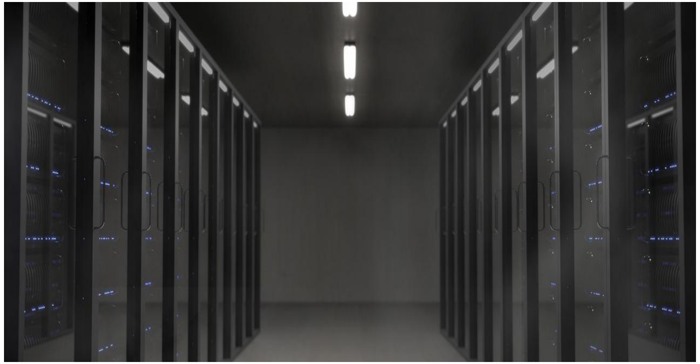
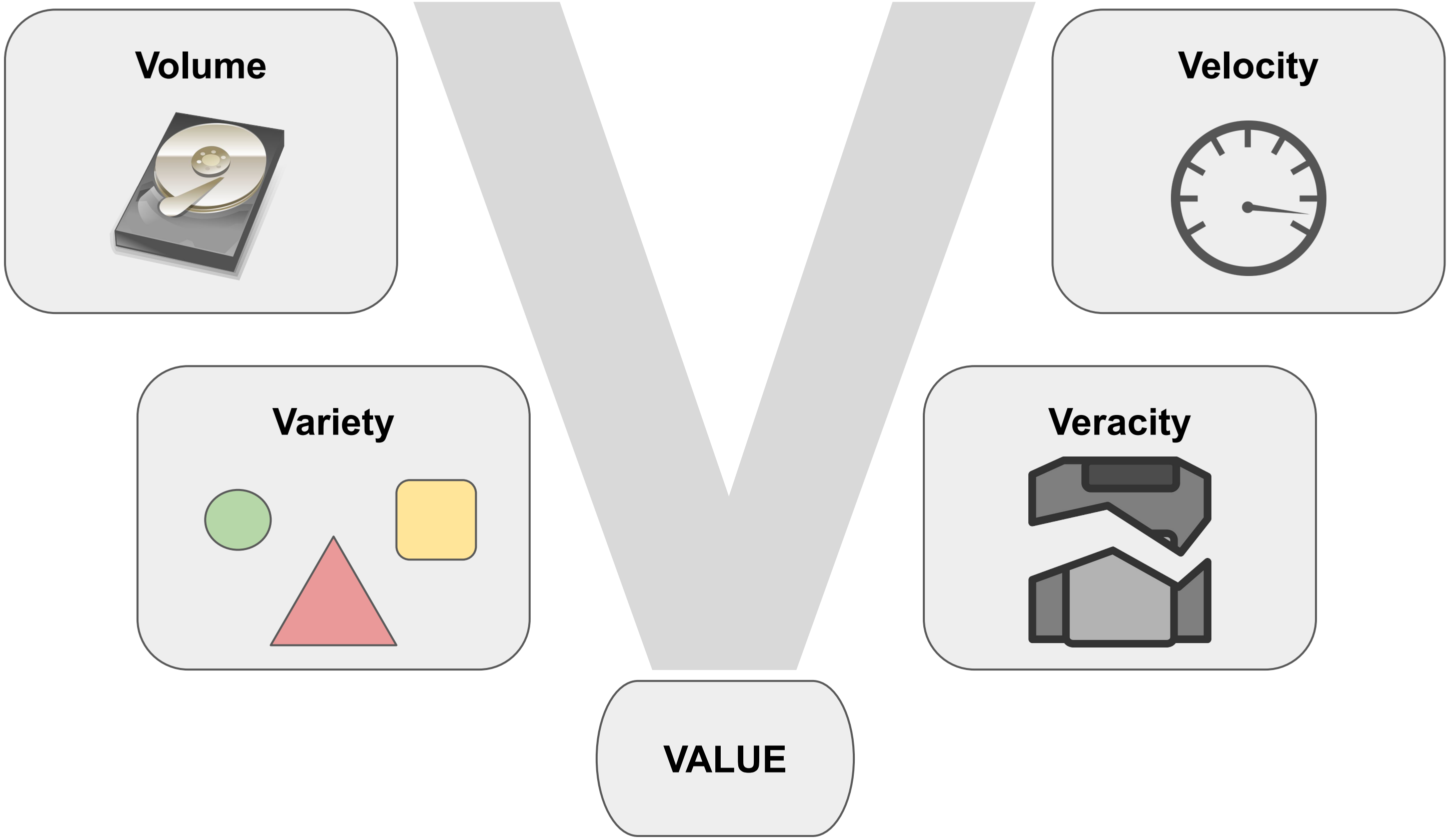
Big Data



Big Data



Volume

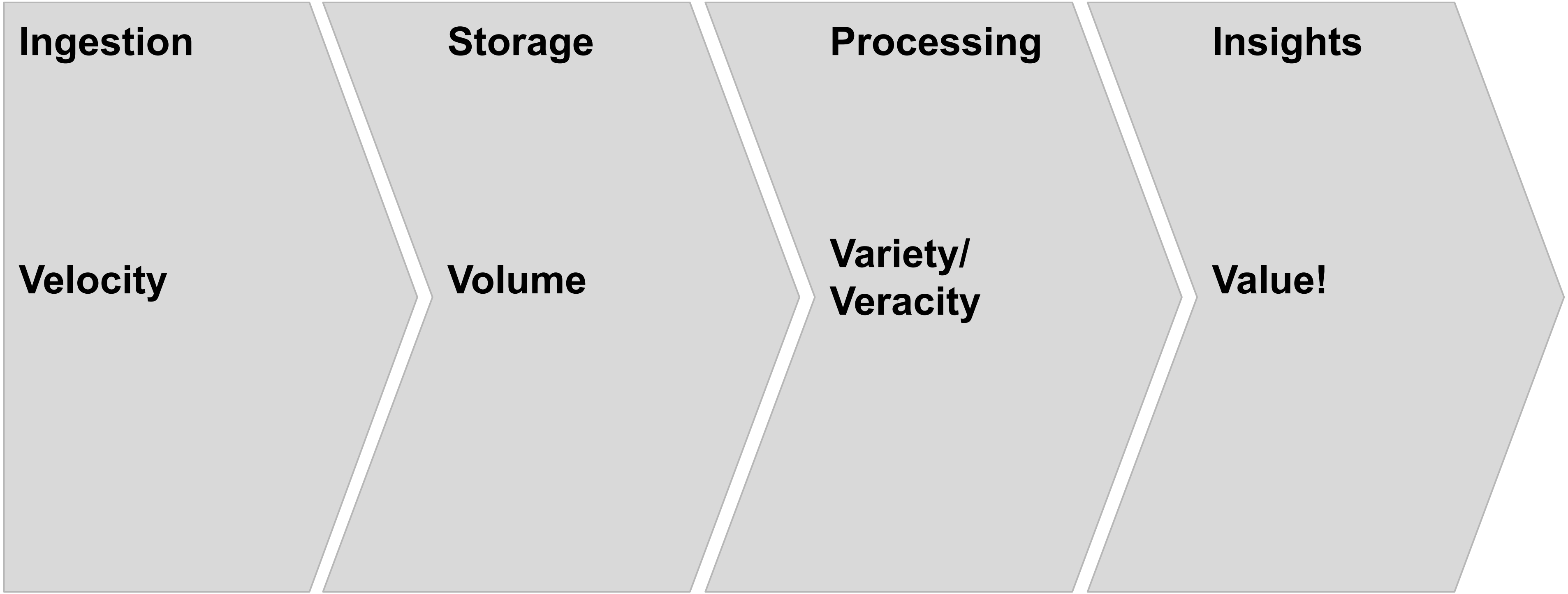
Velocity

Variety

Veracity

VALUE

Big Data Flow



Ingestion

Velocity

Storage

Volume

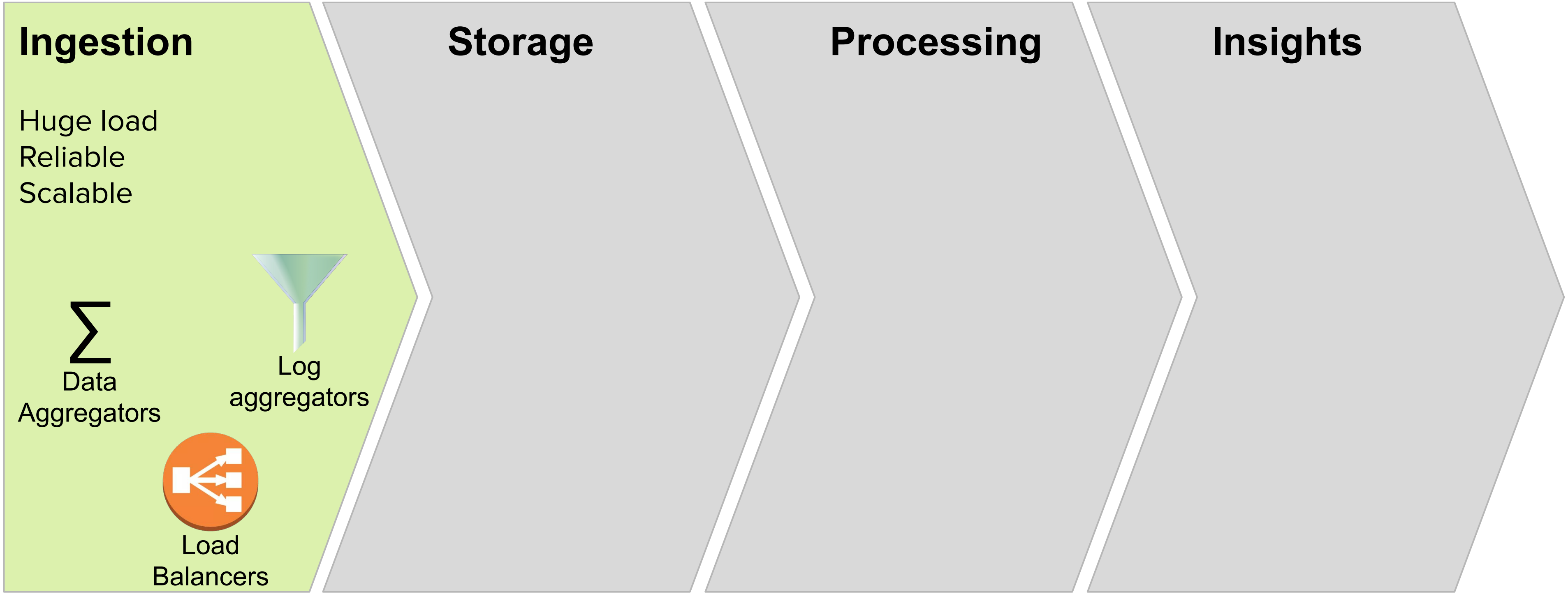
Processing

Variety/ Veracity

Insights

Value!

Big Data Flow



Ingestion

Huge load
Reliable
Scalable

∑
Data Aggregators

Log aggregators

Load Balancers

Storage

Processing

Insights

Data Aggregators

- Listens to events from servers and:
  - Aggregates values for counters
  - Generates stats for timers (max, min, mean, std_dev, percentiles)
  - Maintains sets of values
- Memory efficient (stores only counters/variables to calculate stats)
- Instead of sending millions requests to database, only aggregated statistics are sent

Load balancers

- Distribute load across multiple servers
- Add/Remove servers as load increases/decreases
- No need for over-provisioning

Log Aggregators

- Front layer before actual data storage
- Dedicated to process and stream logs to destination
- Scalable to cope with the load
- Some systems enable real-time analytics on the stream of incoming data

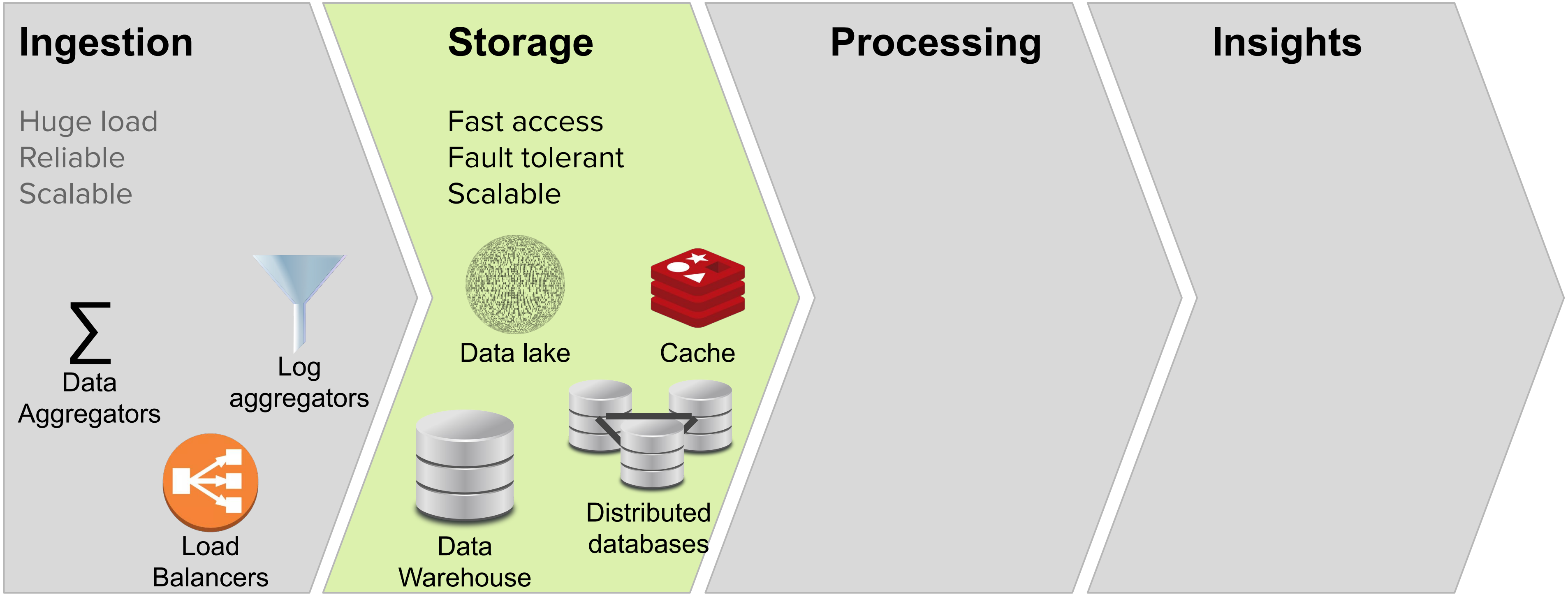
Probabilistic Data Structures

- They do not provide definite answer
  - Can return false positive or false negative
  - But will be firm on True Positives or True Negatives
- Used to avoid costly checks
- Example: Bloom filter
  - Checks if an item is in a set
  - "possibly in set" or "definitely not in set"
  - Chrome Browser - checks locally if url could be malicious, if yes then perform second online check that will detect false positives
  - Some databases checks if rows exist
  - Caching servers use it to prevent caching one-hit websites

ETL / ELT

- Extract
    - pull data from source (other databases, data lake, websites, etc.)
- Transform
    - clean, filter, map, join, apply business rules
- Load
    - load data to destination data storage
- With new, scalable, cloud-based databases a valid choice is often to load “raw” data and then do transformations using SQL
    - This process is often still referred as ELT

Big Data Flow



Ingestion

Huge load
Reliable
Scalable

Data Aggregators

Log aggregators

Load Balancers

Storage

Fast access
Fault tolerant
Scalable

Data lake

Cache

Data Warehouse

Distributed databases

Processing

Insights

Data lake

- Vast storage of raw data
- Data usually stored in files
  - Multiple rows/units of data in one file
  - Some basic grouping (by log type, date, etc.)
- Excellent to dump all incoming data
- Should be further processed to be useful

- Still should have some design in mind not to turn into data swamp

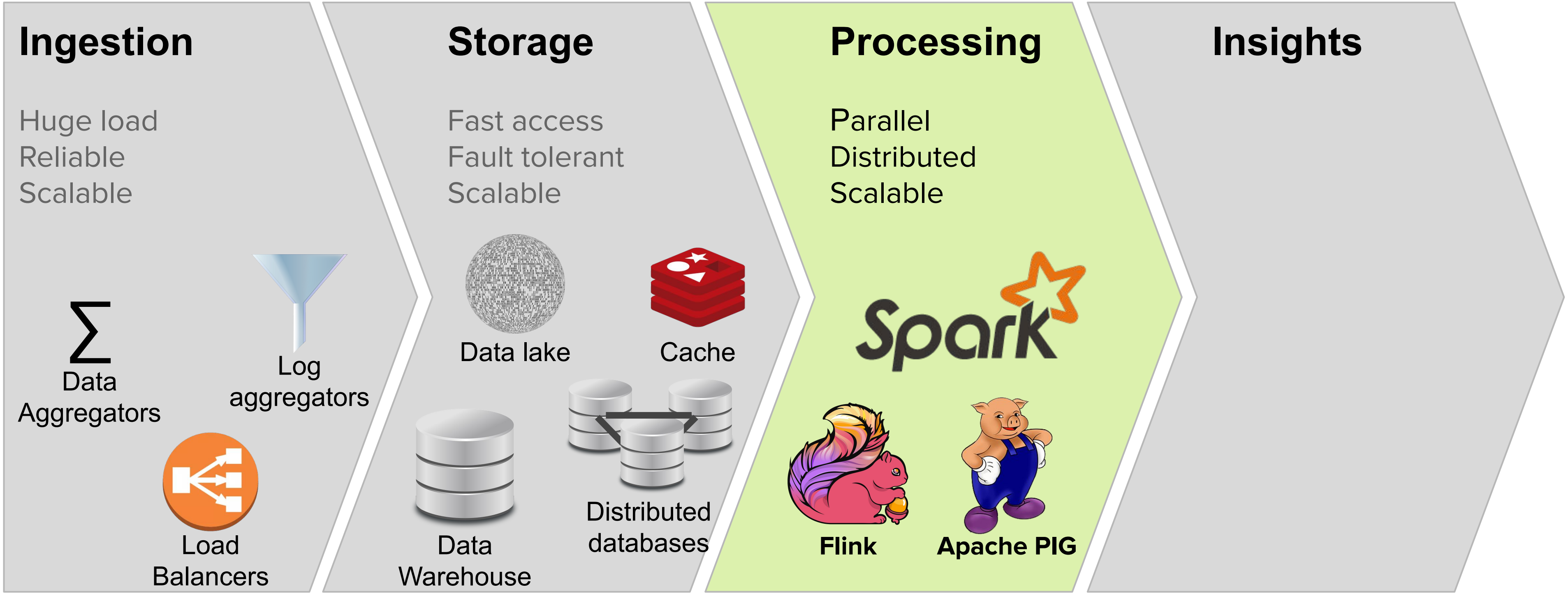
HDFS

- Hadoop Distributed File System
- A file system to store files across cluster
  - Divides files into chunks
  - Stores chunks at different nodes
  - Duplicates chunks (to prevent data loss in case a node goes down)
- Enables to store large amount of data
  - Low possibility of data loss
  - Fast access

No-SQL

- Databases able to store unstructured data
- SQL is still important and have multum use cases
  - NO should corresponds rather to Not Only
- Popular for many big data applications
  - Simple design
  - Easier to scale horizontally (by adding more servers vs making server bigger)
- Different types of storage
  - Key-value storage
  - Wide column (each row may have different columns)
  - Graph
  - Document

Big Data Flow



Ingestion

Huge load
Reliable
Scalable

Data Aggregators

Log aggregators

Load Balancers

Storage

Fast access
Fault tolerant
Scalable

Data lake

Cache

Data Warehouse

Distributed databases

Processing

Parallel
Distributed
Scalable

Spark

Flink

Apache PIG

Insights

Hadoop

- A collection of software packages to facilitate orchestration of network of computers
- Hadoop cluster consist of multiple nodes (one being a master node)
- Components:
  - HDFS - distributed file system
  - MapReduce - programming model for handling large amount of data
  - YARN - Yet Another Resource Negotiator - cluster management
- Thanks for distributed files jobs can run in parallel working on data that is on a given node

Aggregating data (group - map - reduce)

Spark/PIG program loads logs:

```
{“store_id”:1, “date”:”2016-10-31 00:00:00”, “user_id”:”x1”, “entry_page”:”home”, … }
{“store_id”:2, “date”:”2016-10-31 00:00:01”, “user_id”:”x2”, “entry_page”:”product”, … }
…
{“store_id”:1, “date”:”2016-10-31 00:15:31”, “user_id”:”x1”, “revenue”:500.00, … }
…
```

Use **`store_id`**, **`date`**, and **`user_id`** to group and maps logs into sessions:

```
{“store_id”:1, “session_start_date”:”2016-10-31 00:00:00”, “user_id”:”x1”,
“entry_page”:”home”, “revenue”:500.00, … }
{“store_id”:2, “session_start_date”:”2016-10-31 00:00:01”, “user_id”:”x2”,
“entry_page”:”product”, “revenue”:null, … }
…
```

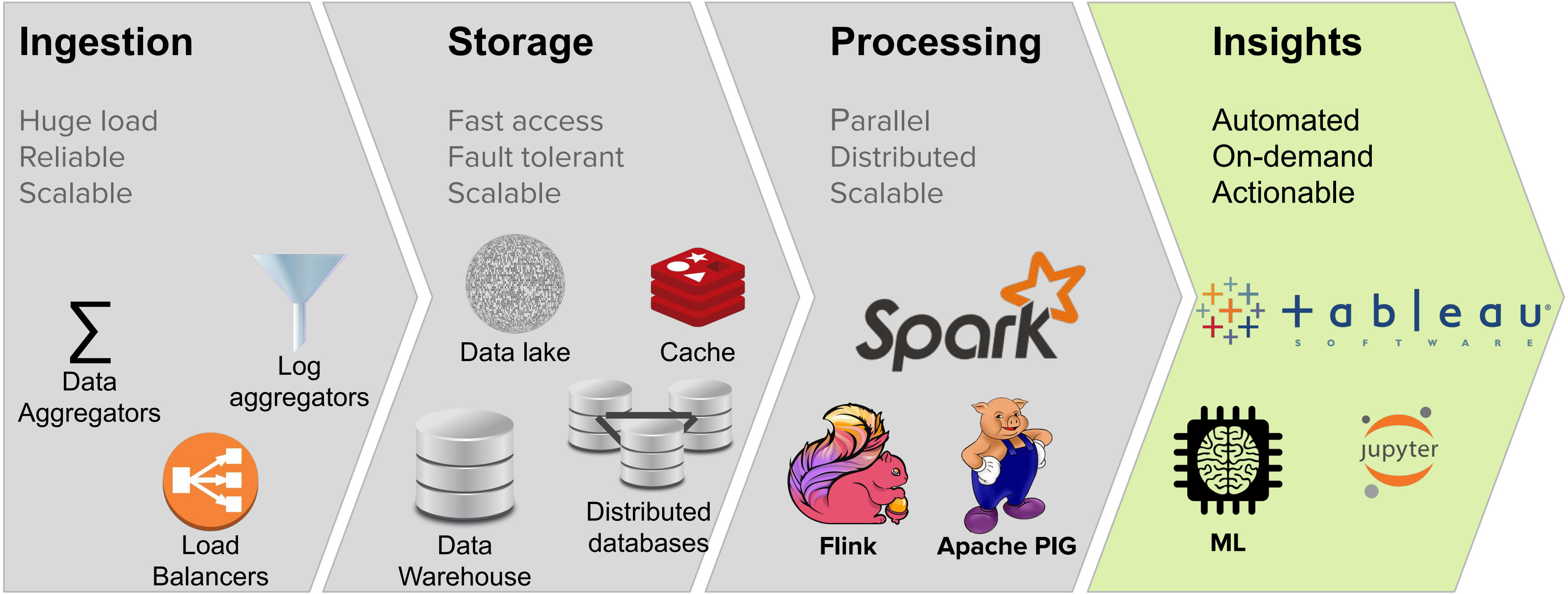
Aggregating data (group - map - reduce)

Aggregate sessions using **`store_id`**, day(**`session_start_date`**), **`entry_page`**, and other relevant fields*:

```
{“store_id”:1, “date”:”2016-10-31”, “entry_page”:”home”, “num_sessions”: 154304,
“num_transaction”:11446, “revenue”:1022770.23, … }
{“store_id”:1, “date”:”2016-10-31”, “entry_page”:”product”, “num_sessions”: 93070,
“num_transaction”:6327, “revenue”:520129.45, … }
…
{“store_id”:2, “date”:”2016-10-31”, “entry_page”:”home”, “num_sessions”: 127829,
“num_transaction”:2243, “revenue”:452317.34, … }
…
```

* Some fields may need special processing (e.g. translating **`user_agent`** string to human readable form)

Big Data Flow



Ingestion

Huge load
Reliable
Scalable

Data Aggregators

Log aggregators

Load Balancers

Storage

Fast access
Fault tolerant
Scalable

Data lake

Cache

Data Warehouse

Distributed databases

Processing

Parallel
Distributed
Scalable

Flink

Apache PIG

Insights

Automated
On-demand
Actionable

ML