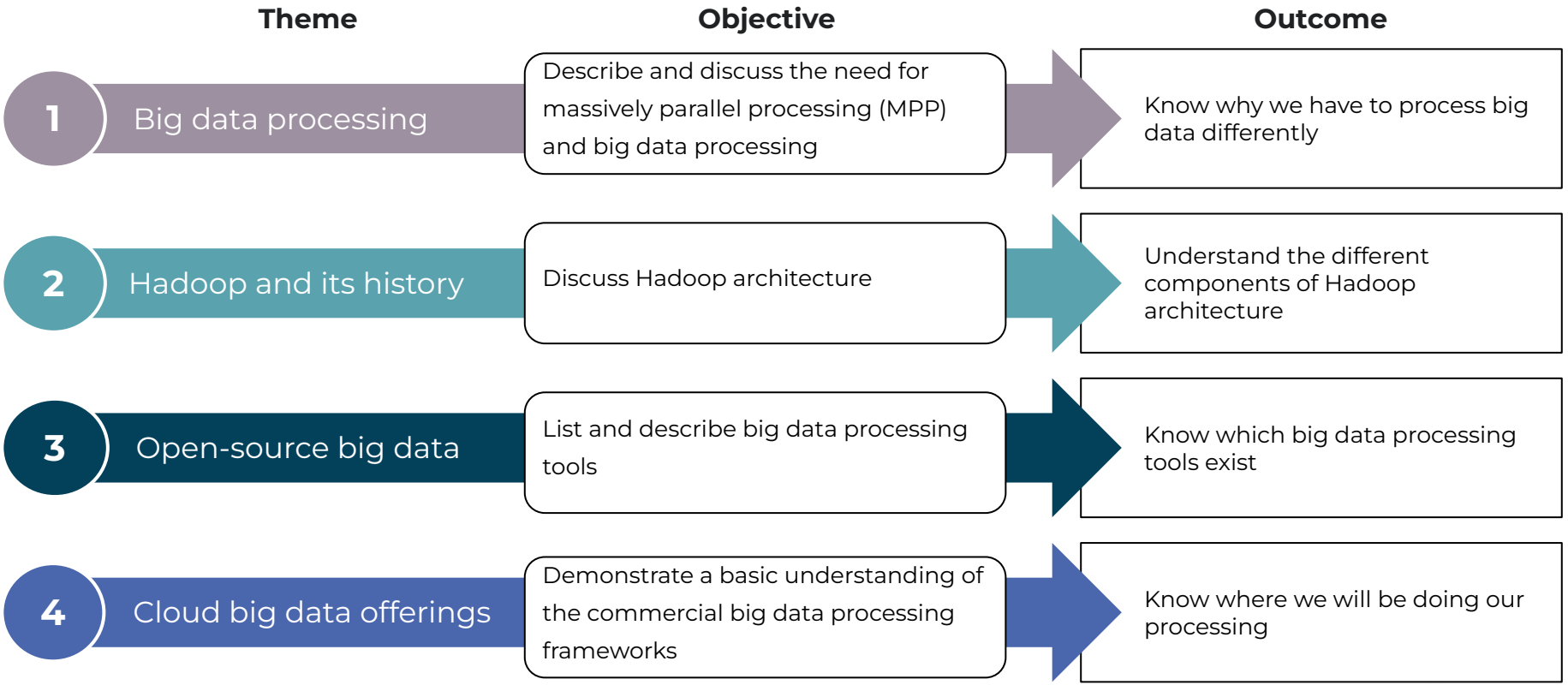
EXPLORE | DIGITAL SKILLS

Introduction to Big Data Processing

Big data train executive summary

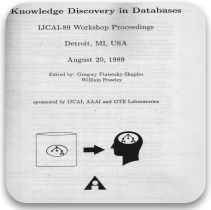


EXPLORE | DIGITAL SKILLS

A brief history of big data and analytics

‘62 - John Tukey suggests a shift in the world of stats

‘64 - US Government plans the world’s first data centre



‘89 - The first in a series of workshops hosted by Knowledge Discovery in Databases is run

‘01 - Software as a Service is launched

‘05 - The term “big data” was labelled by Roger Mougalas

‘06 - Hadoop 0.1.0 released

‘09 - NoSQL introduced and Spark was built

‘19 - Python becomes one of most widely used languages

‘20 - Spark 3.0 released and birth of the lakehouse

‘20 - Big data grows at a rate of 40% per year

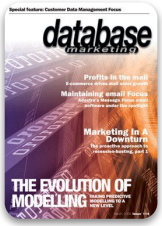
‘25 - More than 20 smart cities globally

1960 — 1970 — 1980 — 1990 — 2000 — 2010 — 2020

iasc isi
International Association for Statistical Computing

‘70 - IBM mathematician Edgar F Codd presents his framework for a “relational database”

‘77 - International Society for Statistical Computing is founded



‘91 - Tim Berners-Lee announced the birth of what would become the Internet as we know it today

‘95 - First super computer created

‘99 - Jacob Zahavi writes about how we’re “mining data for nuggets of knowledge”

‘10 - Hive was created

‘11 - Data lakes preferred over data warehouses

‘13 - IBM: “90% of all data created in last 2 years”

‘15 - Data engineering role is formalised

‘16 - According to Forrester, 40% of firms are implementing big data technology

DELTA LAKE

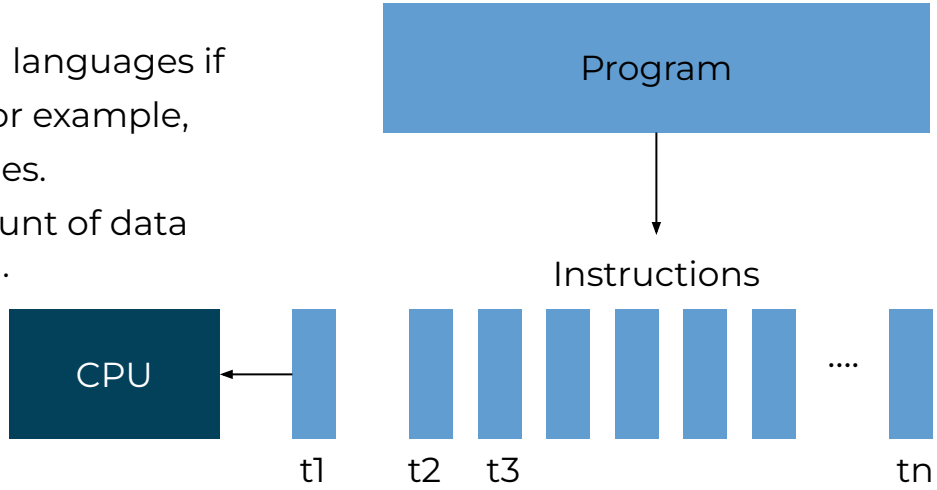
What does it all mean?

Exponential growth of big data	With a growth in the number of devices collecting data that are connected to the internet, big data is set to continue the exponential growth predicted by Gordon Moore.
Limited computational capability growth	Today, due to physical limitations, compute speeds are not improving at the rate predicted by Moore's law.
Required to process data more efficiently	Since the flood of data is only going to increase, we need to process data more efficiently. Enter the data engineer.
Efficient data pipelines required	One of the core components of being a data engineer is not just setting up and maintaining data pipelines but doing so efficiently.
Massively parallel processing (MPP)	One of the toolkits of the data engineer is massively parallel processing (MPP), which forms a central part of the data engineer's day to day and for which many tools have been developed, the main ones which we will discuss in this train.

Serial computation

When computer science first started out, programs were typically written in series, i.e. programs were a **serial set of instructions** for the computer to execute.

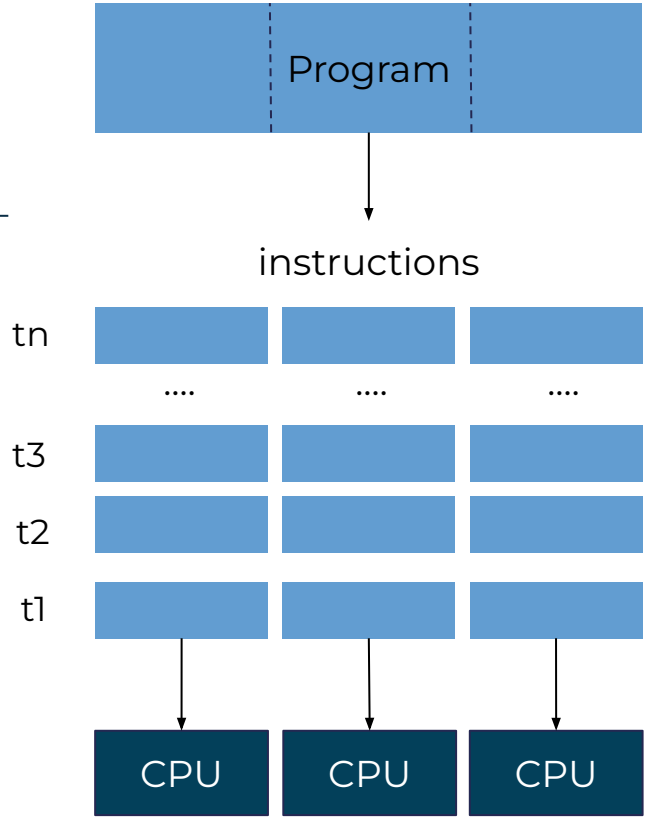
- They were executed on a **single computer** and each instruction was executed in turn until the program was done.
- This is still the case for some programming languages if parallelism is not implemented explicitly, for example, Python performs most computation in series.
- This places a **large bottleneck** on the amount of data that can be processed by any one program.



Distributed computation

In contrast to serial computing, distributed computation leverages **multiple processing units** to perform various tasks in **parallel**.

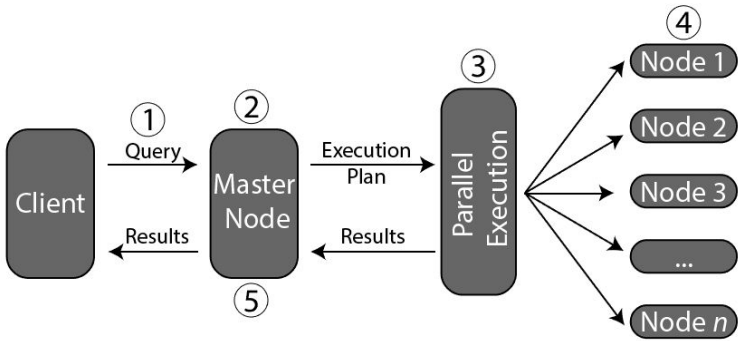
- Programs are broken up into **independent parts** that can be **executed in isolation** and reassembled later.
- These programs can run on many architectures, such as single CPU machines with multiple cores, large servers, or graphical processing units containing thousands of cores.
- The extreme of this is called **massively parallel processing (MPP)**, where hundreds or thousands of nodes (computing units) work in parallel to perform some computation.



Big data processing

Currently there is a massive need for people with toolkits to process big data.

To assist in the processing of big data, **massively parallel processing** techniques have been developed.



A Massively Parallel Processing diagram.

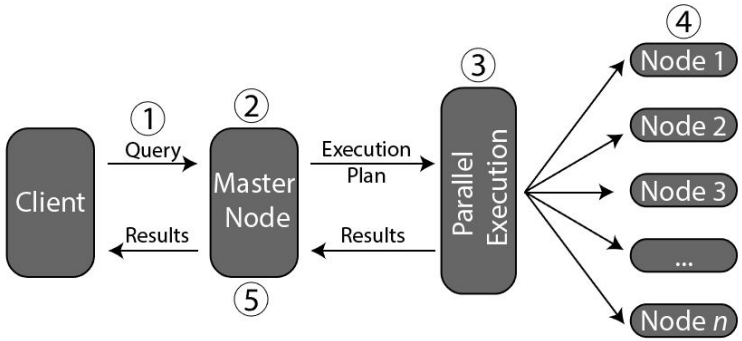
This includes **Apache Hadoop**, **Spark**, and **Hive**.

These technologies leverage the increase of **computational power**, **memory**, and **cloud computing** to achieve the most efficient way of transforming loads of data.

In this train, we will be diving deep into these and other technologies.

Big data processing

Parallel computing is well suited to processing massive amounts of data. To assist in the processing of big data, **massively parallel processing (MPP)** techniques have been developed.



A Massively Parallel Processing diagram.

MPP is specifically applicable to the **processing of big datasets**, since each dataset can be split into smaller subsets that can be processed independently.

Various frameworks have been developed to aid in coding that is natively MPP enabled.

This includes **Apache Hadoop**, **Spark**, and **Hive**.

These technologies leverage the increase of **computational power**, **memory**, and **cloud computing** to achieve the most efficient way of transforming loads of data.

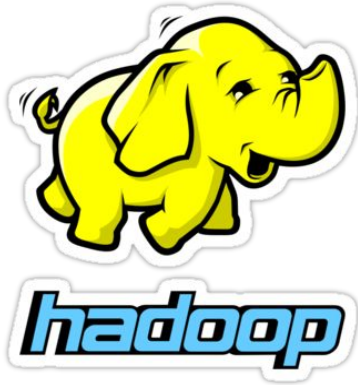
In this train, we will be diving deep into these and other technologies.

What is Hadoop?

Apache Hadoop is a collection of open-source software that is used in the **processing of big data**. Hadoop exploits the parallelisable nature of massive datasets, utilising networks of connected computers, or clusters, to process massive amounts of data.

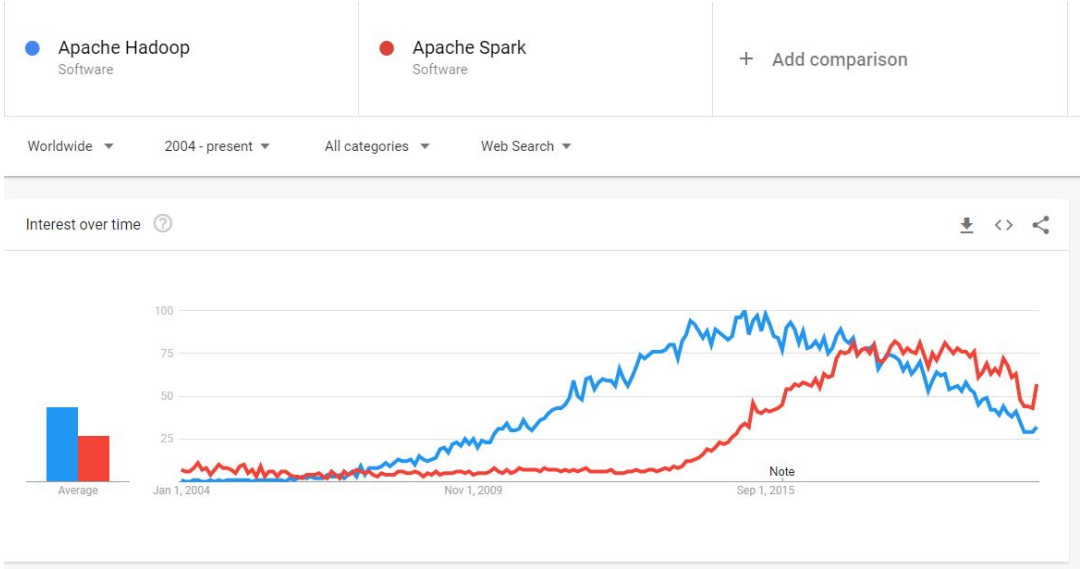
Fundamentally, Hadoop splits files into large blocks and distributes them across **nodes** on a cluster. Instructions for processing are then distributed to the nodes on a cluster to process the data in **parallel**. This approach takes advantage of **data locality**, where the nodes only manipulate the data sent to them, which is faster and more efficient than conventional supercomputer architecture. Each node also only processes a subset of the data, thus creating a hive of workers to process the data at a **much quicker rate**.

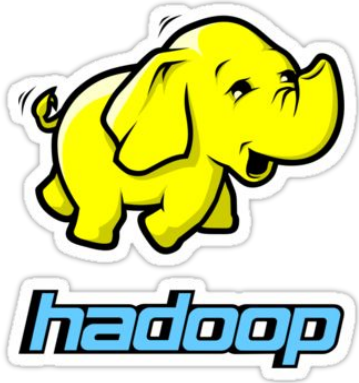
Why use Hadoop?
• Can store and process a very large amount of data. • Can store a variety of data – structured and unstructured. • Fault-tolerant. • Scalable. • Supports SQL tasks (through Hive).



Why learn about Hadoop?

- Hadoop is the ancestor of all big data processing, starting in 2006 based on an idea Google published.
- In recent times, its popularity has been declining due to the emergence of other big data offerings (for example, Apache Spark).
- Most new big data platforms are built on top of some component of the Hadoop system. It is thus important to have an idea of how Hadoop works.



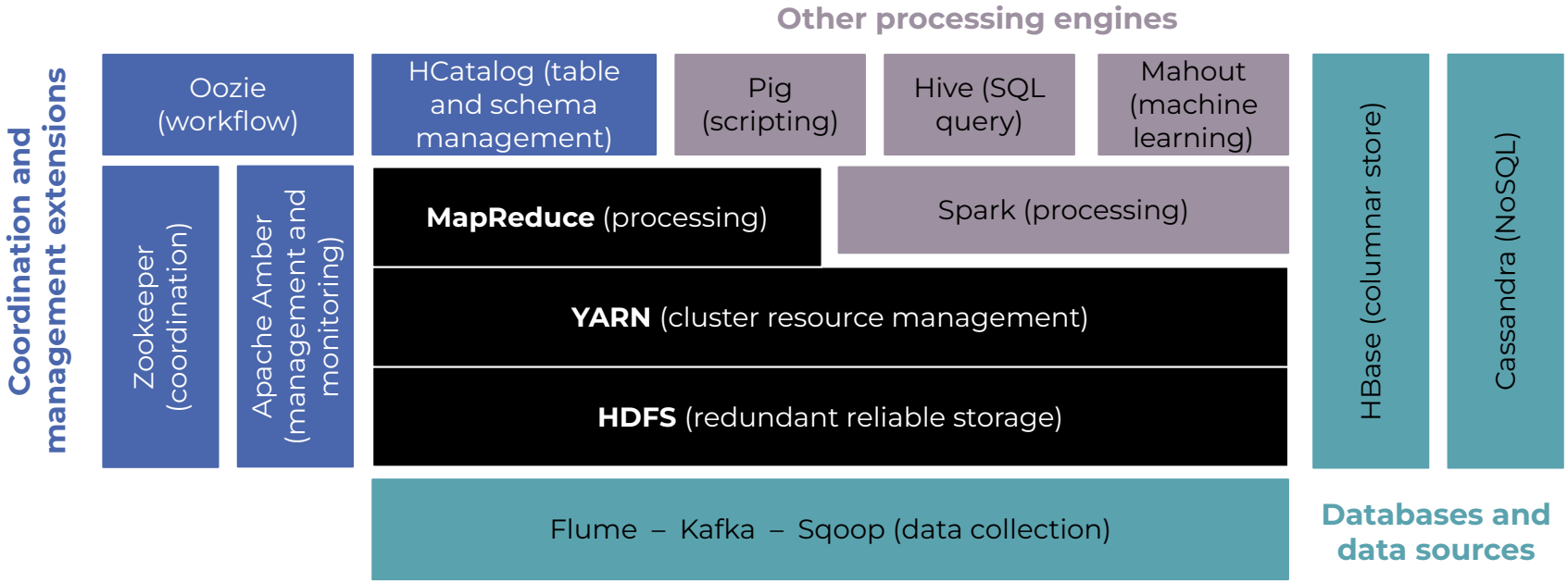


EXPLORE DIGITAL SKILLS

Hadoop components

Apache Hadoop mainly consists of three core components: the **MapReduce** programming model, the **YARN** resource manager, and a storage model known as the **Hadoop File System (HDFS)**.

The Hadoop ecosystem is very large, supporting and being the base of various components:

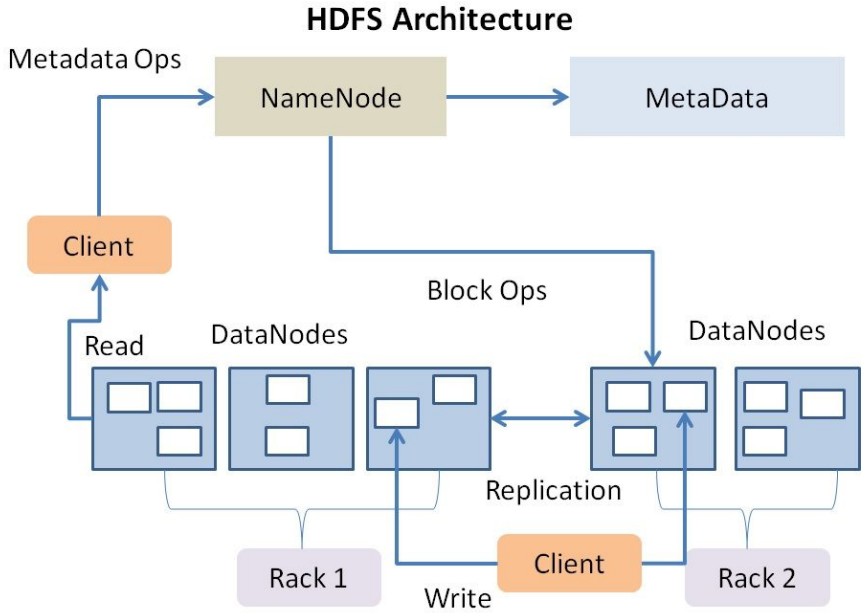


Hadoop Distributed File System (HDFS)

HDFS is the primary data storage system used by Hadoop applications, based on a Distributed File System (DFS).

What advantages are there for using a DFS?

- **Breaks data into separate blocks** and distributes it to different nodes.
- Breaking data into x number of blocks allows it to be processed x number of times in parallel.
- Copies data multiple times and distributes copies to individual nodes.
    - **Fault-tolerant**: nodes that crash can be found elsewhere.
    - Ensures processing can continue while data is recovered.



A HDFS Architecture diagram.

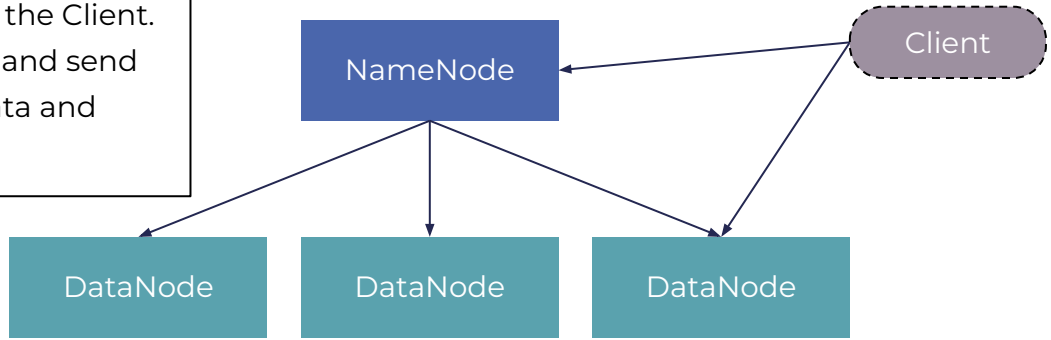
HDFS in detail

HDFS in its simplest form:

- Hadoop has a master node, called the **NameNode**:
  - The Name node is the master ***daemon***.
  - Records metadata.
  - Maintains DataNodes.
- Hadoop has x number of worker nodes, called **DataNodes**:
  - Acts as workers within the cluster.
  - Stores the data.
  - Serves read and write requests from the Client.
- The Client interfaces with HDFS to retrieve and send data to the cluster, writing/reading both data and metadata.

Daemons are background processes, not created through direct interaction of a user.

Here the daemons are created by Hadoop at launch.

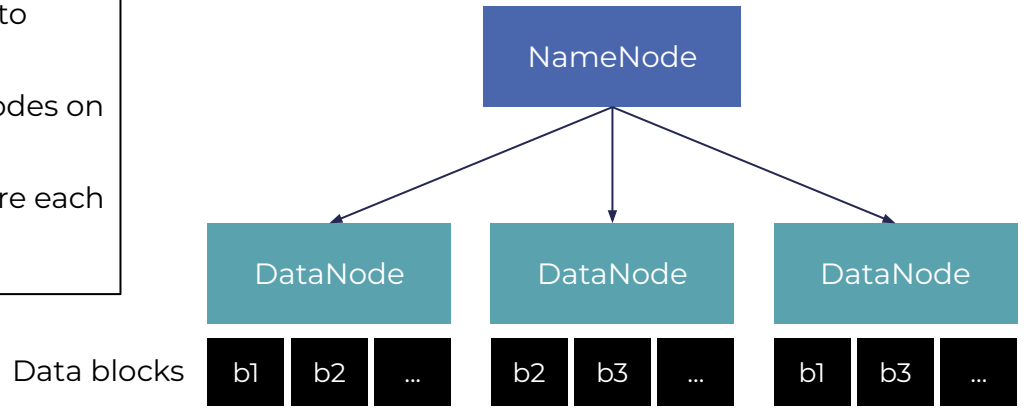


More information read the Hadoop docs: https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html

HDFS in detail

How the **data** is **stored** on HDFS:

- When a file is loaded into HDFS it is split into **blocks** (128 MB blocks – but configurable).
- Each block is stored into one of the DataNodes on the cluster.
- The NameNode contains metadata of where each block of data is located.



How Hadoop deals with **failures** and an increase in load:

- Each block is replicated three times in three other DataNodes.
- Horizontal scaling also allows Hadoop to infinitely scale the storage capacity of the cluster.

The great thing about Hadoop is that all of this happens in the background. We don't have to do or worry about anything just mentioned.

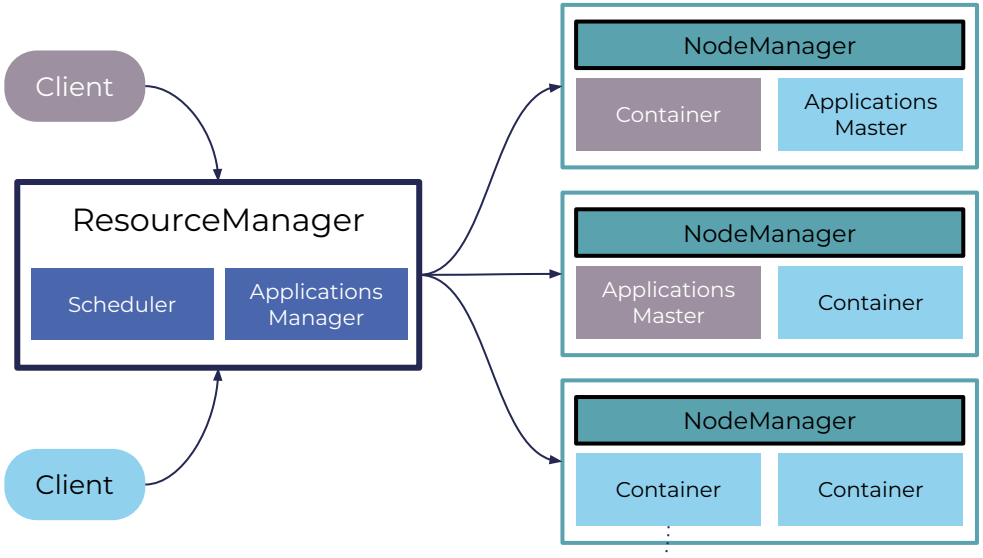
Yet Another Resource Negotiator (YARN)

YARN helps with job scheduling of various applications and resource management in the cluster.

Why YARN works:

- **Efficiency**: Separates the resource management layer from the processing layer:
  - This greatly increases the efficiency of the system.
  - Progressive resource allocation allows for optimal utilisation of available resources.
- **Multi-tenancy**: YARN allows different workloads to be run on Hadoop, such as batch, stream, graph, and interactive, in addition to just using single batch-focussed MapReduce processing, which was the focus of initial Hadoop deployments.
- **Scalability**: As with HDFS, a very large number of nodes can theoretically be added to the cluster to increase the Hadoop cluster processing capacity.

YARN in detail



Apache Hadoop YARN architecture.

Physical resources like CPU, memory, and storage on a single node.

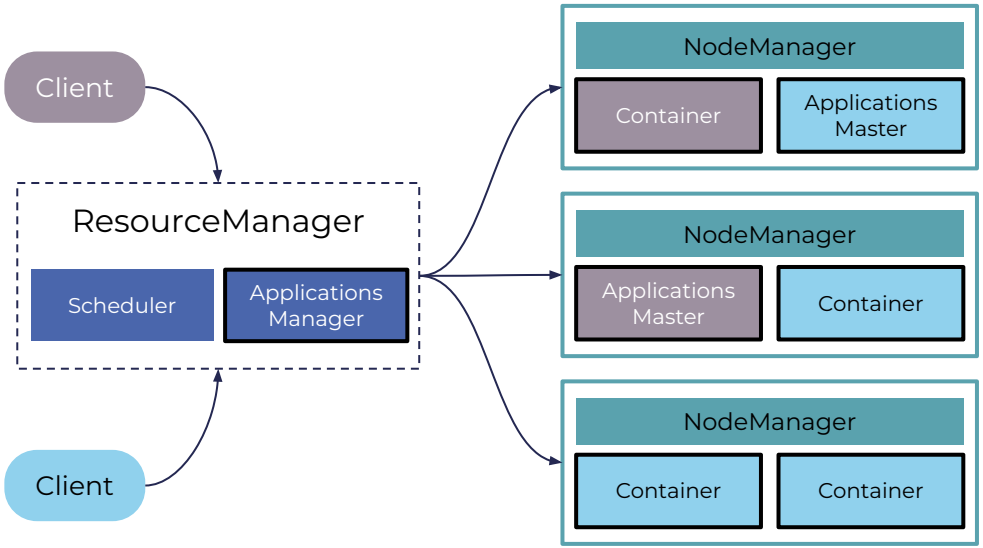
ResourceManager:

- Consist of Scheduler and ApplicationsManager.
- Master daemon.
- Accepts and schedules jobs, and allocates resources.
- Optimises resource allocation through resource allocation and arbitration.
- **Scheduler** allocates resources to running applications subject to resource availability.
- ApplicationsManager accept jobs and negotiates the first container for executing the application specific **ApplicationsMaster**.

NodeManager:

- Monitoring and reporting agent for ResourceManager.
- Manages individual node container and its resources.
- Creates container process when requested by ApplicationMaster.

YARN in detail



Apache Hadoop YARN architecture.

ApplicationsMaster:

- Monitors and tracks applications (single jobs).
- The ApplicationsMaster is responsible for negotiating resources from the Scheduler.
- Requests a Container from NodeManager by sending a Container Launch Context (CLC) which contains everything the application needs to run.

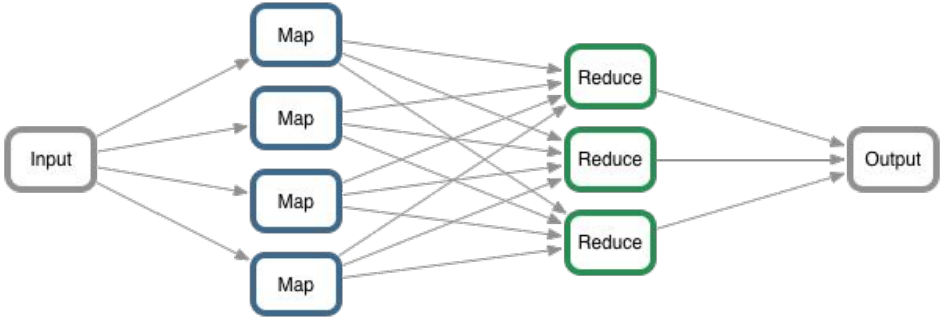
Containers:

- Physical resources like CPU, memory, and storage on a single node.
- Launched by the CLC, which is a combination of metadata, including environment variables, security tokens, and dependencies.

MapReduce

MapReduce processes large, unstructured datasets with a distributed algorithm on a cluster.

This is the fundamental innovation of Hadoop.



01

Splitting

Divide data into fixed-size splits

02

Mapping

Data in each split passed to a mapping function

03

Shuffling

Consolidate the relevant records from Mapping phase output

04

Reducing

Output values from the Shuffling phase are aggregated

Open-source big data engines

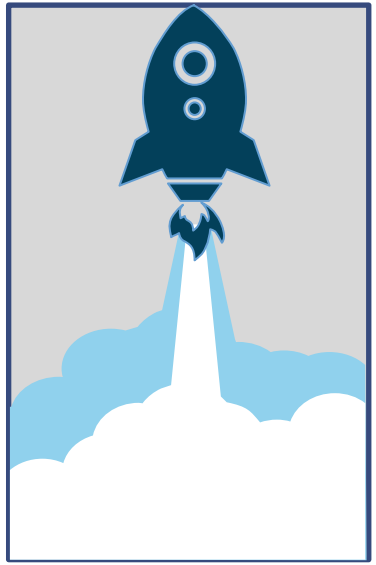
There is a massive list of applications that are being developed for big data processing in the open-source space, specifically as part of the Apache Software Foundation.

Here we want to look at six emerging big data processing, warehousing, or machine learning tools.

Apache Spark (3)
General engine for large-scale data processing

Apache Hive (2)
Data warehousing for distributed storage

Apache Storm (1)
Distributed real-time processing



Apache Flink (4)
Expressive, declarative, fast, and efficient data analysis

Apache Samza (5)
Processing streams from publish-subscribe systems

Apache Samoa (6)
Machine learning for big data streams

Apache Storm

Apache Storm is an **open-source distributed real-time computation system**. It claims to do for real-time processing what Hadoop did for batch processing.

APACHE STORM™

Distributed • Resilient • Real-time

Its applications include real-time analytics, online machine learning, continuous computation, ETL, and distributed remote procedure call (RPC).

RPC allows a program to call a subroutine on another machine.

Apache Storm can integrate with any queueing system and any database system, e.g. Apache Kafka and Amazon Kinesis.

A streaming platform that uses publish-subscribe message brokering for client read/write of data streams.

kafka

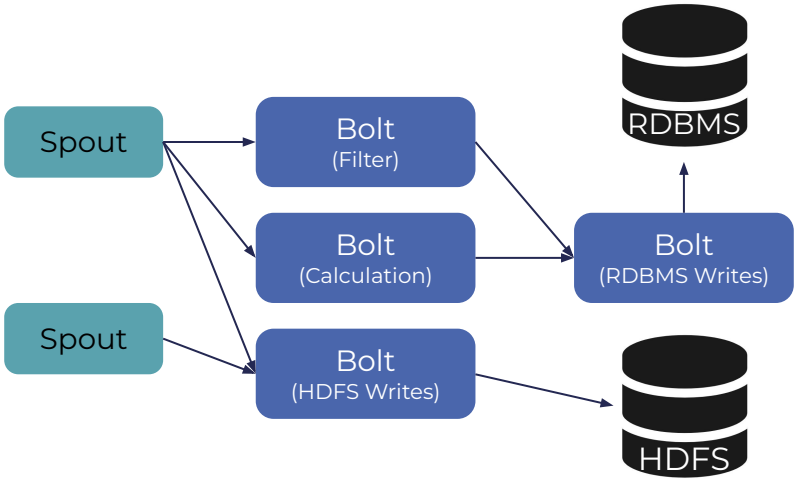
A fully managed streaming platform built in AWS.

Amazon Kinesis

EXPLORE DIGITAL SKILLS

Apache Storm in detail

When programming in Apache Storm you manipulate and transform streams of tuples. A tuple is a named list of values. There are three abstractions in Apache Storm: spout, bolt, and topology.

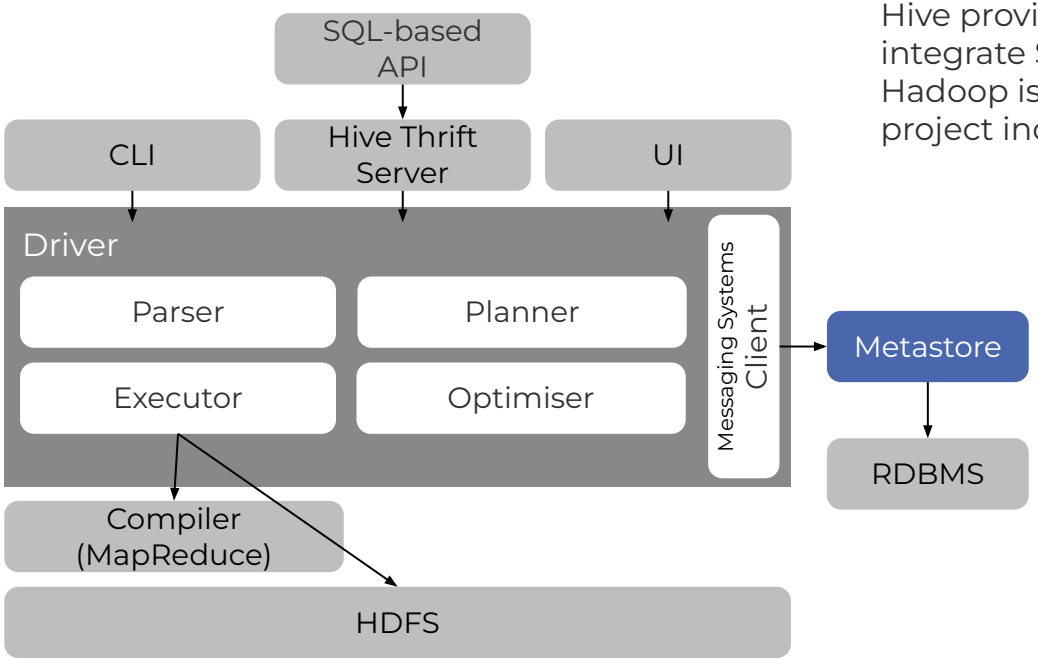


Apache Storm topology.

- A **spout** is a source of streams in a computation, typically reading from a queueing broker; they are input nodes.
- A **bolt** processes any number of input streams and produces any number of new output streams. Most computation logic goes into bolts, e.g., functions, filters, joins, aggregations, and talking to databases.
- A **topology** is a network of spouts and bolts, with each edge in the network representing a bolt subscribing to the output stream of some other spout or bolt.

Apache Hive

Apache Hive is a warehouse project that is built on top of Apache Hadoop, creating a simple interface for interacting with data stored in HDFS.



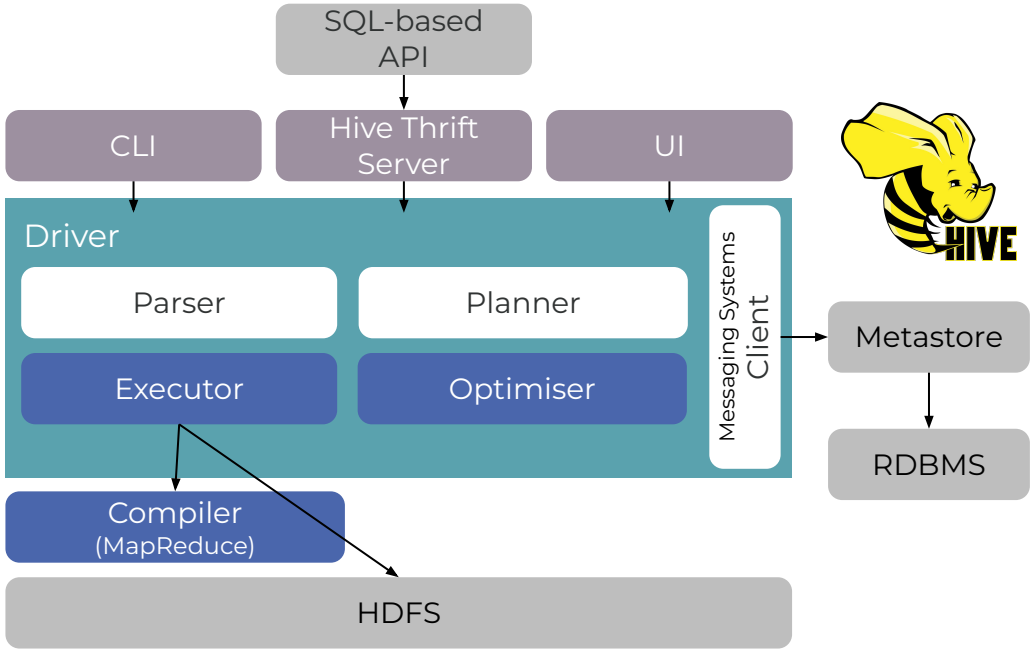
Apache Hive architecture.

Hive provides a SQL-like interface – an abstraction to integrate SQL queries with the underlying Java on which Hadoop is built. Some of the contributors to the Hive project include Facebook, Netflix, and Amazon.

Metastore:

- The central repository that stores the table metadata (not the actual data).
- Metadata stored include its location and schema, information on data partitioning, as well as other data that the developer decides is important.
- The metadata keeps track of the data, replicates it, and provides a backup in the case of data loss.

Apache Hive in detail



Apache Hive architecture.

Driver:

- Acts as a controller by receiving statements to be executed, monitoring progress of execution, and creating sessions.
- Stores metadata created during execution of HiveQL (Hive query language) statements.

Compiler

- Converts the HiveQL query into a MapReduce input, including a method to ensure that the output is in the format as needed by MapReduce.

Optimiser:

- Performs various transformation steps for aggregation and pipeline conversion by a single or multiple joins.
- It also has to split a task while transforming for improved efficiency and scalability.

Executor:

- Executes tasks after compilation and optimisation.
- Interacting directly with the Hadoop Job Tracker for scheduling tasks to be run.

CLI, UI, Thrift Server:

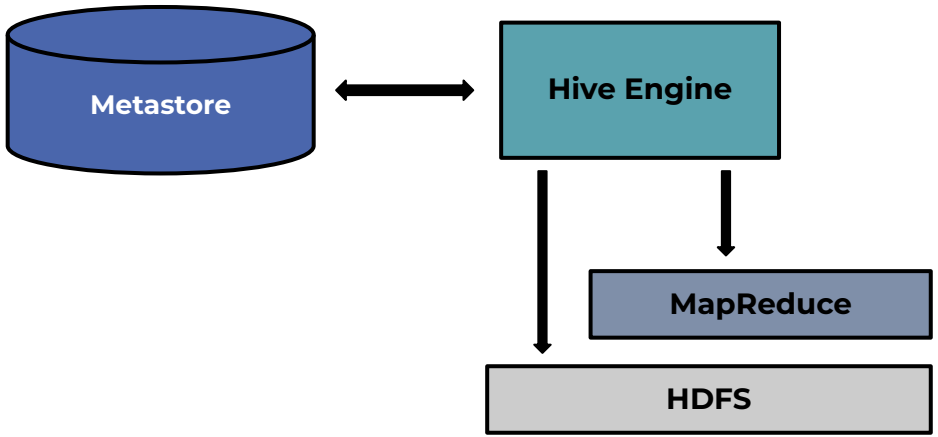
- Command Line Interface (CLI) and User Interface (UI) submit queries and process monitoring of instructions for external users to interact with Hive.
- Thrift Server lets other clients interact with Hive.

Apache Hive

A data warehouse system for data summarisation, querying, and analysis.

How it works

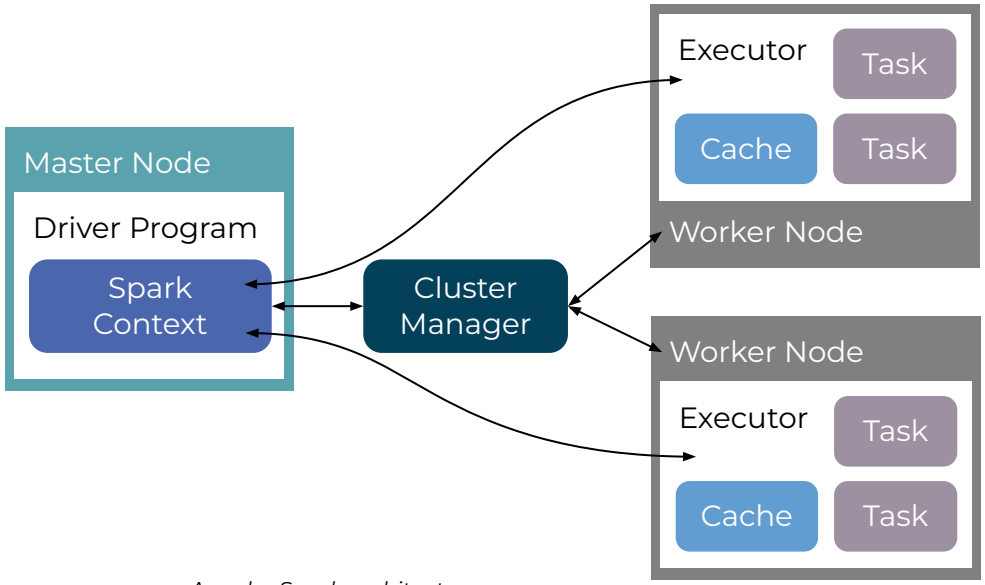
- **Metastore provides structure** by applying table schema on top of data:
  - Provides relational view.
- Has a SQL-like programming interface called HiveQL:
  - HiveQL automatically **translates SQL-like queries into batch MapReduce jobs**.
- Apache Hive Metastore can be built and **implemented on top of** other big data offerings such as **Hadoop** and **Spark**, allowing you to query distributed data in a SQL interface.



Apache Spark

Spark is a unified analytics engine for the processing of big data that provides high-level APIs in Python, Scala, and Java.

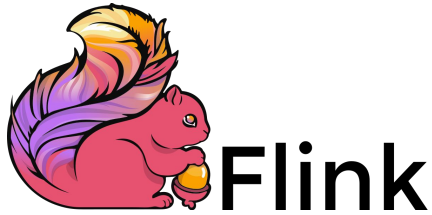
Spark allows processing of batch, streams, and graphs and performing machine learning through various APIs. It follows a simple architecture that is only focussed on the computation implementation.



Apache Spark architecture.

- The **Master Node** contains a driver program that coordinates the execution of tasks.
- Within a Master Node and driver program, a **Spark Context** is defined within which all the execution will occur.
- Cluster resources are maintained through a **Cluster Manager**.
- The Cluster Manager manages **Worker Nodes**, which have one or more tasks executed through the Executor.

Apache Flink



Not to be confused with a cute Pokémon (more here), Apache Flink is an open-source unified stream and batch data processing framework.

The core of Flink is a **dataflow engine** written in Java and Scala. It executes arbitrary dataflow programs in a **data-parallel** and **pipelined** manner with execution of both batch and streaming possible. The dataflow programming model provides event-at-a-time processing on both finite and infinite datasets.

Similar to other big data solutions, Flink comes with high levels of fault tolerance and exactly-once semantics for streaming workloads. Flink supports APIs in Java, Scala, Python, and SQL and it is capable of compiling and optimising code to run on clusters in the cloud.

It consists of two core APIs, a DataStream API for bounded and unbounded data streams and a DataSet API for bounded datasets. Additionally, it supports a Table API, which provides SQL access for relational processing that can be embedded into the DataStream and DataSet APIs.

On execution, Flink programs are mapped into dataflows, which start with one or more sources and terminate with any number of transformations in between. It can be represented as a **directed acyclic dataflow graph**, allowing the application to branch and merge dataflows.

Apache Samza

Apache Samza is an open-source, near real-time, asynchronous framework for the processing of streams. It is developed in Scala and Java.

Samza allows its users to build **stateful applications** that process data in near real-time for Apache Kafka, as well as other sources. Writing in streams allows for sub-second response times.

Samza processes data in **streams**, with a stream consisting of a collection of **immutable messages.** Messages are usually of the same type and modelled as a key-value pair. Streams can have multiples producers or consumers and can be unbounded (Apache Kafka) or bounded (HDFS).

As with Apache Flink, the stream application processes messages from the input streams, transforms them, and emits results to an output stream or database.

It is currently being used by large companies, including LinkedIn, Slack, and TripAdvisor.

Apache Samoa

Still in the incubator.

Apache Samoa is a distributed streaming machine learning (ML) framework that contains a programming abstraction for distributed streaming ML algorithms.

Essentially, Samoa takes the computational advances and advantages of Apache Storm, Apache Samza, and Apache Flink and provides an abstracted view to build **machine learning models on streams**.

It provides a collection of distributed streaming algorithms for the most common data mining and machine learning tasks, such as classification, clustering, regressions, and common operational implementations required to build more complex algorithms.

But where do we actually run this?

All of the above solutions are just **software tools**, which still have to run on hardware, and Apache is not a cloud infrastructure provider...

You have three options:

1. Buy infrastructure, and host the software and servers yourself.
2. Go with an enterprise-grade system. This will not just include a robust hosted solution but also on-demand support and a tailor-made solution.
3. Choose from a range of cloud providers, providing either infrastructure to host the services on, or an already-hosted platform.

Let's dive a bit deeper into **enterprise** and **cloud solutions**.

EXPLORE DIGITAL SKILLS

Enterprise-grade analytics

CLOUDERA

Let's consider an example of an enterprise cloud.

Cloudera is a commercial offering, which provides a software platform for data engineering, data warehousing, machine learning, and analytics.

Its first implementation was an implementation of Apache Hadoop, specifically optimised and built to serve enterprise-class deployments. While being a commercial tool, Cloudera claims to donate more than 50% of its engineering output to Apache-licensed open-source projects.

Cloudera offers various bundles which are either on-premise or within a cloud platform. These offerings range from simple database solutions to more involved complete data engineering and science solutions. More information can be found on their website.

There are numerous cloud providers specifically focussed on enterprise or big data solutions, including Snowflake, Oracle, and IBM.

Using any of these provides will largely depend on the company you work for. For more generic applications, let's look at the biggest players in the cloud computing market.

Cloud big data processing offerings

While not always directly comparable, the major cloud providers all have solutions in data storage, processing, and presentation.

Let's have a look at the processing solutions.

Cloud big data offerings

While not always directly comparable, the major cloud providers all have solutions in data storage, processing, and presentation.

	AWS	Azure	GCP
Data Warehousing	Amazon Athena	Azure Synapse Analytics	Google BigQuery
Big Data Processing	amazon EMR Amazon Elasticsearch Service	databricks	Google Dataflow Spark
Dashboarding	amazon QuickSight	Power BI	Looker Google Data Studio

Amazon Web Services: Storage and dashboards

Amazon Athena

Lakehouse solution

Lakehouses combine the best of data lakes and data warehouses. It provides high-level data abstractions and interfaces typically seen in data warehouses while leveraging data lakes subliminally, so it's easier to manage and more cost effective.

- Amazon Athena provides **interactive query service** to analyse data in blob storage (Amazon S3) using standard SQL.
- It is **serverless**, so you only pay for your usage.
- Anyone with SQL skills can analyse large-scale datasets.
- You still have the flexibility that allows data scientists to analyse the underlying data in the data lake, while analysts can use the same data for reporting.

Amazon Quicksight

Dashboarding

Being able to visualise data is critical when working with big data. It can provide visualisation of ingested data to business analysts or of metrics from system operations to data engineers.

- Provides scalable, serverless, embeddable, machine learning-powered Business Intelligence (BI) service.
- Create and publish interactive BI dashboards that include machine learning-powered insights.
- Serverless, only pay when users access your dashboards or reports.

Amazon Web Services: Processing

Amazon Elasticsearch Service

Managed Searching and Indexing

Elasticsearch is a distributed open-source search and analytics engine built on Apache Lucene. It can index a wide range of data types, including textual, numerical, geospatial, structured, and unstructured. Once indexed, Elasticsearch can run complex queries to retrieve data at lightning speeds.

- Deploy, secure, and run Elasticsearch at scale.
- Uses **open-source Elasticsearch APIs**, managed instances of Kibana (for building dashboards on top of Elasticsearch), and it includes built-in alerting and on-demand or scripted SQL querying.
- As with most AWS services, you can achieve **network isolation** in a virtual private cloud (VPC), along with state-of-the-art data encryption.
- Only pay for your own usage in Elasticsearch.

Amazon Elastic MapReduce (EMR)

Big Data Processing

Amazon EMR is a big data platform that allows hosting of previously discussed open-source data processing tools. EMR lets Amazon do most of the maintenance and accommodates scaling into the petabyte realm.

- Big data processing using **open-source tools** such as Apache Spark, Apache Hive, and Apache Flink.
- Abstracts and **simplifies** the setup of clusters and other time-consuming tasks.
- Can run on Amazon EC2 instances, Amazon Elastic Kubernetes Service, or on-premise using AWS Outposts.
- Since cluster management is abstracted, tuning and monitoring are done for free.
- EMR allows you to **scale** vertically or horizontally within minutes while using a tailor-made setup through AWS AMIs.

Microsoft Azure: Storage and dashboards

Azure Synapse Analytics

Data warehousing, integration, and big data analytics

It brings together the best of having an integrated cloud environment. Built on top of an Apache Spark engine, queries are lightning fast. Synapse is connected to the data lake while allowing SQL-like queries and utilising the latest in lakehouse architecture.

- Combines data warehousing and big data systems, best of both worlds for data scientists and analysts.
- Azure Synapse is easy to use and easily integrated with cloud SQL technologies.
- **Accelerates processing** using an underlying Spark engine.
- Full-fledged data pipelines allow Synapse to be a single point of entry for your full end-to-end data engineering process.
- **Deep integration** with other databases and toolkits such as PowerBI, CosmosDB, and AzureML.

PowerBI

Dashboarding

PowerBI has become one of the most popular dashboarding technologies. Azure PowerBI is a hosted service that allows multiple users to view reports while ensuring performance through underlying Azure virtual machines powering the reports.

- Unified self-service and enterprise analytics solution.
- **Quickly create dashboards** from any of your underlying datasets, databases hosted in the cloud, or even metrics generated by other applications.
- Combined with Azure Analytics, PowerBI provides **insights at scale**, allowing users to develop data-driven cultures within their organisations.

Microsoft Azure: Processing

Databricks

Enterprise analytics

Want to make your data scientists happy? Implement Databricks. From the creators of Apache Spark, it's a managed notebook environment, which comes with the best in data lakehouse solutions and ridiculously fast SQL connectors.

- SQL Analytics: interface with **SQL-like queries** to databases and dashboarding capabilities.
- Databricks Workspace: **interactive workspace**, using Apache Spark.
- Code in Python, Scala, Java, and SQL.
- Collaborative workspace for data engineers, scientists, and machine learning engineers.
- Access to **structured and unstructured** datasets through Databricks Delta Lake.

* Recently, Databricks is also available on AWS and GCP

Stream Analytics

Real-time analytics

With data only increasing in velocity, we need to be able to process data at the speed of light. Azure Stream Analytics is a processing engine for processing high volumes of fast-streaming data from multiple concurrent sources.

- Real-time analytic and complex **event-processing engine**.
- Supports **very large volumes of data**.
- Compatible with a multitude of data sources and can interact with them in parallel.
- Extraction of insights from disparate datasets in real time and triggering of subsequent actions.
- Able to **extract real value** from remote and IoT devices.

Google Cloud Platform: Storage and dashboards

BigQuery

Data warehousing

BigQuery is a fully managed cloud data warehouse. This all-in-one solution allows serverless warehousing, access to data stored in object storage, as well as running data science workloads directly on the BigQuery API.

- **Serverless**, highly **scalable**, and cost-effective multi-cloud data warehouse.
- Analysis of petabytes of data using ANSI SQL with zero operational overhead.
- Supports a **variety of workloads**, including BigQuery ML, BigQuery GIS, BigQuery BI Engine, and Connected Sheets (connection to Google Sheets).

Looker

Dashboarding

Google Cloud Platform joins the other cloud front-runners in providing a hosted dashboard solution. Looker has the advantage of leveraging multi-cloud and on-premise integration and built-in analytics.

- Business intelligence, data applications, and embedded analytics.
- Supports **multi-cloud setups**.
- Abstracts the underlying data complexity to create a common data model.
- Allows for embedded and augmented analytics as well as tailored data applications.

Google Cloud Platform: Processing

Dataflow

Data pipelines

Google Dataflow is a big data processing tool that applies the best in streaming and autoscaling, allowing you to query and manipulate data using SQL, as well as integration with notebooks.

- Stream and batch data processing tool.
- **Serverless pricing** makes it cost-effective.
- Fast ELT/ETL transformations through **autoscaling** of resources and dynamic work rebalancing.
- Flexible scheduling and pricing for batch process and ready-to-use real-time AI patterns.

Dataproc

Big data processing

Managed analytics and processing is what Dataproc brings to the table. Dataproc gives you managed clusters, which can run various versions of the software suites along with preconfigured workflow templates.

- Use of open-source data and analytic tools (Apache Hadoop, Apache Spark, and others).
- Fully managed instances of the above toolkits.
- Pay-as-you-use basis and supporting built-in enterprise-grade security.

Conclusion

We just sprinted through many of the technologies, solutions, and implementations of big data processing that are available to a data engineer at the moment.

This may seem like a lot to take in, but for now, you don't have to worry about remembering all of these. Chances are that the best new processing tool will emerge within the next year, and you will have to learn it anew.

What is important from this train is that you are aware that there are many ways to do things and many toolsets to do it with!

So instead of getting stuck on the specific toolset, rather get stuck in the problem you are trying to solve.