Chapter 1
Introduction to Data Engineering

Contents:

- DE Introduction
- Types of data, Big Data
- Data wrangling & characteristics
- ETL/ELT models and differences
- On Data Warehouses
- Fundamentals of Data Pipelines & different types of DPs
- Design of a Data Pipeline – Detailed approach with an example

Big Data (BD) Pipelines

- BD Pipelines are custom designed DP architectures built to accommodate one or more of the "V"s of big data!
- Considering Velocity trait - Appealing to build streaming data pipelines for big data. Then data can be captured and processed in real time
- Considering the Volume trait - BD pipelines must be scalable w.r.t time. In real-life scenario, there are likely to be many big data events that occur simultaneously or very close together (in time sense), so the big data pipeline must be able to scale to process significant volumes of data concurrently
- Considering the Variety trait - BD pipelines be able to recognize and process data in many different formats—structured, unstructured, and semi-structured

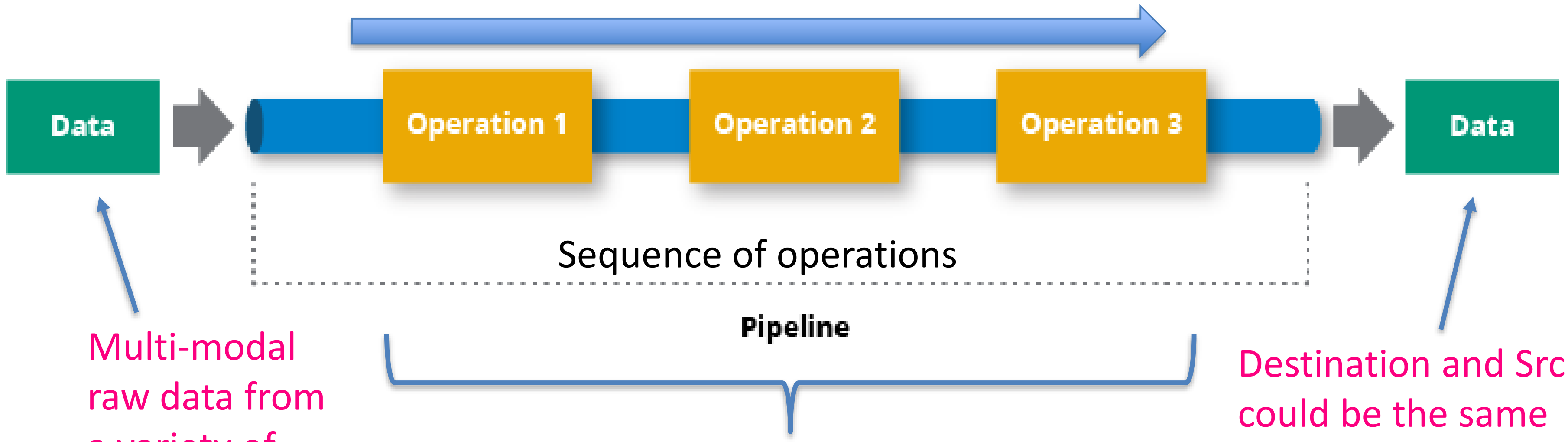
DP Architecture (*Biz plan*) considerations

- Factors that decide a data pipeline architecture – Following are the key questions you need to ask:

1. Do you need to handle streaming data?
2. What rate (velocity) of data do you expect?
3. How much and what types of processing you expect to happen in your DP?
4. Where is the data source? Is it being generated in the cloud or on-site (IoT/edge/fog devices) and where does it need to go?
5. Do you plan to build the pipeline with micro-services using, say, Amazon, Azure, etc?
6. What is your time horizon of handling the data and how much storage you expect to consume over time?

Types of DP Architectures

- Batch-based DP



Multi-modal raw data from a variety of sources

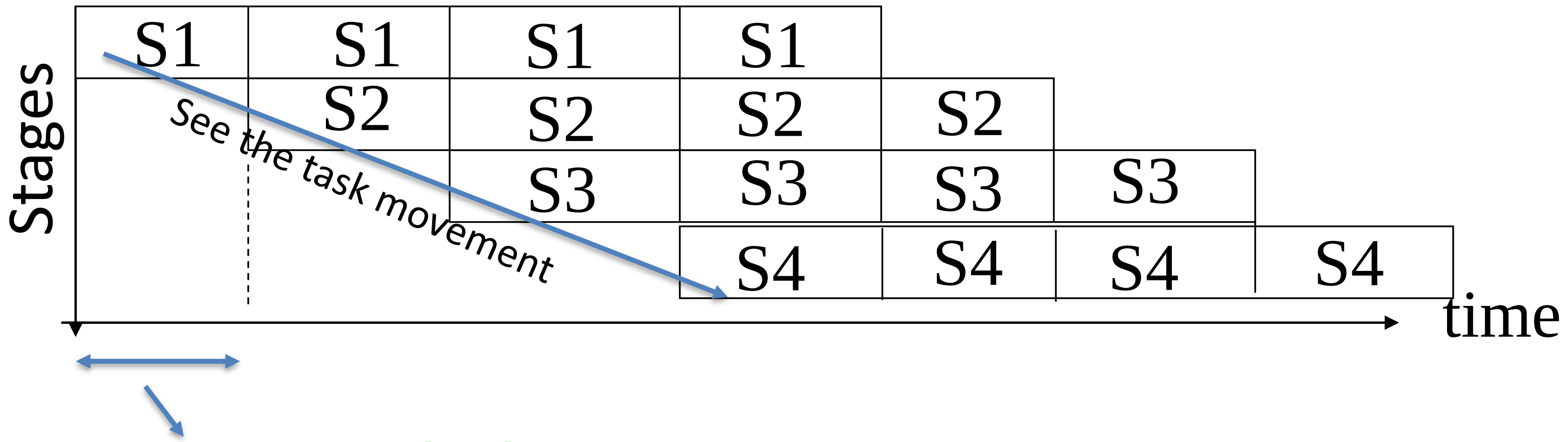
Destination and Src could be the same

Each operational stage can have concurrent processing steps, but all will feed to the next stage; So, output of a stage/operation serves as an input to the next stage;

Let us suppose we have N independent tasks to process on the whole in this pipeline with k stages/operations. Also, let us say, each stage works independently with its input from previous stage and each stage takes (a max of) T units of time to process its input. Then, the total time of processing all N independent tasks using this pipeline will be = ________

Data Pipeline – Timing diagram of events

Timing diagram for 4 stages and 3 tasks in an ideal BDP



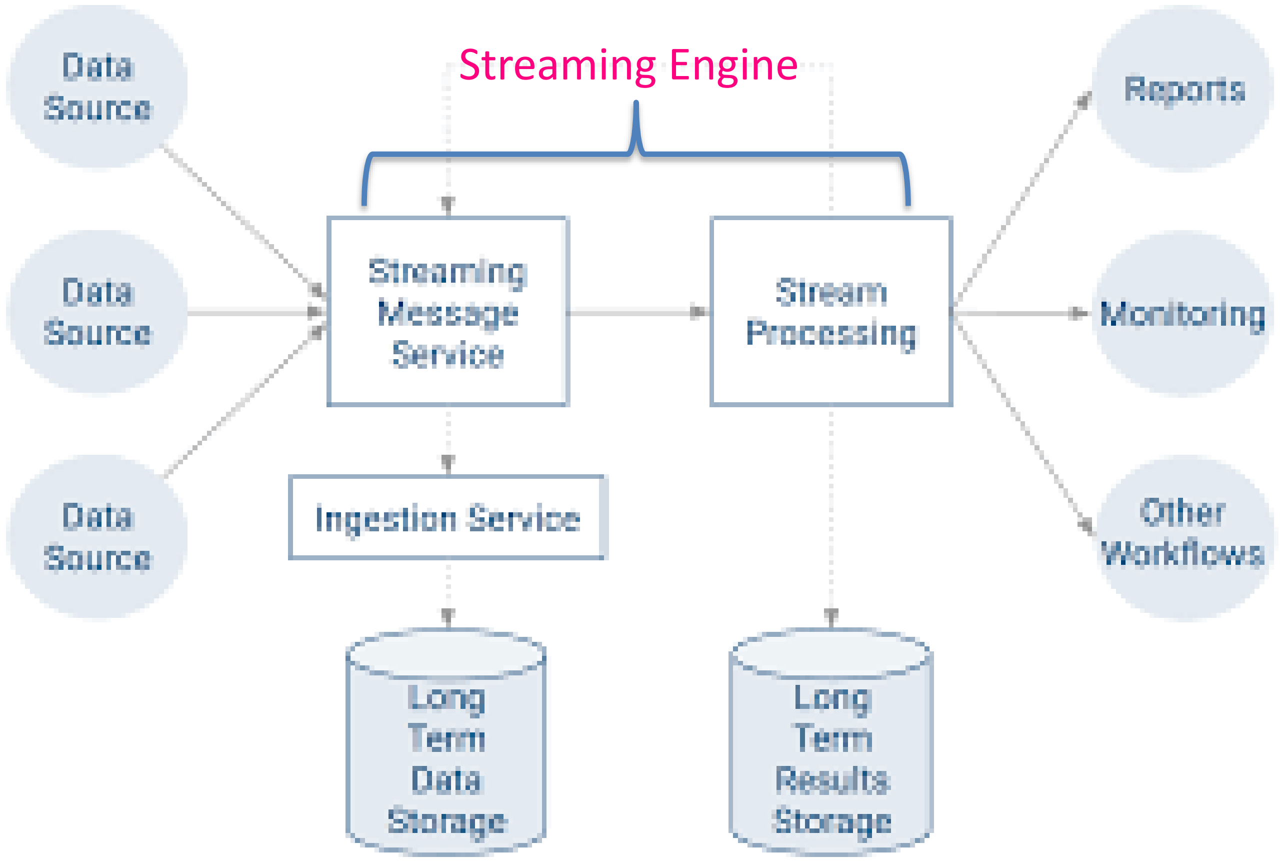
T: Time to process a single task in Stage 1

Q: So, for N tasks, N >> k, the total completion time will be ___

Q: What are the potential assumptions - problems/issues in the above style of working?

Types of DP Architectures

- Streaming-based DP



Apache Kafka is an example of a stream based messaging system

Types of DP Architectures

- Lambda Architecture

Combination of batch and streaming pipelines into one architecture.

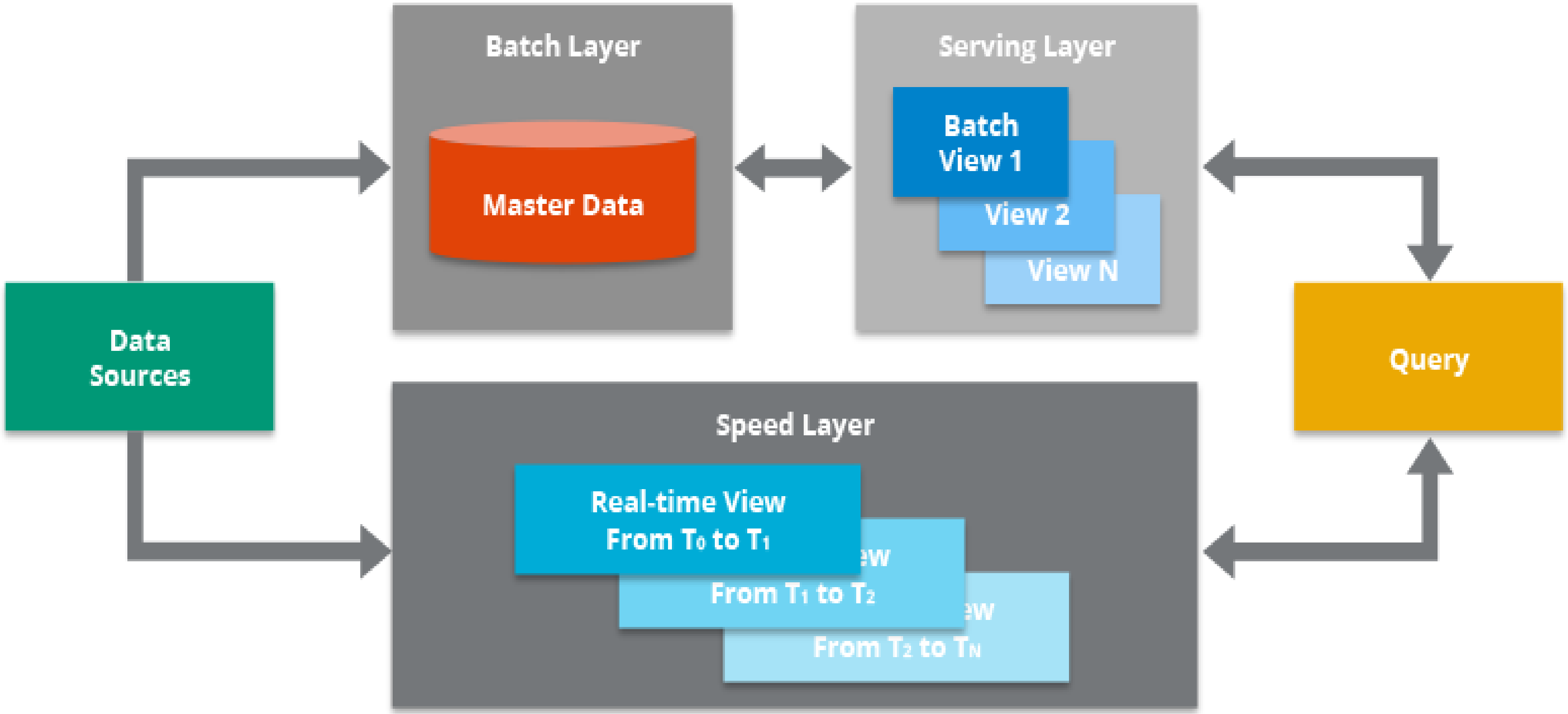
Popular in big data environments - it enables developers to account for both real-time streaming use cases and historical batch analysis.

Primary use -

(i) It encourages storing data in raw format so that we can continually run new data pipelines to correct any code errors in prior pipelines;

(ii) Facilitates creating new data destinations that enable new types of queries;

Lambda Architecture



Components of Lambda architecture

Batch layer

- To this layer, data comes from a data warehouse or a central data source
- Data is partitioned into small batches and sent for processing.
- Primary Component in this layer - **Hadoop** for this layer, uses **HDFS** for storing the data and ***MapReduce*** for batch processing
- Batch processing can be slow and can take up to several hours to complete; however, this layer is extremely reliable for processing large datasets and get detailed insights.
- **Use:** Once a batch is processed, several batch views can be created and pushed to Service layer for storage and querying

Components of Lambda architecture

Speed layer

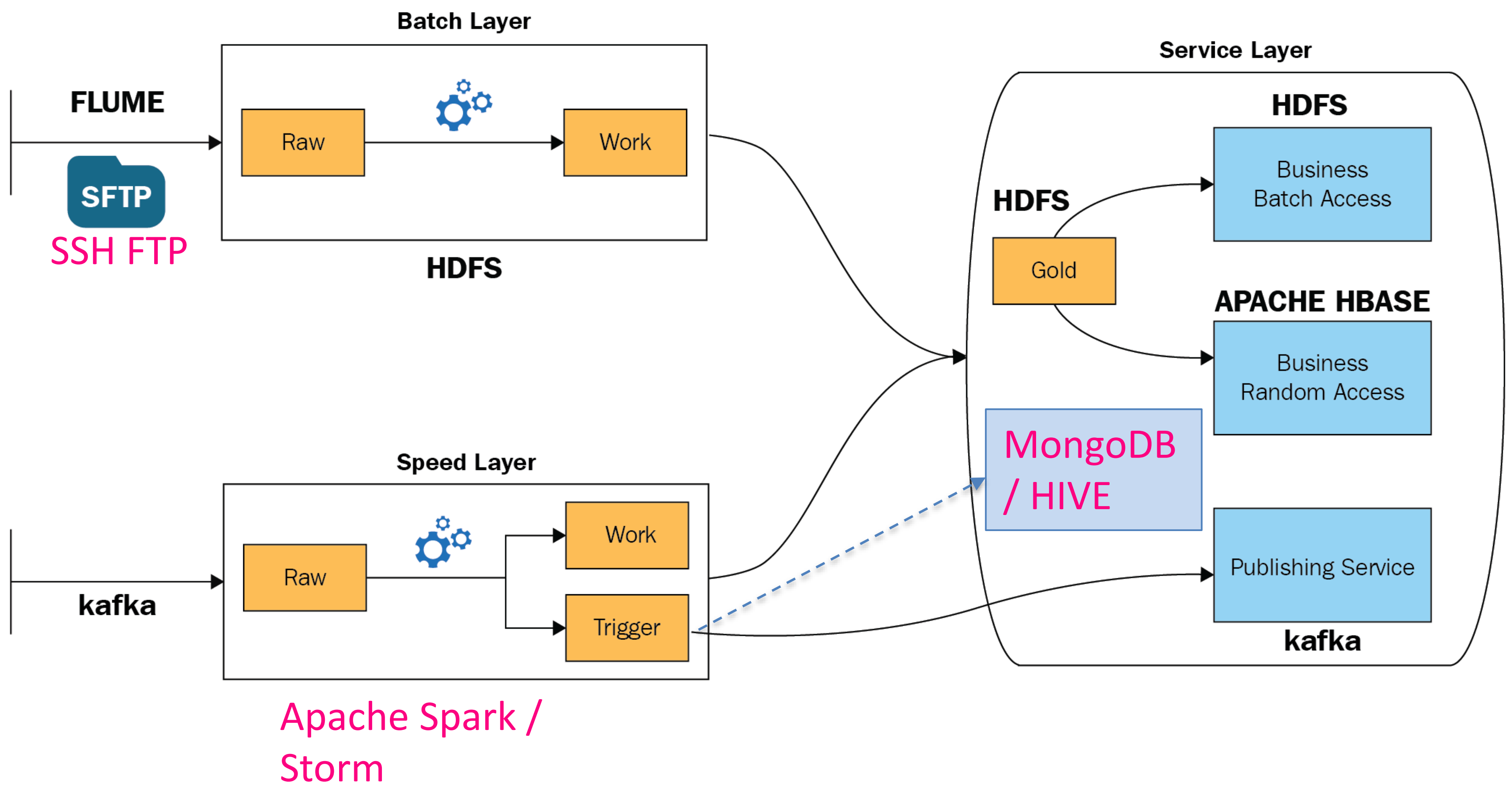
- Input to this layer is a continuous stream of data
- Technologies in use - *Apache Spark or Storm* for real-time and near real-time processing.
- The data that is pushed to this layer is quickly processed within milliseconds to seconds and the corresponding real-time views are pushed to the Service layer; If you need fast processing, this is an imperative layer
- This layer is able to perform real-time analytics

Components of Lambda architecture

Service layer

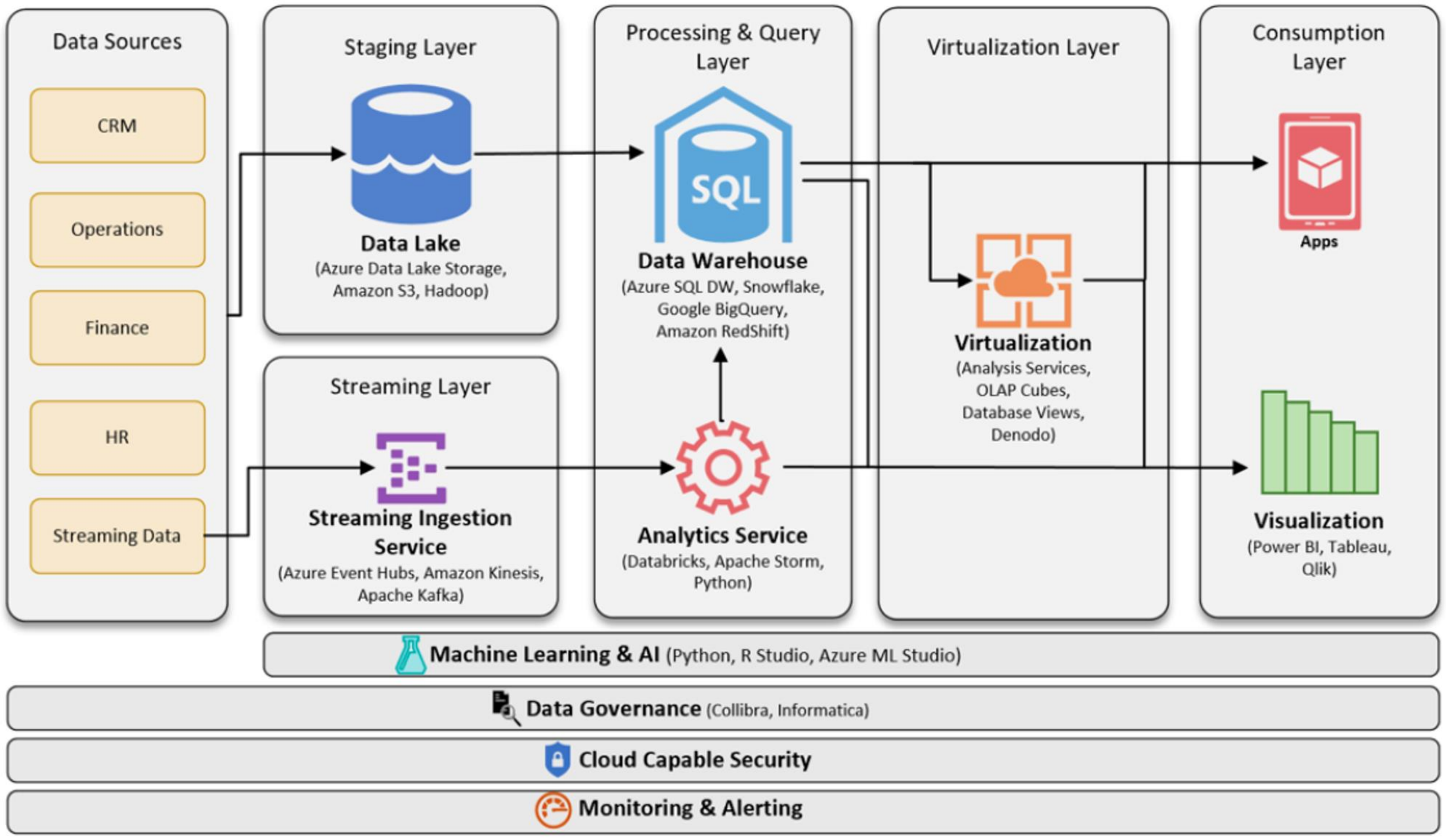
- Inputs to this layer is from both batch and speed layers;
- Primarily data are pushed into this layer for indexing purposes (depending on applic. - simple indices, based on data characs, etc)
- Technologies in use - Database technologies like Cassandra, HIVE, MongoDB can be used to store and manage the indexed data;
- **Use:** The data stored in this layer can be further queried by any in-house application for business/financial analytics

Lambda Architecture



Apache Flume – Distributed framework / software for data collection, aggregation and migration of large amounts of log data; Architecture is based on streaming data flows.

Lambda Architecture – Real-life implementation example


Data Sources
CRM
Operations
Finance
HR
Streaming Data
Staging Layer
Data Lake
(Azure Data Lake Storage, Amazon S3, Hadoop)
Streaming Layer
Streaming Ingestion Service
(Azure Event Hubs, Amazon Kinesis, Apache Kafka)
Processing & Query Layer
SQL
Data Warehouse
(Azure SQL DW, Snowflake, Google BigQuery, Amazon RedShift)
Analytics Service
(Databricks, Apache Storm, Python)
Virtualization Layer
Virtualization
(Analysis Services, OLAP Cubes, Database Views, Denodo)
Consumption Layer
Apps
Visualization
(Power BI, Tableau, Qlik)
Machine Learning & AI (Python, R Studio, Azure ML Studio)
Data Governance (Collibra, Informatica)
Cloud Capable Security
Monitoring & Alerting

DE + DS roles

Source: Credera

Lambda Architecture – *Pros & Cons*

Pros

- Trade-off between speed, reliability, and scalability
- Since it stores all the historical data, fault-tolerance with *rollback and checkpointing algorithms* guarantees a low probability of errors even if the system crashes.
- Access to both real-time and offline results in supporting several data analysis scenarios
- Since it stores all the historical data, optimizations may be more effective and perhaps even simpler to implement

Cons

- Although the offline layer and the real-time streams face different scenarios, their internal processing logic remains the same, so there are many duplicate modules and coding overhead!
- Reprocesses every batch cycle, which is not beneficial in specific scenarios.
- Migration of a dataset modeled with Lambda architecture to other platforms is not straightforward, if not difficult and to reorganize.

What does it lack currently? - User queries are required to be served on an ad-hoc basis using immutable data storage; Quick responses are required, and the system should handle various updates in new data streams; None of the stored records shall be erased, and it should allow the addition of updates and new data to the database.

(c) Bharadwaj V Dept of ECE NUS (2023)

Kappa Architecture

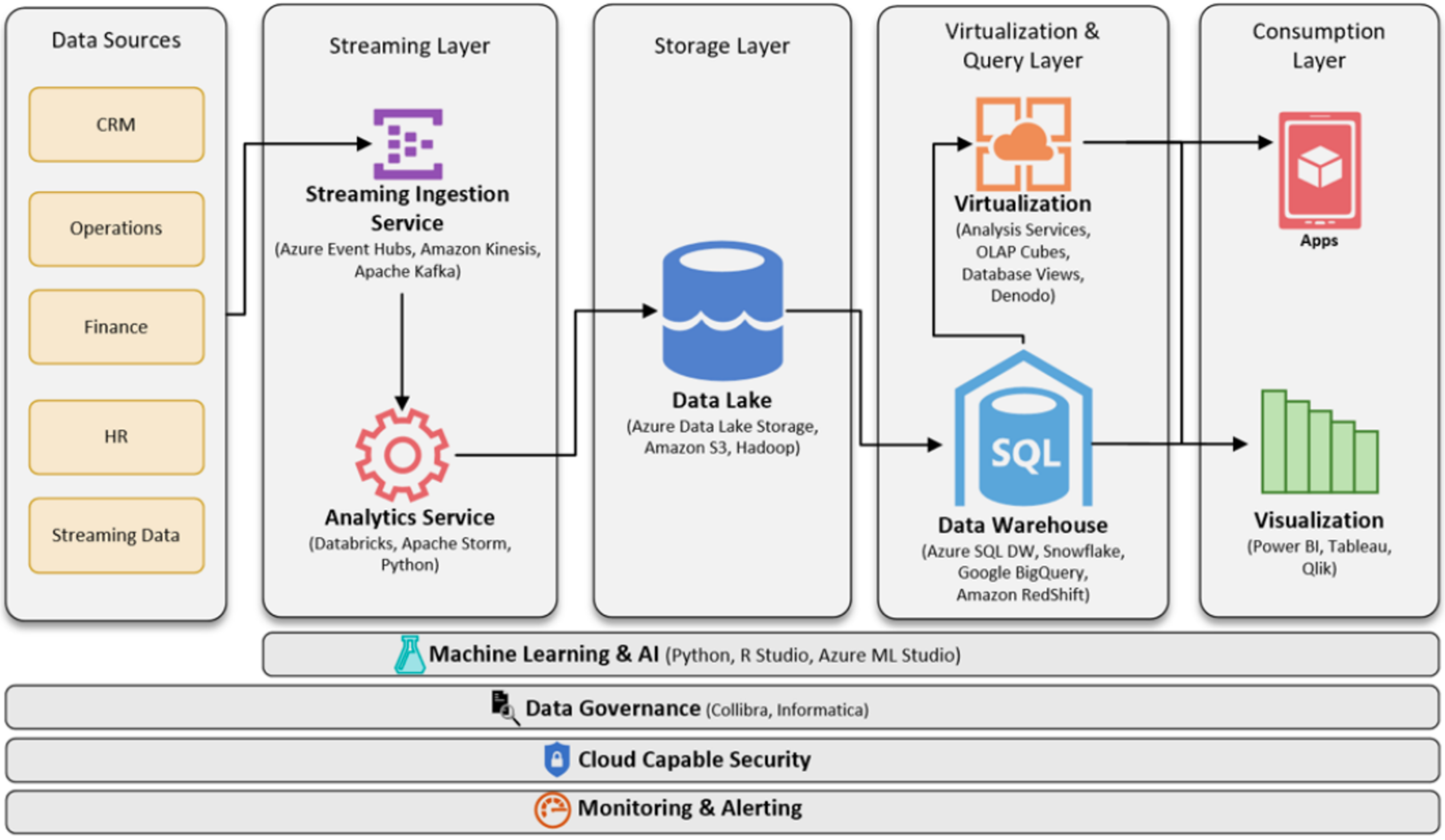
Kappa is designed with the idea of "replaying" data!

Replaying data is important to prevent data loss and a fault-tolerant system must often replay any stored data to any new tools that become part of your cloud eco-system;

Kappa avoids maintaining two different code bases for the batch and speed layers – Here, the central idea is to handle real-time data processing, and continuous data re-processing using a single stream processing engine and avoid a multi-layered approach as in Lambda architecture

Kappa is used where Batch processing is not necessary and mainly adopted for handling streaming data in real-time.

Kappa Architecture – Real-life implementation example


Data Sources
CRM
Operations
Finance
HR
Streaming Data
Streaming Layer
Streaming Ingestion Service
(Azure Event Hubs, Amazon Kinesis, Apache Kafka)
Analytics Service
(Databricks, Apache Storm, Python)
Storage Layer
Data Lake
(Azure Data Lake Storage, Amazon S3, Hadoop)
Virtualization & Query Layer
Virtualization
(Analysis Services, OLAP Cubes, Database Views, Denodo)
SQL
Data Warehouse
(Azure SQL DW, Snowflake, Google BigQuery, Amazon RedShift)
Consumption Layer
Apps
Visualization
(Power BI, Tableau, Qlik)
Machine Learning & AI (Python, R Studio, Azure ML Studio)
Data Governance (Collibra, Informatica)
Cloud Capable Security
Monitoring & Alerting

DE + DS roles

Source: Credera

Kappa Architecture – *Pros & Cons*

Pros

- Applications can have direct access for reading and writing to Kafka (or other message queues) as developed;
- For existing event sources, stream writes can be done directly from database logs (or other datastore equivalents), eliminating the need for batch processing during ingress, resulting in fewer resources;
- Treating every data point in your organization as a streaming event also provides you the ability to 'time travel' to any point and see the state of all data in an organization

Cons

- The complication of this architecture mainly revolves around having to process this data in a stream operations that are generally easier to do in batch processing;
- Although Kappa architecture looks concise, it is not easy to implement, especially for the data replay;
- Unlike Lambda arch., Kappa does not have catalog services to auto-discover and document file and database systems;

What does it lack currently? - It is very beneficial to use the same code base to process historical and real-time data; Since this is a stream processing platform it can be allowed to interact with the database at any time;

(c) Bharadwaj V Dept of ECE NUS (2023)

Final remarks - Which architecture to choose?

Three compelling points to remember!

Kappa should not be viewed as a replacement for Lambda since some use-cases deployed using the Lambda architecture cannot be migrated;

If you are looking for an architecture that is more reliable in updating the *data lake* as well as efficient in training the ML/DL models to predict upcoming events, then we must use Lambda architecture as it fully takes advantage of both the batch layer and speed layer to ensure minimum errors without compromising on speed.

On the other hand, if you want to deploy big data architecture using less expensive hardware and require it to deal effectively with unique events occurring continuously (in time sense), then we must adopt the Kappa architecture for our real-time data processing needs.

(c) Bharadwaj V Dept of ECE NUS (2023)

Design of a Data Pipeline – Detailed approach

- What are the steps to follow in realizing a working prototype / design?

1) Identify the application – First convince yourself that it needs a DP!
2) Identify what type of DP category we are looking for – batch type, stream type or hybrid
3) Understand the working mechanism of your application to the level of finer details
4) *Devil is in the details!* What are the parameters/variables that you need to track and put in your design?

Design of a DP (Cont’d)

- Application – Need to extract information about the visitors to your start-up company’s web-page!

Why do companies need this info?

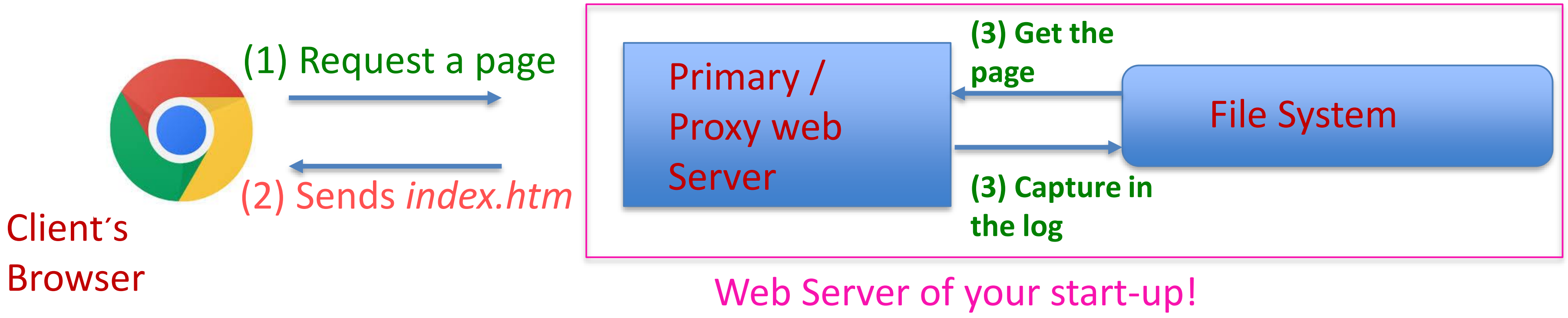
-- How many users from each country visit your site each day?

-- What countries to focus your marketing efforts on?

- At the simplest level, *it gives them a measure of your marketing efforts! Plus, where you need to put additional marketing focus.*

Design of a DP (Cont'd)

- Understanding your application



Step 1 - Client sends a request to the web server asking for a certain page.
Step 2 - Web server then loads the page from the filesystem and returns it to the client
Step 3 - Web server concurrently writes a line to a log file on the filesystem that contains some metadata about the client and the request.

Companies use this log later to see who all visited which pages on the website at what time, and perform analytics!

Conclude - So, we can clearly see that this is a time continuous activity! This means we are about to handle "streaming data"!

Design of a DP (Cont’d)

- Format of the log:

*$remote_addr - $remote_user [$time_local] "$request"* ***$status $body_bytes_sent*** *"$http_referer" "$http_user_agent“*

Sample log: (may be in a text file or a SQL table entry)

X.X.X.X - - [09/Apr/2020:01:15:59 +0000] "GET /blog/assets/css/jupyter.css HTTP/1.1" ***500 40127*** *"https://q.p.h.t/elebv/mycompany/"Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/700.21 Ubuntu Chromium/53.0.2785.143 Chrome/p.q.r.s Safari/700.21 university/1.0 http://www.ece.cnl.nus.edu.sg/)"*

Design of a DP (Cont'd)

- Understanding the format of the log:

*$remote_addr - $remote_user [$time_local] "$request"* ***$status $body_bytes_sent*** *"$http_referer" "$http_user_agent"*

$remote_addr — IP address X.X.X.X of the client making the request to the server
$remote_user — after authentication, this is the user name. if none, left blank;

$time_local — Local time when the request was made;

$request — Type of request, and the URL that it was made to ("GET …")

$status — Response status code from the server (500 in the first line)

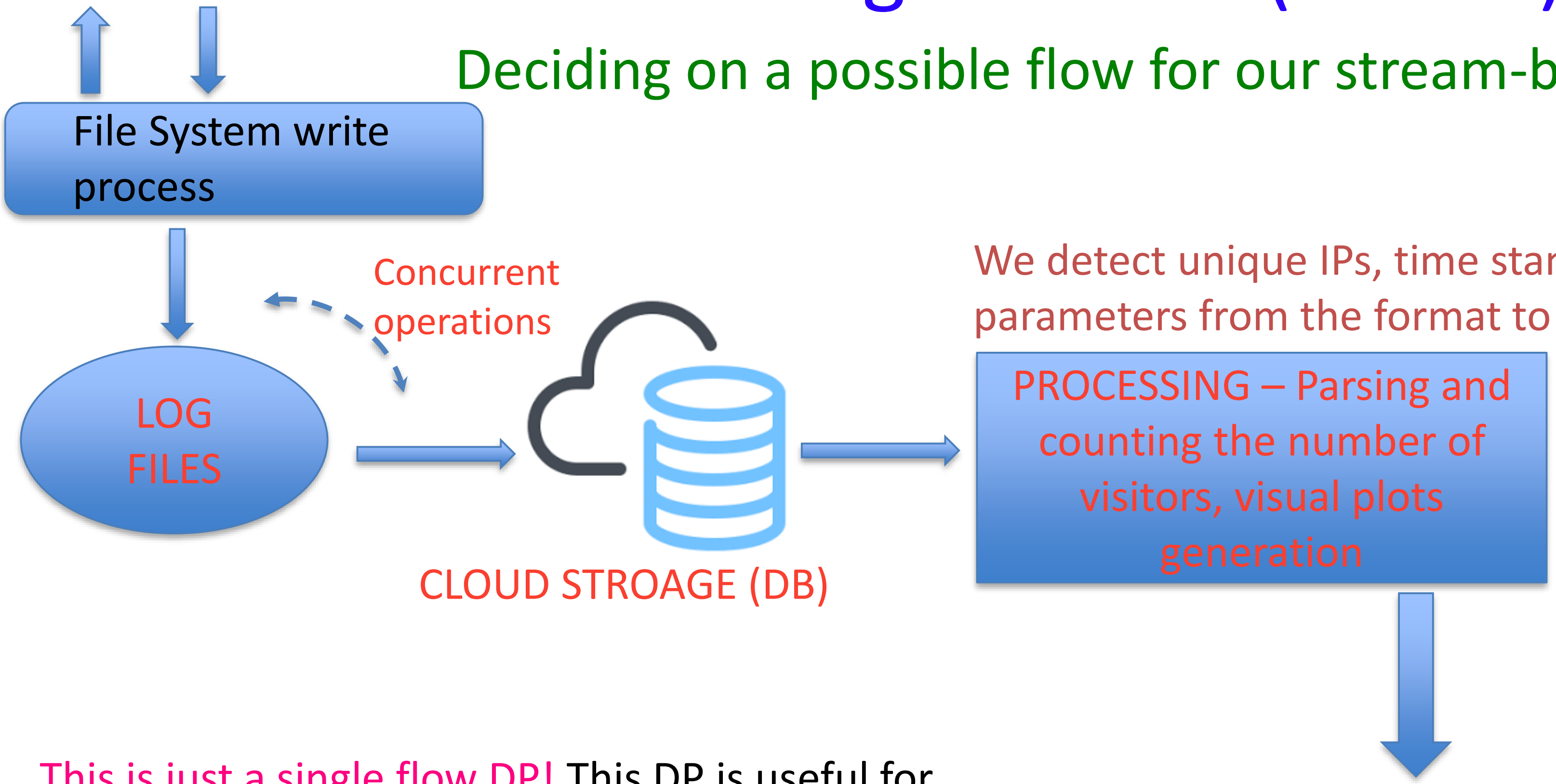
$body_bytes_sent — Number of bytes sent by the server to the client in the response body (40127)

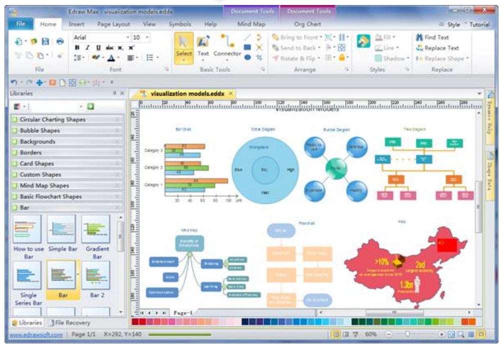
$http_referrer — Page that the client was on before sending the current request. https:// https://q.p.h.t/elebv/mycompany/

$http_user_agent — Information about the browser and system of the client. "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/700.21 Ubuntu Chromium/53.0.2785.143 Chrome/p.q.r.s Safari/700.21 university/1.0 http://www.ece.cnl.nus.edu.sg/)

Design of a DP (Cont'd)

Deciding on a possible flow for our stream-based DP!


File System write process
LOG FILES
Concurrent operations
CLOUD STROAGE (DB)
We detect unique IPs, time stamps, parameters from the format to do this
PROCESSING – Parsing and counting the number of visitors, visual plots generation


DASHBOARD

This is just a single flow DP! This DP is useful for marketing team.

Any extensions possible using the same data we have collected? *What about an engineering team who wants to know the type of browser from which the requests are sent and that they want to know the most popular browser users are using*

Design of a DP (Cont'd)

Deciding on two possible flows for our stream-based DP!

File System write process

LOG FILES

Concurrent operations

CLOUD STROAGE (DB)

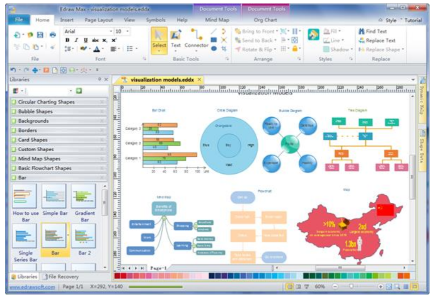
We detect unique IPs, time stamps, parameters from the format to do this

PROCESSING – Parsing and counting the number of visitors, visual plots generation

PROCESSING – Parsing and counting the number & type of browsers, visual plots generation

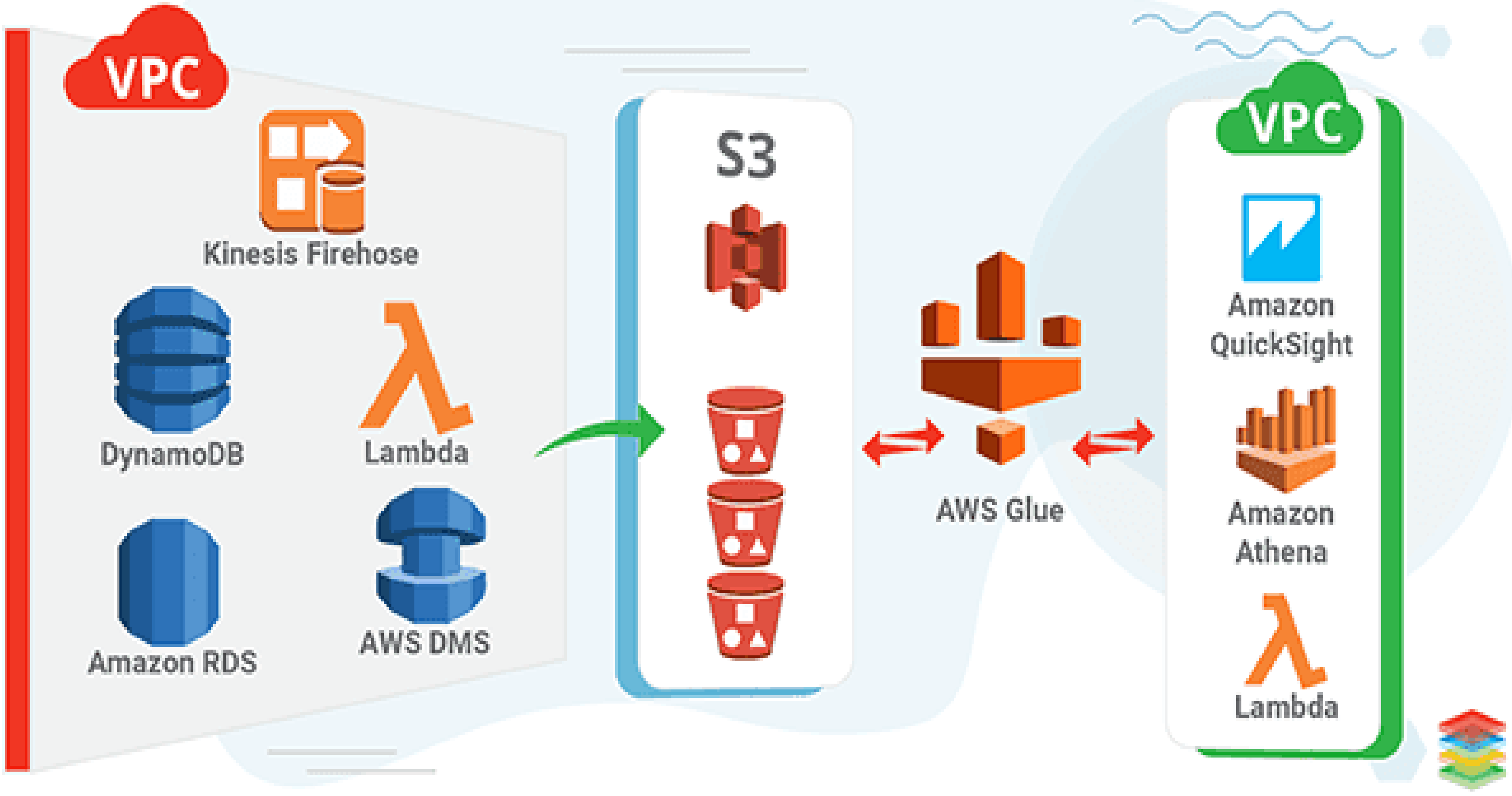
We use http_user_agent parameter from the format to do this

DASHBOARD Engg. Team



DASHBOARD Mkt'ing Team

Big Data Pipeline on AWS



Ref: AWS

Understanding the Big Data DP Components

On the input side of the DP

Kinesis Firehouse - Service that serves as a tool for the ingestion of streaming data from various data sources to the data sinks in a secured way; Able to handle a large amount of data stream workloads and scale accordingly

DynamoDB - Fully managed NoSQL database with fast/high performance

Lambda - We can also define lambda function in case we want to perform any data transformation; Firehose comes with pre-configured AWS Lambda blueprints and templates that make it even easy to implement it;

Amazon RDS - Fully managed relational database that can accommodate large dataset;. Has numerous options for the database you want, e.g., AWS aurora, Postgres, Mssql, MariaDB.

AWS DMS - Amazon's data migration service component

Understanding the BD DP Components

In the middle layer of the DP

Amazon S3 - Low-cost highly-scalable object storage; Different types of instances possible; Users may choose on their needs and budget constraints;

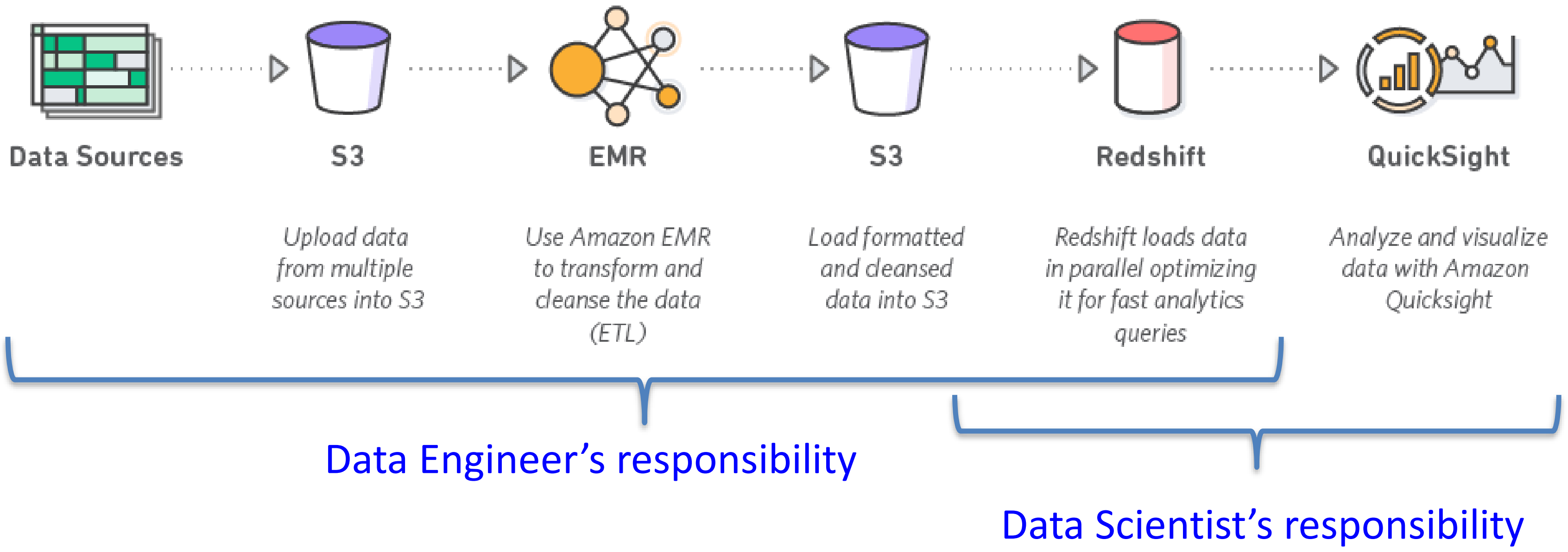
AWS Glue - AWS Glue is a serverless **ETL job service**. While using this, a data engineer need not worry about setting up and managing the underlying infrastructure for running the ETL job;

In the terminal layer of the DP

Amazon QuickSight – Biz intelligence tool best known for stunning visualizations, interactive dashboards, and accurate machine learning insights;

Amazon Athena – A query engine that runs over the data stored in S3; Redshift is another option; but if you don't have Redshift, Athena does the job for us; charges you per query basis!

A complete work-flow (Example: Amazon framework)


Data Sources
S3
EMR
S3
Redshift
QuickSight
Upload data from multiple sources into S3
Use Amazon EMR to transform and cleanse the data (ETL)
Load formatted and cleansed data into S3
Redshift loads data in parallel optimizing it for fast analytics queries
Analyze and visualize data with Amazon Quicksight
Data Engineer's responsibility
Data Scientist's responsibility

In Chapter 4, we will see certain supportive technologies/platforms to handle and perform DE tasks!

Thank You!