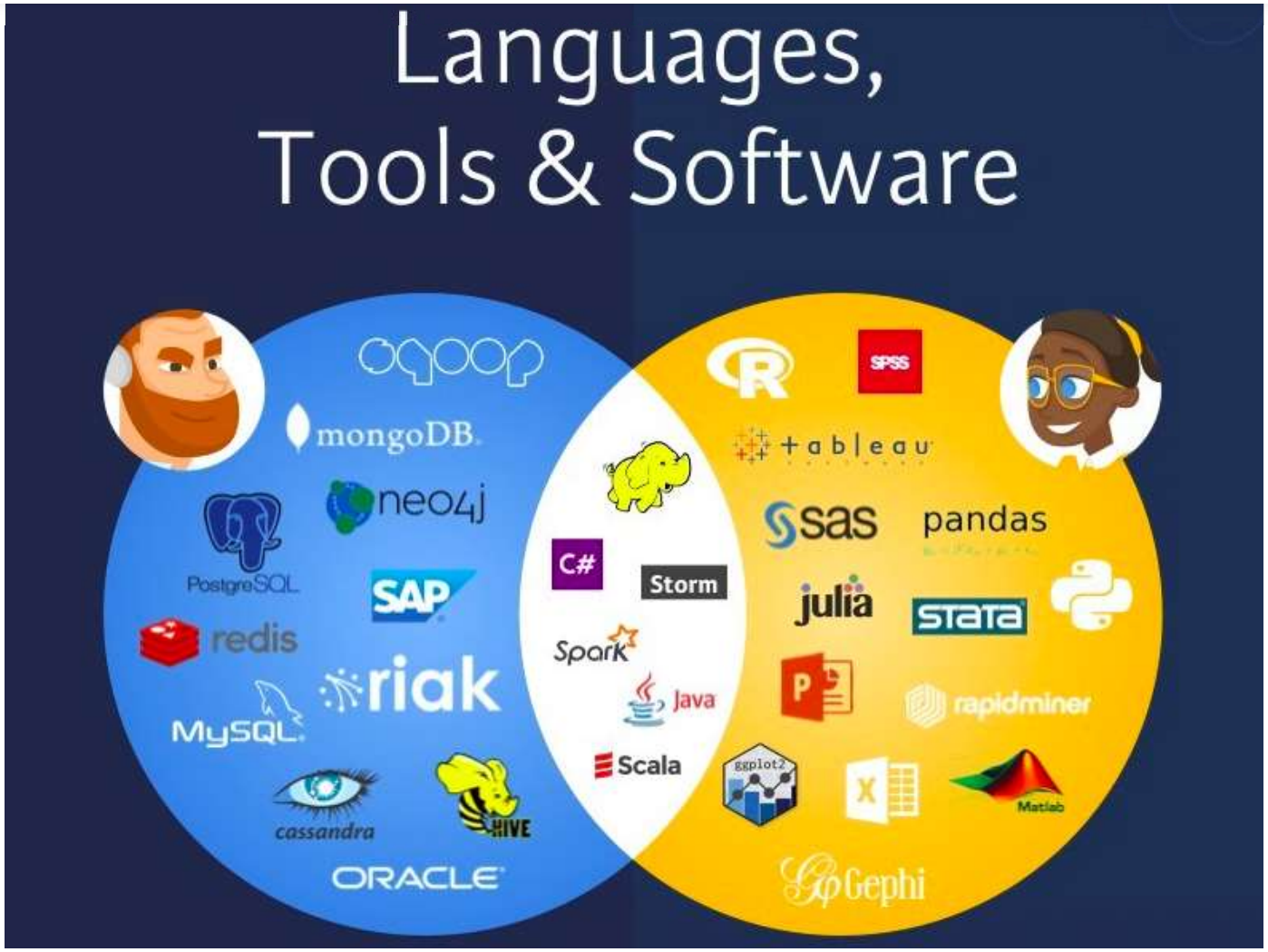
It's worth taking a second to differentiate between two often confused roles in the big data space. That of a "Data engineer" and a "Data Scientist".

A Data Engineer develops, tests and maintains architectures such as databases and large-scale processing systems. A Data Scientist performs analysis on the final output of said systems (that's an oversimplification).

These are two very different specialisations — with Data engineers usually coming from a programming background and Data scientists from a more mathematical background.

Ultimately the job of a data engineer is to ensure the architecture that is in place supports the requirements of the data scientists and stakeholders of the business.



Data Engineer vs Data Scientist:- Source — www.datacamp.com

Like most things in technology big data is a fairly new field, with Hadoop only

the exact definition/skills/job-title of a “data engineer” within the next 100 years or so.

You will find job listings for “Big Data Developer”, “Big Data Engineer”, “Data Engineer”, “Scala Developer” maybe even “Spark Developer” etc. which are all roughly describing the same job or at the very least all come under my definition of a Data Engineer.

Fundamentals 📚 && Ongoing Trends 📈

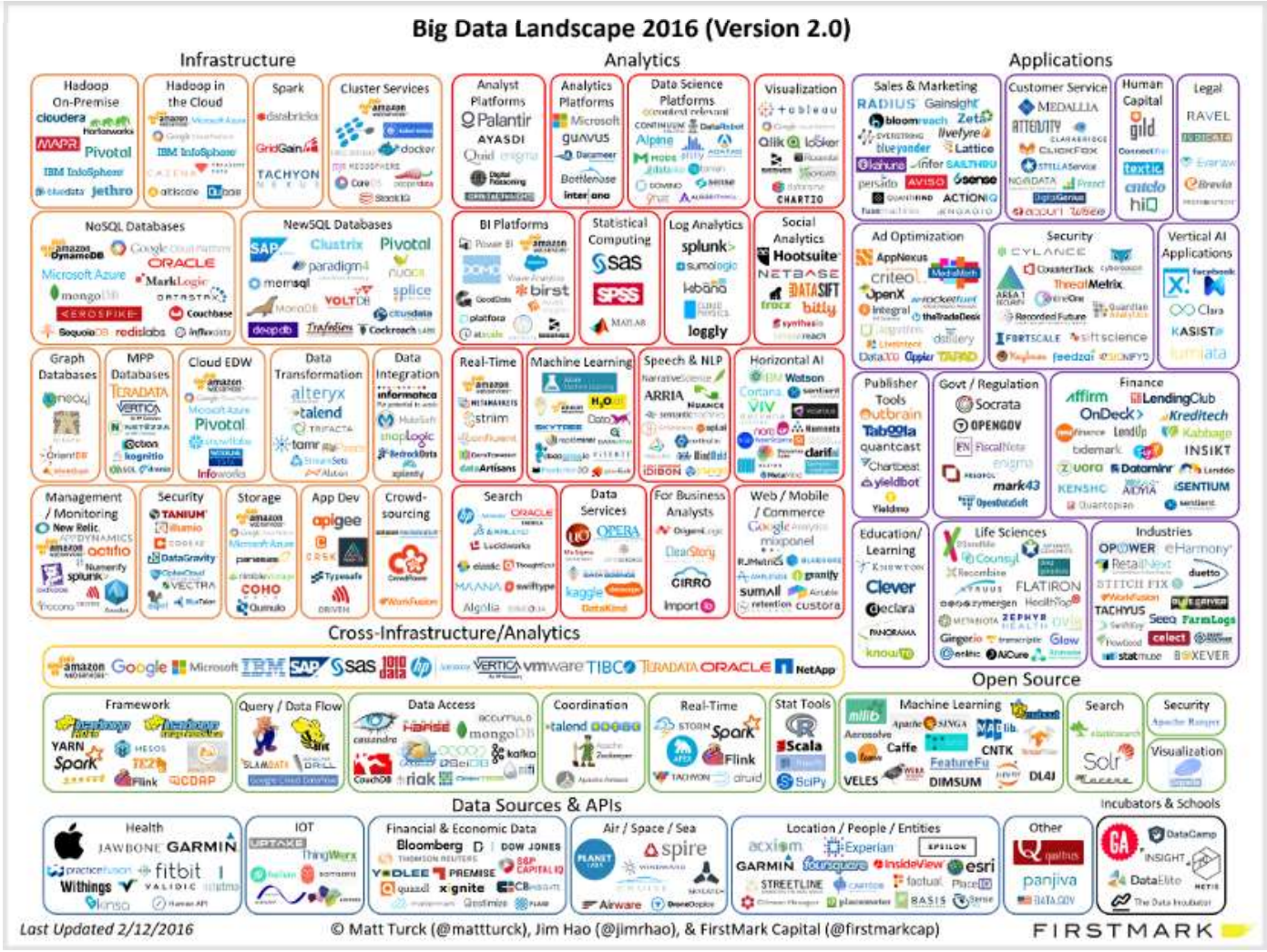
The fundamental computer science fields that underlie data engineering are Concurrency and distributed computing. You will see that many of the most prominent technologies in this field are centred around these challenging concepts (see Hadoop clusters, Spark clusters, noSQL database sharding).

There are 3 important aspects in big data architecture: Scalability, Scalability and Scalability (maintaining performance despite ever increasing volumes of data). You can attribute the inception — and many of the ongoing technological trends in the data engineering space to trying to improve said

- The rise of noSQL database management systems (To solve the inherent scalability problems of relational databases)
- The rise of functional programming paradigms (Make concurrency easier to reason with. Big data processing systems are often — by there very nature — distributed and highly concurrent)

The Big Data Landscape ▲

The following image serves as an overview of the the Big Data field. It can seem… overwhelming:



The big data landscape

If its any consolation, as a data engineer you are mainly concerned with the

In the next couple of sections, I will get into specifics about some of the technologies listed in the open source section of the above image and how they fit into the big picture.

Hadoop

Its hard to talk about data engineering without bringing up Hadoop. As you read on this post will probably start to feel like a “Hadoop getting started” guide. Its true, Hadoop has been — and still is a very central part of the big data field. It is not however the be-all and end-all of big data. You should always analyse the problem you’re trying to solve and decide if you truly need the multi petabyte storage capacity Hadoop offers — for it is not without its drawbacks.

When someone says “we’re using Hadoop to build this system” they may actually be referring to one — or all — of Hadoops modules (usually they are referring to HDFS):

- **Hadoop Common**

- **Hadoop MapReduce**

Hadoop common is the core Java library used for supporting the other Hadoop modules.

HDFS is the backbone of Hadoop — can be thought of as the "cluster storage layer". As the name suggests data is stored much like a file system in a normal computer — except it is distributed across many physical machines.

YARN (Yet another resource manager) is used to manage compute power in a Hadoop cluster. Can be thought of as the "Cluster compute layer". When you run applications such as "Hadoop MapReduce" or "Spark" on your Hadoop cluster, YARN will make sure they have sufficient resources to do so. An alternative to YARN would be "Apache Mesos", however, the later attempts to manage/allocate resources to a much wider scope of applications (web servers etc.) and is not strictly part of the Hadoop ecosystem.

Hadoop MapReduce is an application that performs MapReduce jobs against data stored in HDFS. MapReduce jobs can be written in a number of languages




What made Hadoop so ground-breaking was the scalability of HDFS (Hadoop Distributed File System). Even now nothing comes close to the storage capacity of HDFS which max's out at ~100PB.

Spark ⚡

If you've ever worked in the Javascript space you will know how fast moving technology can be. The big data space isn't quite as fast moving; however, technologies do fall in and out of favour as the data engineering field evolves.

The technology that caused the biggest shakeup of the field since the open-sourcing of Hadoop was, in my opinion, Apache Spark.

Spark manages the processing of massive amounts of data and has largely superseded "Hadoop MapReduce" in the batch data processing field. Spark Core is the central part of Spark and provides all of its general purpose data processing functionality. Spark also has additional libraries for things such as real time data processing (spark streaming) and more.

Spark runs up to 100x faster then Hadoop MapReduce in memory and up to

completed and works backwards to determine the most optimum way to carry them out.

Spark isn't strictly part of the Hadoop ecosystem as it can be run independently of a Hadoop cluster. It is however very common for it to be run as part of one.

Hive 🐝

Hive makes your Hadoop cluster feel like a relational database (in reality it most certainly isn't). It allows you to write SQL (Specifically HiveQL which has a slightly more limited syntax) queries against data stored in HDFS.

It does this by translating your SQL commands into Hadoop MapReduce/Tez commands (depending on which batch processing application you are using). Said commands are then passed to YARN in order to be executed on the cluster.

Hive is only suitable for OLAP (on-line analytics processing) systems. It is unsuitable for performing real-time transactions as part of an OLTP (on-line




Without getting into too much detail Zookeeper is essentially a service that runs inside you Hadoop cluster (or any distributed system for that matter — it isn't tightly coupled to Hadoop) which keeps track of information that must be synchronised across your cluster. Information such as:

- Which node is the master?
- What tasks are assigned to which workers?
- Which workers are currently unavailable?

Zookeeper can be queried by any application running within a Hadoop cluster such as MapReduce, Spark, etc.

Noteworthy Hadoop Tech Mentions 🌟

Here are some Hadoop ecosystem technologies you may encounter in the wild — most of which have fallen out of favour in recent years.

- **Tez:** Is a drop in replacement for Hadoop MapReduce before both were — for the most part — superseded by Spark. Tez utilized a DAG engine (same




- **Pig:** Pig is a platform that allows you to write scripts in a language called "Pig Latin" which had a SQL-like syntax. It served as a more intuitive alternative to writing Hadoop MapReduce code in Python or Java. Once written Pig scripts could be translated into MapReduce commands and then run on your Hadoop cluster.

A brief Hadoop operations tangent 🔧

Instead of hiring a team of data infrastructure engineers to build onsite hadoop platforms many companies use a vendor.

Vendors take a number of forms. Some vendors provide cloud hosted services such as Amazon AWS EMR (Elastic Map Reduce) or Microsoft HDInsight. These services take a lot of the complexity out of setting up and managing your own Hadoop cluster.

Other vendors such as Cloudera/Hortonworks provide support/training/consulting to help companies build their own custom hadoop platforms on premises/in the cloud.

You would be forgiven for thinking: “Great! lets just hook my backend RESTful services into Hadoop and get them to query HDFS using hive when they receive an HTTP request! A petabyte scale single source of truth for my data! Oh and I can run big batch processes against it whenever I want to analyse user trends! Perfect!!”

> If only it were so simple.

Hadoop was designed for storing large amounts of data and running batch processing applications against said data (known as an OLAP system). As such It is unsuitable for serving customer facing applications such as web applications (OLTP system).

Due to this limitation, if you want to make data stored in Hadoop available to customers (assuming said customers are expecting millisecond latency which — in the context of web applications is very likely) the data must be exported from Hadoop into a database more suited to random access queries (HDFS uses sequential queries)(Said databases are sometimes referred to as

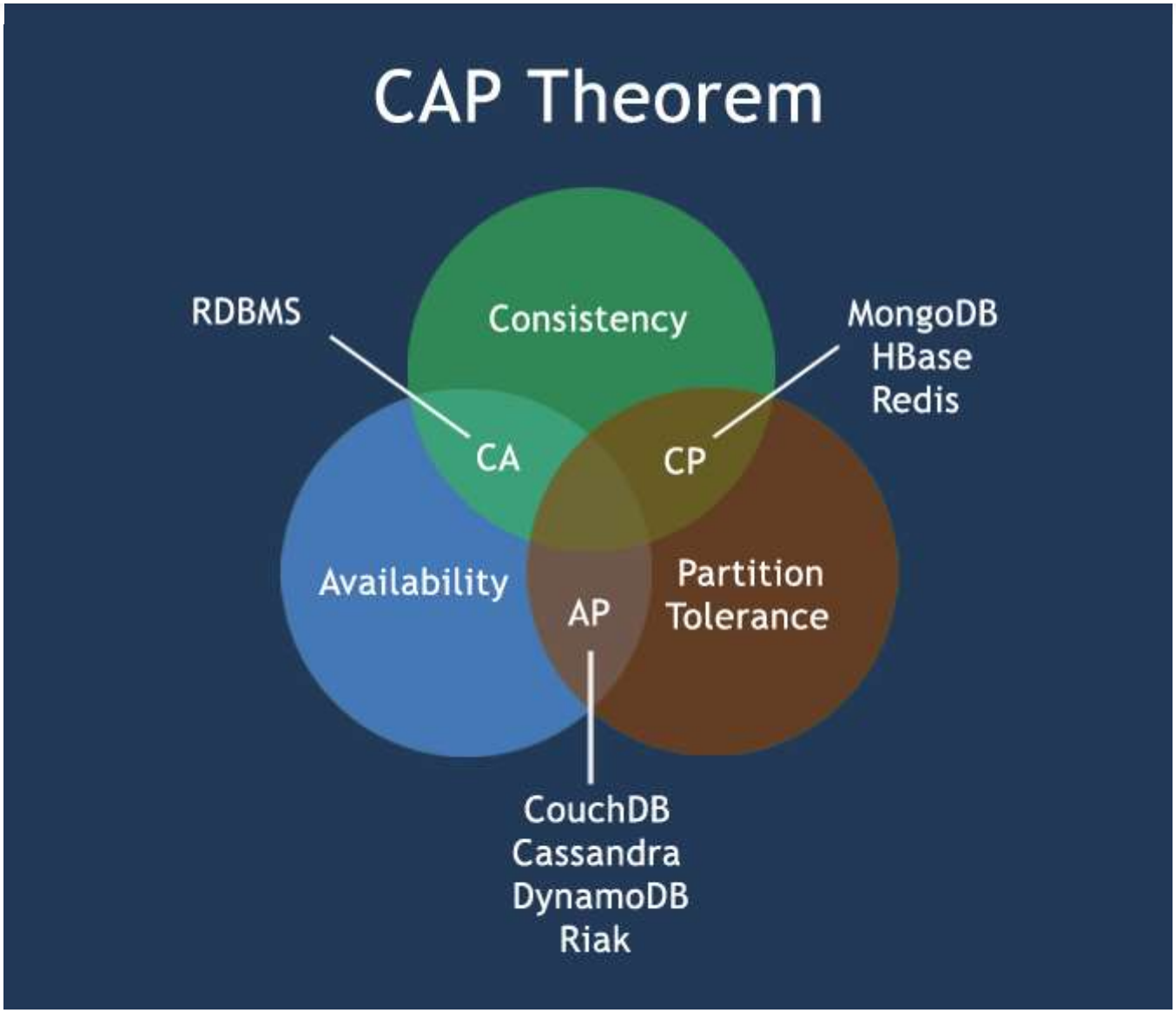
It can be a very difficult decision deciding what type of database to export your Hadoop data to. There is no silver bullet. You will have to weigh up many considerations including:

- **Scale** (Partition Tolerance): How many users are you going to be serving? Will your database fit on a single machine?
- **Consistency**: Do you need to guarantee your users that every read request they make (read request could be triggered by an HTTP request) receives the most recent write?
- **Availability**: Do all requests made by the user need to receive a response? (In an available system, if our client sends a request to a server — and the server has not crashed, then the server must eventually respond to the client. The server is not allowed to ignore the client's requests)

The CAP Theorem can help make this choice easier. As a quick primer; the CAP theorem states that a distributed system cannot simultaneously be consistent, available, and partition tolerant. You must select two.

> *N.B. The Hadoop Distributed File System (HDFS) chooses Partition Tolerance*



CAP Theorem

integrity (most follow the eventual consistency consistency model) in order to work at scale.

> ***N.B.*** *The newest attempt to fix the availability/consistency trade-off that must be made for databases that wish to operate at global scale (be partition tolerant) is Googles "Cloud Spanner" service. Spanner is a CP system that Google describe this as "effectively CA" source. Considering the Spanner project is being led by Eric Brewer — the guy behind CAP Theorem — it may be one to watch. It is however — for now — closed source and super pricey. CockroachDB is an open source alternative attempting to clone it in much the same vain as Cassandra and Hbase cloned Googles Bigtable database.*

Non-Relational (noSQL) Databases 💔

It may be the case that your database is going to serve customers at a massive scale (millions and millions of users). If this is true then chances are you will need more then one physical database server in order to handle not only the number of transactions but store the huge amount of data. This need to scale horizontally to multiple servers makes Partition-Tolerance non- negotiable. This means entering the world of noSQL — and the choice becomes between




- Hbase (CP)
- MongoDB (CP)
- Cassandra (AP)
- DynamoDB (AP)

I wont go into the nuances of each — this post is already long enough, just know there are pros and cons to each.

Relational Databases 💔

If you aren't deploying to massive scale (millions of customers, sometimes referred to as "internet scale") a good old fashioned relational database will perform perfectly well.

A popular tool for transferring bulk data between Hadoop and relational databases is Apache Squoop.

Popular relational databases include:

- MariaDB

Roles & Responsibilities 👔

The exact roles and responsibilites you will perform as a data engineer will vary greatly from company to company. Make sure you read the job spec and remember — every job has periods of uninteresting/monotonous work.

As a Data Engineer you may be involved in projects such as the following:

- **Building ETL (Extract-Transform-Load) pipelines:** Not to be confused with "data ingestion" which is simply moving data from one place to another. ETL pipelines are a fundamental component of any data system. They **extract** data from many disparate sources, **transform** (aka wrangling) the data (often making it fit the data model defined by your data warehouse) then **load** said data into your data warehouse. These are systems built from scratch using programming languages such as Python, Java, Scala, Go, etc.
- **Building metric analysis tools:** Tools used to query the data pipeline for statistics such as customer engagement, growth or operational efficiency.

organisation load or extract data to/from the warehouse (As a gatekeeper of sorts).

The Future 🔮

In this section I will briefly cover how I see the Data Engineering field evolving in the coming years.

Continued HDFS Reliance

For the foreseeable future Hadoop HDFS will remain the backbone of large scale (multiple petabyte) data storage. It is still one of the best if not the only option for storing extremely large amounts of data and performing batch data processing against said data.

Stream Processing

Instead of processing data using batch jobs that run on Hadoop clusters every hour/day there is a movement towards real-time stream processing. This would mean processing data as and when it is received. This allows for more timely action to be taken against the results of said processing. This topic

Although there is a gradual shift towards real time data processing over traditional batch processing — the number of applicable use cases for real time processing is still fairly small. Companies may realise the potential benefits of getting information (processed data) fast (fraud detection is a very popular use case) but the significant cost this incurs is rarely justified.

Brief Terminology Guide 📕

I'm including a terminology guide in this post. It is something I believe would have been very useful to me when I was starting out.

- **Data Warehouse:** The term "Data warehouse" is thrown around a lot in this field. It is a very general term which simply means "a large store of data accumulated from a wide range of sources within a company". The term does not imply the way said data is stored — be that relational or non-relational — nor does it imply any particular data storage technology was used such as Hadoop/Amazon Redshift/mySQL/etc. To be honest the term is used all over the place and you will find many who disagree with my definition.
- **Data Mart:** A subset of the data warehouse that is usually oriented to a

- **Data Lake:** As tempting a[illegible] term as marketing fluff it can serve as a useful ana[illegible] unstructured data (the shallowness of a lake acting as the analogy for flat unstructured data). However, this analogy is problematic in that it implies that the term "data warehouse" must in turn refer to a structured store, which it does not necessarily.

- **OLAP/OLTP:** On-Line Analytical Processing / On-Line Transaction Processing. Terms used to characterise systems. Systems that are designed to perform large batch processing jobs that could take several minutes to several hours are termed OLAP systems. Systems that are designed with low latency in mind (milliseconds) are termed OLTP systems

- **Structured/Unstructued data:** When someone talks about structured data they are usually (or should be!) referring to data stored in a Relational database (Since Relational databases are based on Edgar F. Codd's relational data model which assumes strictly structured data). Unstructed data is just a catch all term for "everything else". Sometimes the term "semi-structured" in used to describe data stored in noSQL databases. This term does make some sense because although noSQL databases do not define schemas that must be strictly conformed to — most of them impose a data model that stored data must conform to.

I hope this has proved to be an insightful introduction to data engineering. I apologise if it felt like a Hadoop guide at times, love it or hate it Hadoop can be difficult to avoid :).

As always feedback — both good and bad — is always appreciated!

External Links/Inspired by/Further Reading 😳

The Rise of the Data Engineer (The first article I read about Data Engineering): https://goo.gl/rZn7hG

Udemy course on Hadoop that inspired most of this post (highly recomended): https://goo.gl/L2deDt

A Beginner's Guide to Data Engineering — Part I (Somewhat similar guide to this one but written from the perspective of a data scientist, great read!): https://goo.gl/S4qfC8

Follow me on Twitter: https://twitter.com/Richard534