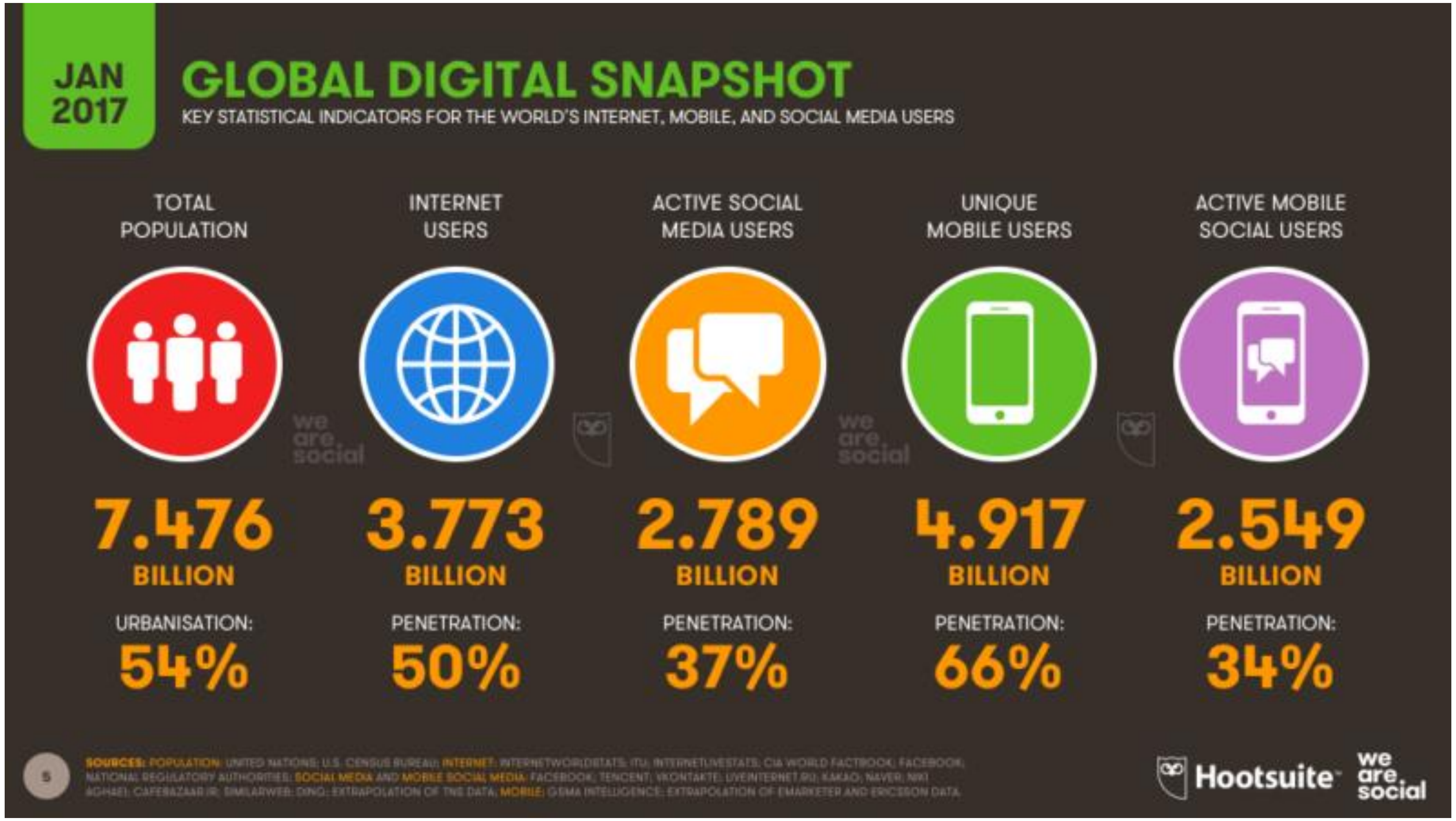
Introduction to Data Systems & Data Architectures

Bachar Wehbi <me@bachwehbi.net>

We live in a global digital age



JAN 2017

GLOBAL DIGITAL SNAPSHOT

KEY STATISTICAL INDICATORS FOR THE WORLD'S INTERNET, MOBILE, AND SOCIAL MEDIA USERS

TOTAL POPULATION	INTERNET USERS	ACTIVE SOCIAL MEDIA USERS	UNIQUE MOBILE USERS	ACTIVE MOBILE SOCIAL USERS
7.476 BILLION	3.773 BILLION	2.789 BILLION	4.917 BILLION	2.549 BILLION
URBANISATION: 54%	PENETRATION: 50%	PENETRATION: 37%	PENETRATION: 66%	PENETRATION: 34%

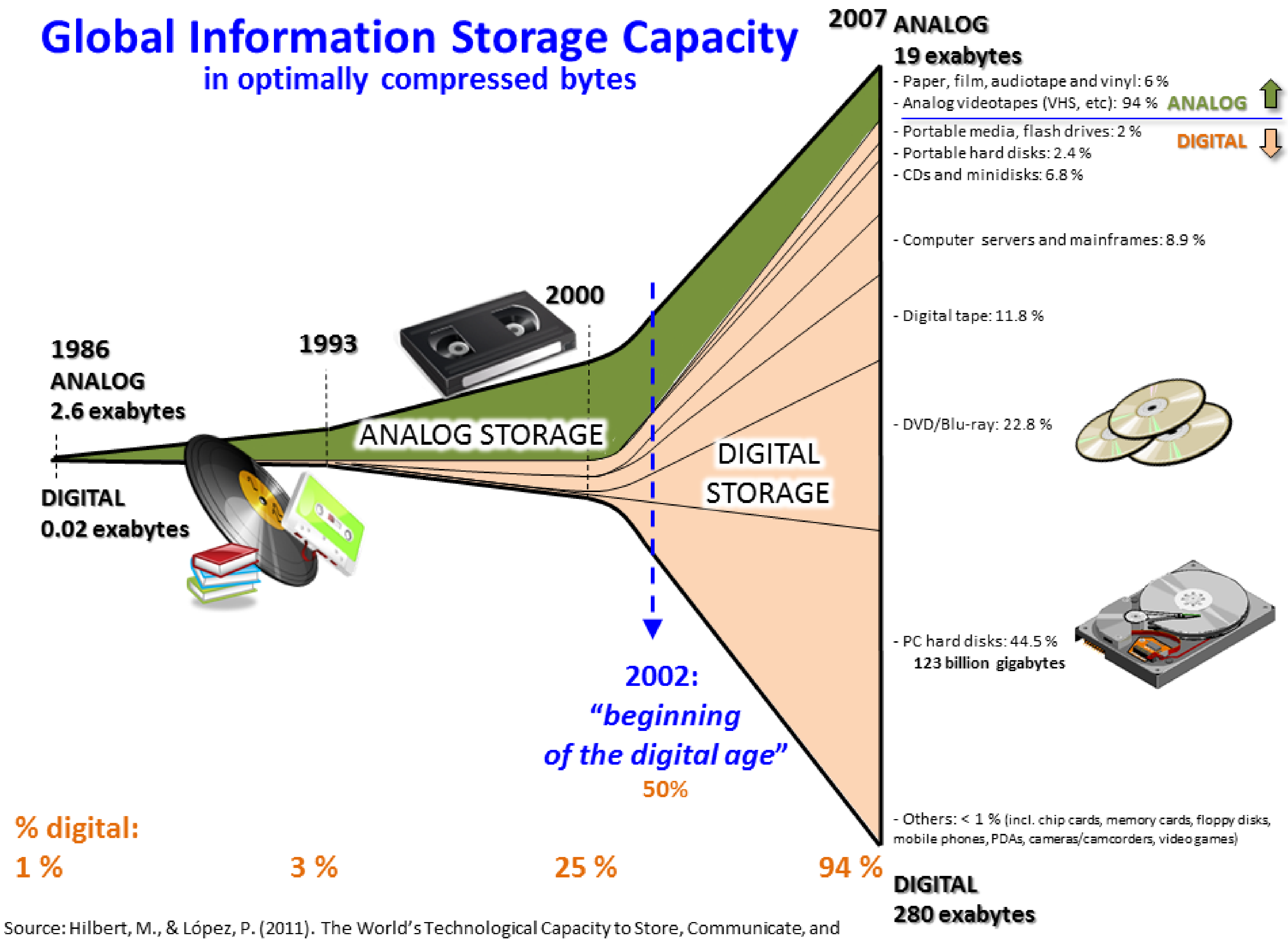
5

SOURCES: POPULATION: UNITED NATIONS; U.S. CENSUS BUREAU; INTERNET: INTERNETWORLDSTATS; ITU; INTERNETLIVESTATS; CIA WORLD FACTBOOK; FACEBOOK; NATIONAL REGULATORY AUTHORITIES; SOCIAL MEDIA AND MOBILE SOCIAL MEDIA: FACEBOOK; TENCENT; VKONTAKTE; LIVEINTERNET.RU; KAKAO; NAVER; NIKI AGHAEI; CAFEBAZAAR.IR; SIMILARWEB; DING; EXTRAPOLATION OF TNS DATA; MOBILE: GSMA INTELLIGENCE; EXTRAPOLATION OF EMARKETER AND ERICSSON DATA.

Hootsuite we are social

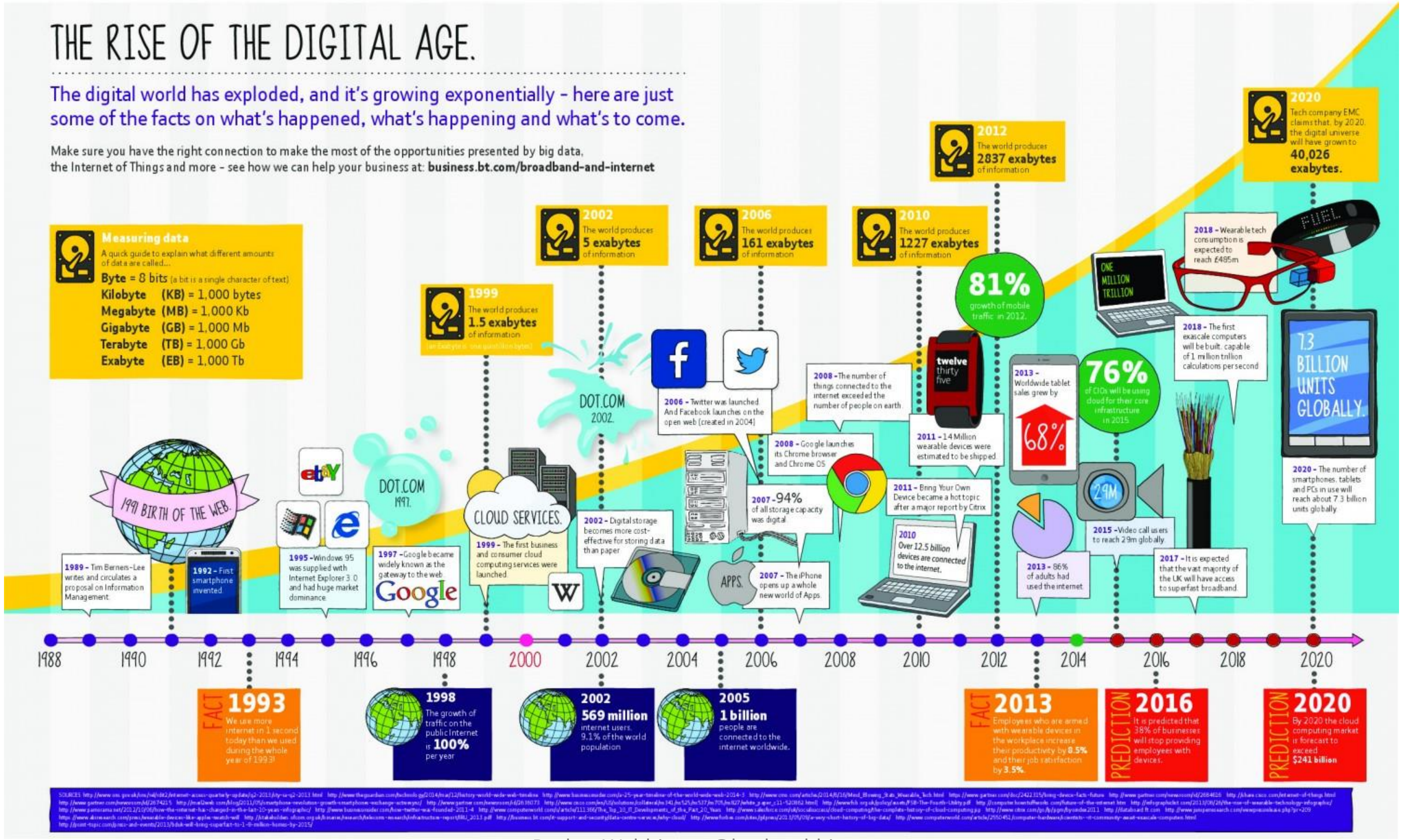
Global Information Storage Capacity

in optimally compressed bytes



Source: Hilbert, M., & López, P. (2011). The World's Technological Capacity to Store, Communicate, and Compute Information. *Science*, 332(6025), 60 –65. http://www.martinhilbert.net/WorldInfoCapacity.html

BT Business



THE RISE OF THE DIGITAL AGE.

The digital world has exploded, and it's growing exponentially – here are just some of the facts on what's happened, what's happening and what's to come.

Make sure you have the right connection to make the most of the opportunities presented by big data, the Internet of Things and more – see how we can help your business at: **business.bt.com/broadband-and-internet**

Measuring data
A quick guide to explain what different amounts of data are called...

Byte	= 8 bits (a bit is a single character of text)
Kilobyte	**(KB)** = 1,000 bytes
Megabyte	**(MB)** = 1,000 Kb
Gigabyte	**(GB)** = 1,000 Mb
Terabyte	**(TB)** = 1,000 Gb
Exabyte	**(EB)** = 1,000 Tb

1999 The world produces **1.5 exabytes** of information (an Exabyte is one quintillion bytes)

2002 The world produces **5 exabytes** of information

2006 The world produces **161 exabytes** of information

2010 The world produces **1227 exabytes** of information

2012 The world produces **2837 exabytes** of information

2020 Tech company EMC claims that, by 2020, the digital universe will have grown to **40,026 exabytes.**

1991 BIRTH OF THE WEB.

1989 – Tim Berners-Lee writes and circulates a proposal on Information Management.

1992 – First smartphone invented.

1995 – Windows 95 was supplied with Internet Explorer 3.0 and had huge market dominance.

DOT.COM 1997.

1997 – Google became widely known as the gateway to the web.

CLOUD SERVICES.

1999 – The first business and consumer cloud computing services were launched.

DOT.COM 2002.

2002 – Digital storage becomes more cost-effective for storing data than paper.

2006 – Twitter was launched. And Facebook launches on the open web (created in 2004)

2007 – 94% of all storage capacity was digital.

APPS.

2007 – The iPhone opens up a whole new world of Apps.

2008 – The number of things connected to the internet exceeded the number of people on earth.

2008 – Google launches its Chrome browser and Chrome OS

2010 Over 12.5 billion devices are connected to the internet.

2011 – Bring Your Own Device became a hot topic after a major report by Citrix

2011 – 14 Million wearable devices were estimated to be shipped.

81% growth of mobile traffic in 2012.

2013 – Worldwide tablet sales grew by 68%

2013 – 86% of adults had used the internet

76% of CIOs will be using cloud for their core infrastructure in 2015.

2015 – Video call users to reach 29m globally.

2017 – It is expected that the vast majority of the UK will have access to superfast broadband.

ONE MILLION TRILLION

2018 – The first exascale computers will be built, capable of 1 million trillion calculations per second.

2018 – Wearable tech consumption is expected to reach £485m

7.3 BILLION UNITS GLOBALLY.

2020 – The number of smartphones, tablets and PCs in use will reach about 7.3 billion units globally.

FACT **1993** We use more internet in 1 second today than we used during the whole year of 1993!

1998 The growth of traffic on the public Internet is **100%** per year

2002 **569 million** internet users, 9.1% of the world population

2005 **1 billion** people are connected to the internet worldwide.

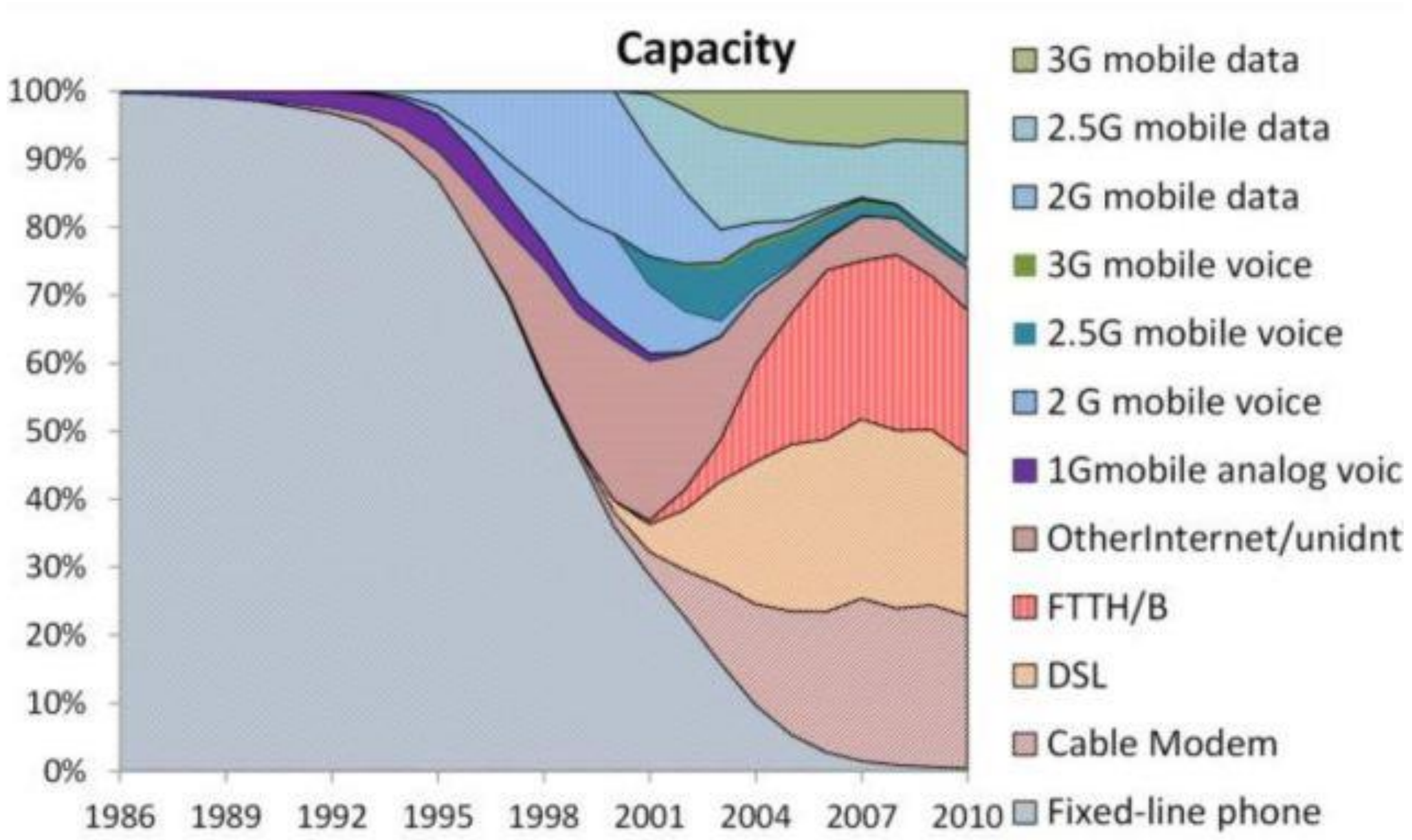
FACT **2013** Employees who are armed with wearable devices in the workplace increase their productivity by **8.5%** and their job satisfaction by **3.5%**.

PREDICTION **2016** It is predicted that 38% of businesses will stop providing employees with devices.

PREDICTION **2020** By 2020 the cloud computing market is forecast to exceed **$241 billion**

SOURCES [illegible]

The rise and fall of connectivity technologies



http://www.martinhilbert.net/mapping-the-dimensions-and-characteristics-of-the-worlds-technological-communication-capacity-during-the-period-of-digitization-1986-20072010/

Data Systems

Data Systems: Properties

- Reliability
  - The system should continue to work correctly even in the presence of faults & errors
- Scalability
  - As the system grows there should be a reasonable and affordable ways of dealing with this growth
- Maintainability
  - When different people work on the system over time, they should all be able to work productively

Data Systems: Reliability

- Reliable systems can tolerate and cope with faults. These systems are called fault tolerant or resilient.
- Faults have different natures:
  - Hardware faults: when a HW dies (Hard disk, RAM, …) we can:
    - Use redundancy: i.e. RAID 5
    - Software fault-tolerance in distributed systems.
    - Usually random and independent from each other
  - Software Errors: like software bugs, unresponsive dependency, cascading failures
    - Difficult to prevent. Testing, process isolation, measuring and monitoring.
  - Human errors: Humans are known to be unreliable ☺
    - A study shown that configuration errors by operators are the leading cause of outages. HW faults (servers & network) responsible for only 10 – 25% of outages.

Data Systems: Scalability

- Scalability isn’t one-dimentional either a system scales or not!
- Rather it is: if a system grows in a particular way, what are the options for coping with that growth.
- OR, how can we add resources to handle the additional load
- Scalability is all about coping with load. But what is load?
  - Requests per second, rows per second, volume per second
  - Read/write ratio in a database
  - Number of simultaneous connections/requests
  - Average, peak, percentile

Data Systems: Scalability

- In order to understand how the system behaves when the load changes, we need to understand the system performance
    - When you increase the load and keep the resources unchanged, how is the performance affected
    - When you increase the load, how much you need to increase the resources to keep the same performance
- Performance is measured using:
    - throughput (volume/sec), latency, response time
    - Measured in median values and percentiles (more important)
- How to cope with load?
    - Scale up (vertical scaling): Use high end machine. A system that can be run on the same machine is usually simpler.
    - Scale out (horizontal scaling): known as share nothing architecture. Use a distributed system to cope with load.
    - Both: use a few high end machines: simpler than using a large number of low end machines.

Data Systems: Scalability

- Example:
  - System A: 1,000,000 requests/sec with 1KB per request
  - System B: 10 requests/min with 6GB per request

  - Both systems have the same throughput: 1GBps but are fundamentally different

Data Systems: Maintainability

- The major cost of a system is not the initial development
- Rather, the debug and maintenance phases are those that cost the most

Dev cost < debug cost < maintenance cost

- **Operability**: make it easy for operations teams to keep the system running smoothly
- **Simplicity**: Make it easy for new devs to understand the system by avoiding unnecessary complexity
- **Evolvability** or extensibility: make it easy to make changes in the future.
- Recommendations for good maintainability:
    - Continuous monitoring of the system
    - Tracking the root cause of issues
    - Keeping systems up-to-data including security patches
    - Use abstractions to hide the complexity whenever possible
    - Precise and up-to-date documentation

Data Systems: Twitter Use Case

- Twitter was born in 2006. It has two main operations:
- Post a tweet:
  - user publishes a message to their followers.
  - Average 4.6K requests/sec (2012), 7.5K requests/sec (2016). 12K requests/sec in peak time (2012)
- Home timeline:
  - A user can view tweets posted by the accounts they follow.
  - 300K requests/sec (2012)

- Discussion around designing a Data system for Twitter.

Data Systems: Twitter Use Case

- Handling tens of thousands of writes per seconds (posting tweets) is fairly easy. This is not the challenge!
- The challenge is due to the **fan-out**:
  - Each user follows many accounts and is followed by many accounts.
  - Constructing the user timeline is equivalent to reading the tweets from the accounts the user follows sorted by time.

Data Systems: Twitter Use Case

- Strategy 1: Use a regular relational database
- *Tweets*, *users* and *follows* tables.
- Post a tweet:

```
INSERT INTO tweets (sender_id, text)
        VALUES (current_user, curent_tweet)
```

- Read timeline: find the accounts the user follows and find he tweets for each of those accounts then merge them while sorting by time:

```
Select tweets.*, users.* FROM tweets
        JOIN users ON tweets.sender_id = users.id
        JOIN follows ON follows.followee_id = users.id
        WHERE follows.followee_id = current_user
```

Data Systems: Twitter Use Case

- Strategy 2: Maintain a cache for each user’s timeline
- When a user posts a tweet, check all of its followers and insert the tweet to their timelines.
- Reading the timeline is then very straight forward.
- The downside is that posting a tweet now requires a lot of writes (writing to followers caches)
  - Average 345K writes/sec (2012)
  - Some accounts (celebrities) have up to 30million followers!!!
    - Twitter tries to deliver tweets to followers within 5 seconds

Data Systems: Twitter Use Case

- Strategy 3: Hybrid approach
- Posted tweets continue to be fanned out to followers timelines for most accounts
- Accounts with a large number of followers (celebrities) are excepted from this fan-out
- Tweets from celebrities are fetched separately.
- Reading the timeline is now:
  - Read from account timeline
  - Read from celebrities the account follows
  - Merge both to construct the final home timeline

Big Data Use Cases

Big Data: Use Cases

- Data storage at a scale
- Extract Transform Load (ETL)
- Data exploration and analysis
- Reporting
- Text mining
- Search & Indexing
- Predictive Analytics
- Recommendation
- Sentiment Analysis
- Object & Pattern recognition
- Machine Learning & Artificial Intelligence

- These use cases have one thing in common: The Data and its nature
- This is what is commonly referred to as 3V
  - Volume
  - Variety
  - Velocity
- IBM added a 4th V
  - Veracity
- But it is all about Value

What is Big Data

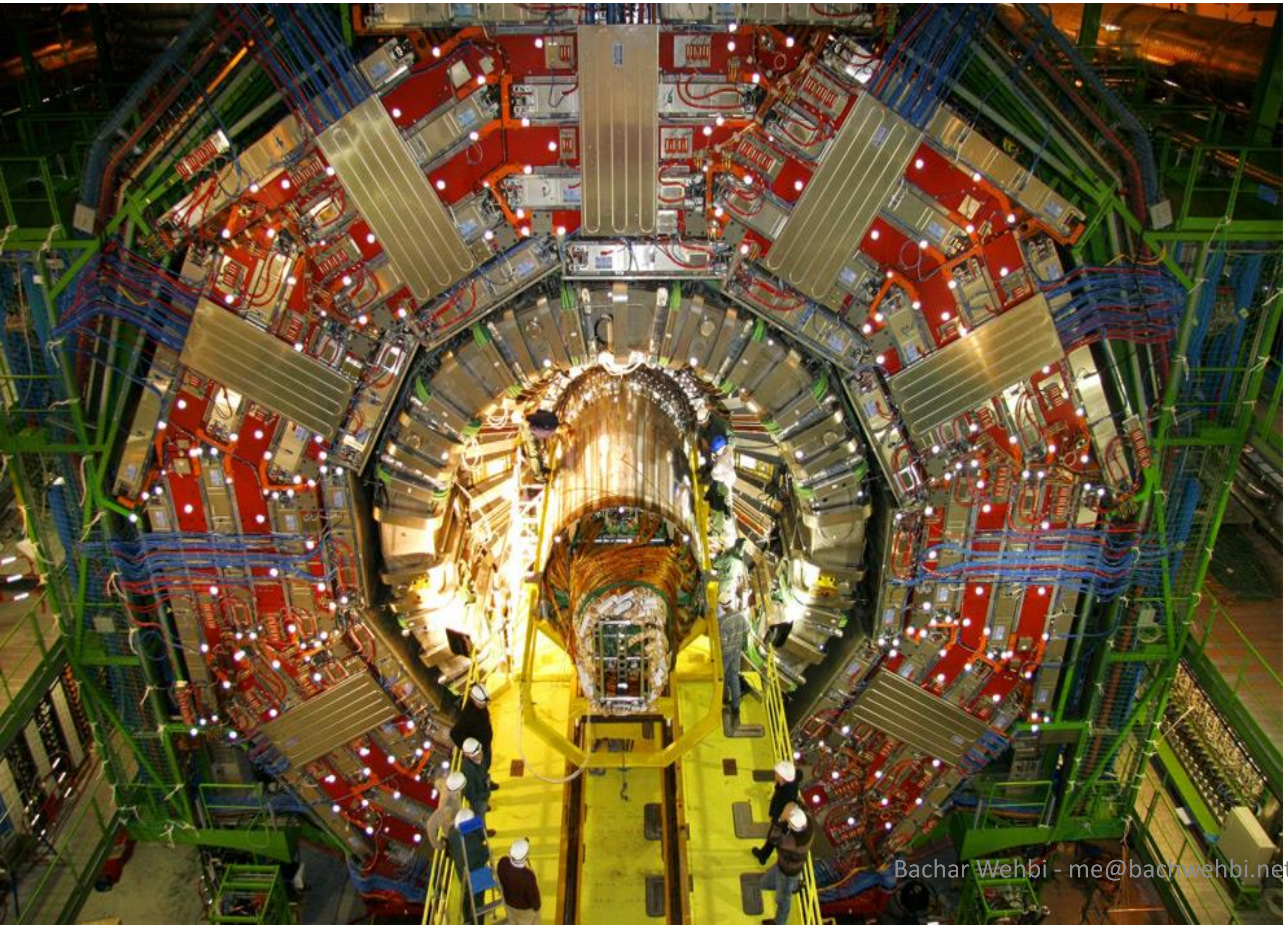
Big data is **data sets** that are so **voluminous and complex** that traditional data-processing application software are inadequate to deal with them.

Big data challenges include **capturing data**, **data storage**, **data analysis**, **search**, **sharing**, transfer, visualization, querying, updating, information privacy and data source.

There are five concepts associated with big data:

volume, variety, velocity and, the recently added, veracity and value

Big Volume



Cern’s Large Hydron Collider (LHC)

Particles collide with sensors 1 Billion times per second

Generate 1 PB of raw data per second

7.5 million seconds of experiments in 2016

49 PB of cleaned experiments data in 2016

https://home.cern/about/updates/2017/07/cern-data-centre-passes-200-petabyte-milestone

Big Volume

Imagine The size of a cluster with these specs!

Samsung 30TB SSD Disk!

AWS EC2: x1.32xlarge

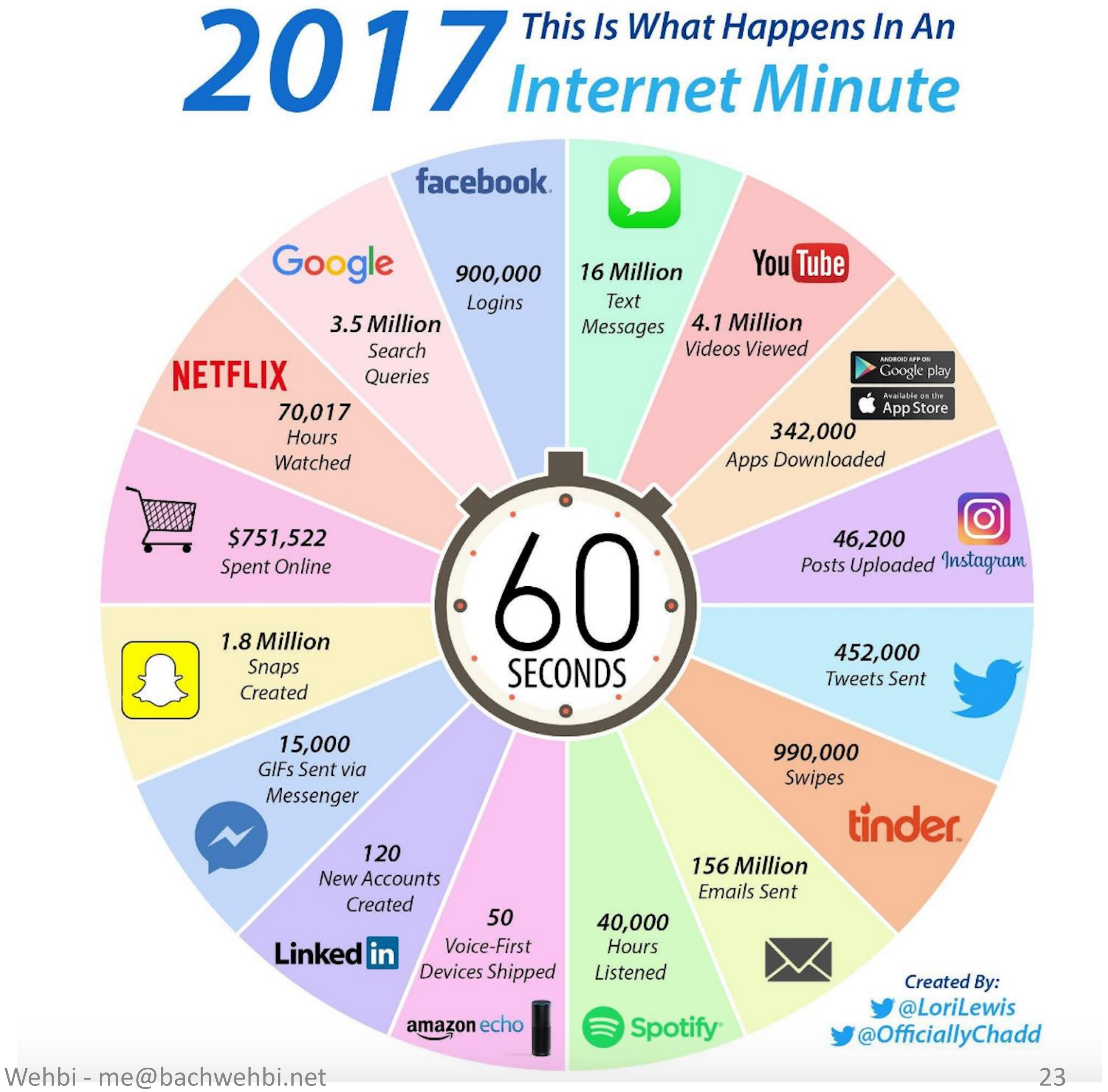
128 Cores – 3.9TB RAM – 2 * 1.9 TB SSD

https://news.samsung.com/global/samsung-electronics-begins-mass-production-of-industrys-largest-capacity-ssd-30-72tb-for-next-generation-enterprise-systems
https://aws.amazon.com/ec2/instance-types/

Big Velocity

- **Velocity**: the speed at which the data is generated and processed.

- Big data is often available in real-time.



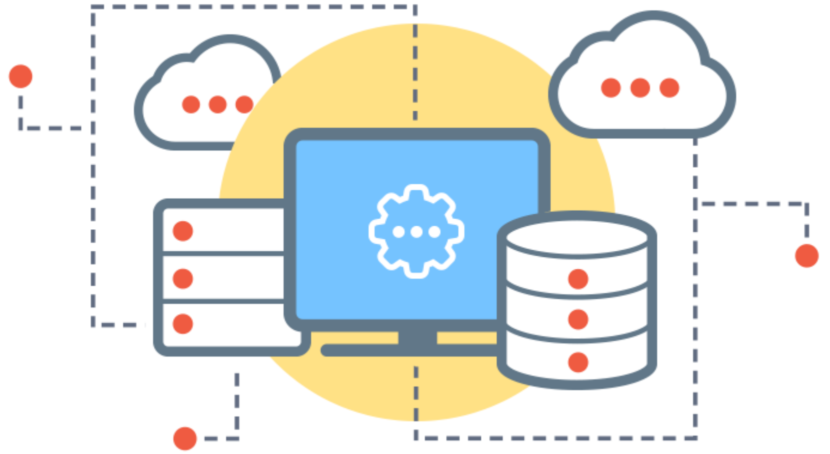
Big Velocity

"From the dawn of civilization to 2003, five exabytes of data were created. The same amount was created in the last two days.“

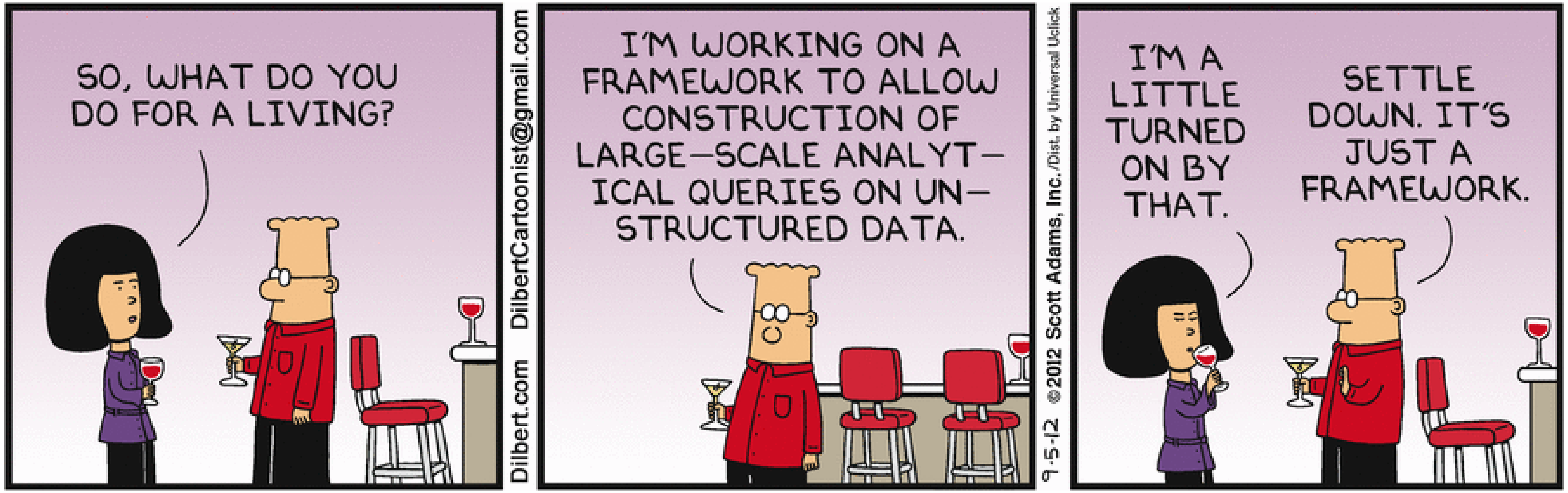
(Eric Schmidt - 2010)

Big Variety

- **Variety**: The type and nature of the data.
- Big data draws from text, images, audio, video.
- Data sources are both internal and external to the organization.
- Data is structured, semi-structured and unstructured.

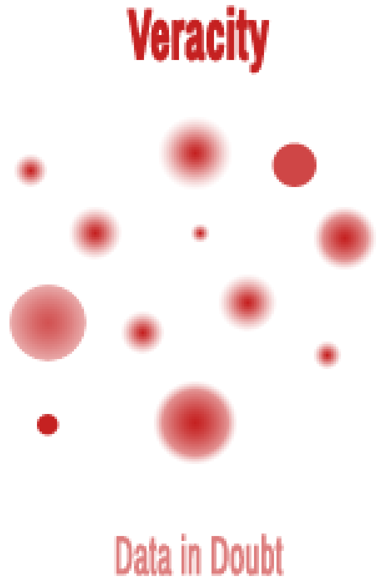


Big Variety


SO, WHAT DO YOU DO FOR A LIVING?
I'M WORKING ON A FRAMEWORK TO ALLOW CONSTRUCTION OF LARGE-SCALE ANALYTICAL QUERIES ON UNSTRUCTURED DATA.
I'M A LITTLE TURNED ON BY THAT.
SETTLE DOWN. IT'S JUST A FRAMEWORK.
Dilbert.com DilbertCartoonist@gmail.com
9-5-12 ©2012 Scott Adams, Inc./Dist. by Universal Uclick

Big Veracity

- **Veracity:** refers to the quality of trustworthiness of the data.
- Big Data gives you the opportunity to analyse all of the data, but on the other hand the data contains lots of noise.
- Company has to be able to transform the data into trustworthy insight and discard noise.
- Example of untrustworthy noise "fake news", Spam, Bots, etc.



Big Value

Datafication

- CRM
- Sales
- Products
- ERP
- Email
- Social
- Video
- Audio
- Click Stream

Volume

Velocity

Variety

Veracity

Analysing Data

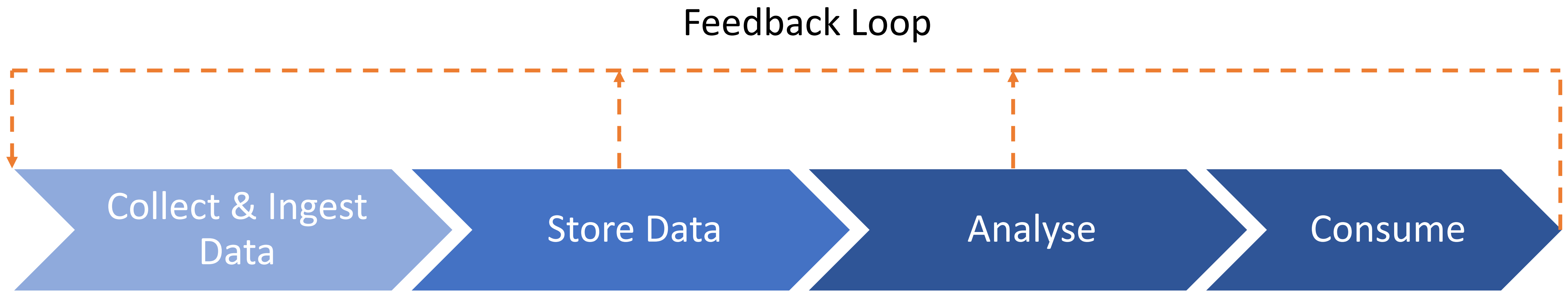
- Reporting
- Text mining
- Search & Indexing
- Predictive Analytics
- Recommendation
- Sentiment Analysis
- Object recognition
- Artificial Intelligence

Value

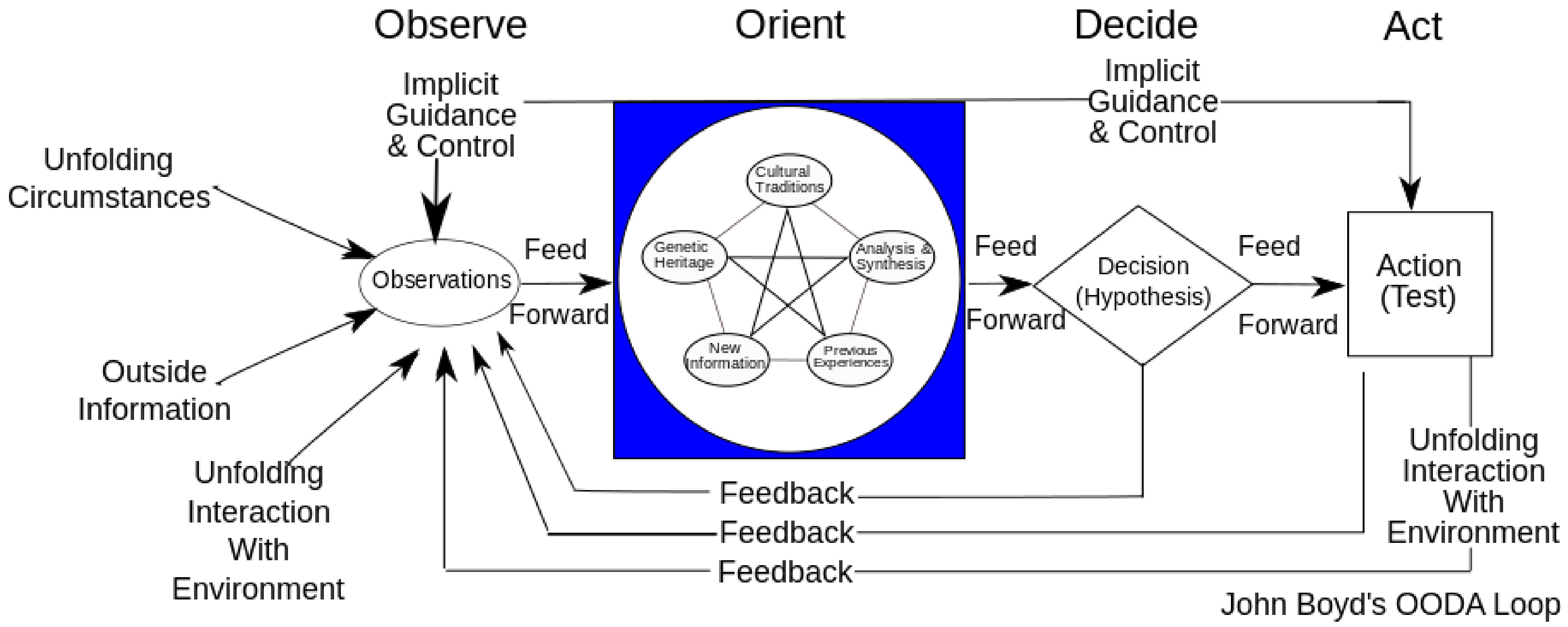
Big Data: Approach

Iterative Process

Continuous Improvement



Big Data: OODA LOOP



Components of a (BIG) Data Platform

Components of Data Platform

Collect & Ingest Data

- Load data from source to destination
- Batch ingestion for mass of data
- Streaming ingestion for real-time data

Store Data

- Massively scalable
- Structured and non structured data
- Varying performance objectives (bulk/random read write)

Analyse

- Distributed and massively scalable
- Batch processing for large amounts of data (periodic)
- Streaming processing for real-time or close to real time

Consume

- Querying data in SQL or NoSQL stores
- Feeds reporting and business intelligence tools
- Search data

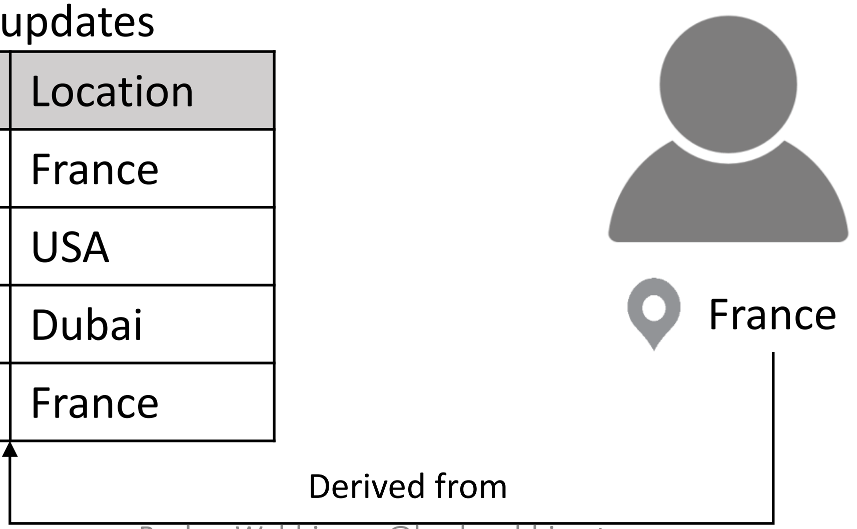
What is Data?

Data

- Data is more than Information
- Not all information is equal
  - Some information is derived from other pieces of information

Profile updates

Timestamp	Location
2010	France
2014	USA
2017	Dubai
2018	France



Data

- An indivisible unit that holds to be true simply because it exists
  - Time based: fact known to be true at some moment of time
  - Inherently immutable: no such thing as update or delete!!!

Profile updates

Timestamp	Location
2010	France
2014	USA
2017	Dubai
2018	France

Still true

We are now in 2018

In its raw form, Data can only be created and read
This simplifies dramatically the CRUD model used in RDB

Data

- Data is therefore timestamped events
- In the past, events used to change the master data
    - CRUD model in relational databases
- In Data Architecture: events are the master data, the ground truth

→ Let’s store everything

Data System

Query = Function(All Data)

Simple definition that includes every usage of data from joins, aggregations to advanced analytics and machine learning

Data System

Number of persons living in each country

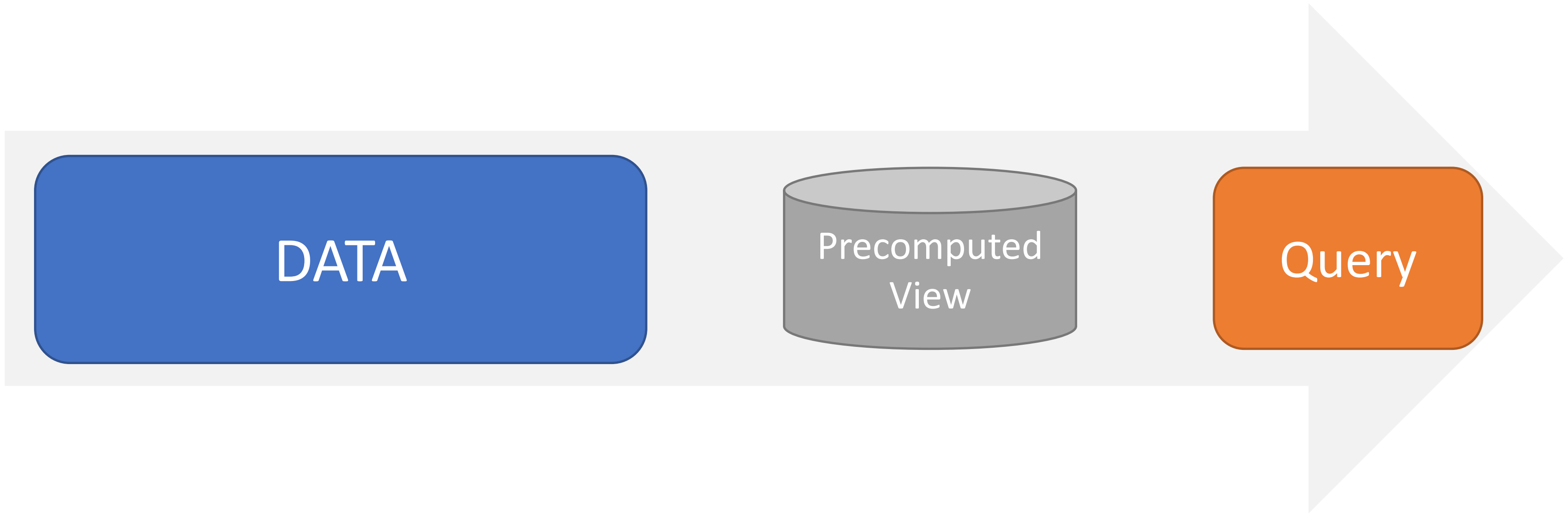
Timestamp	Name	Location
2010	Sara	France
2014	Bachar	USA
2016	Sara	USA
2016	Nadine	France
2017	Nadine	Emirates
2018	Bachar	France

$\Rightarrow$

Country	Count
France	1
USA	1
Emirates	1

Data System

Query = Function(All Data)



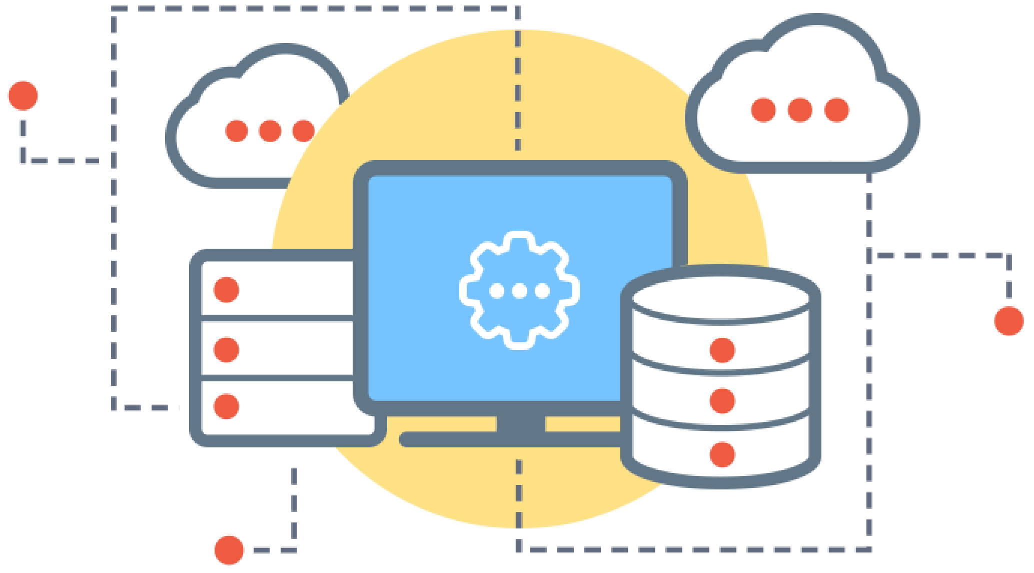
Data Architecture

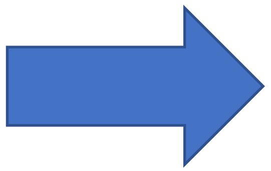
Data is a value store

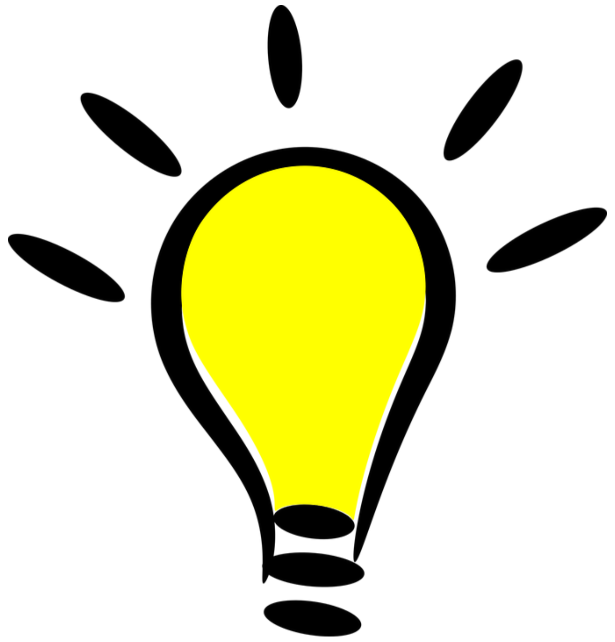
Data

Questions

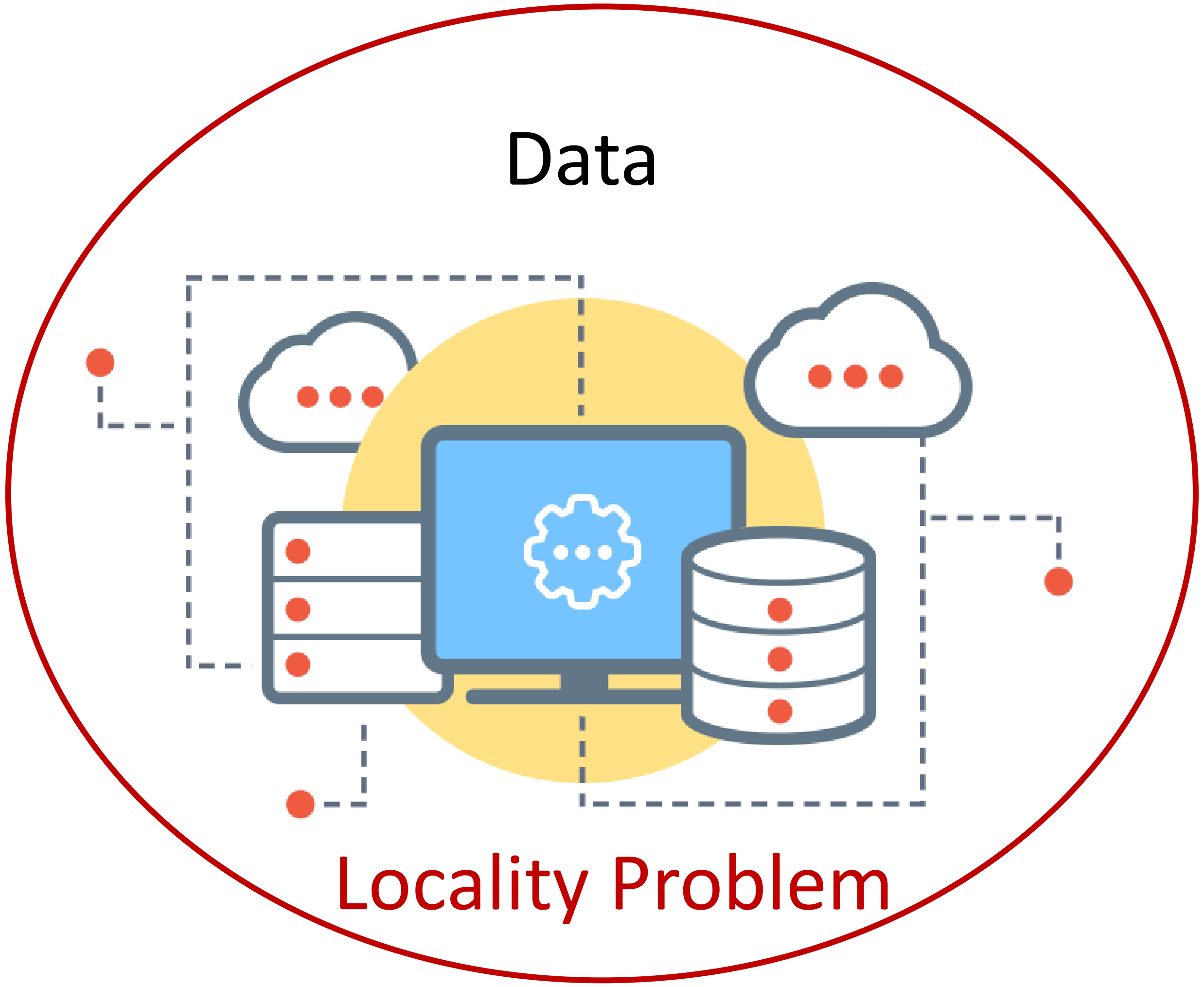
Results

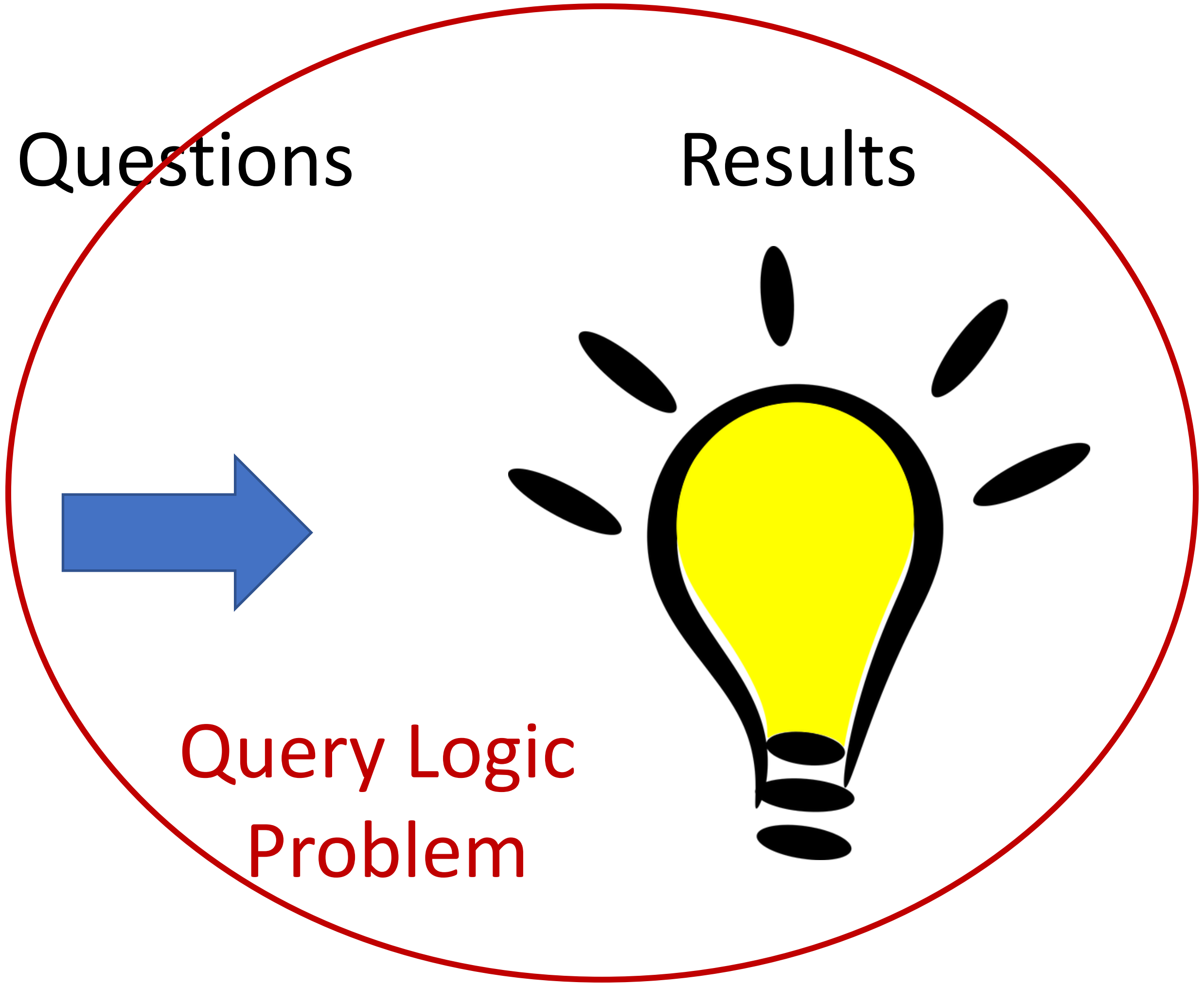






Data is a value store





Really hard when data is spread in multiple systems

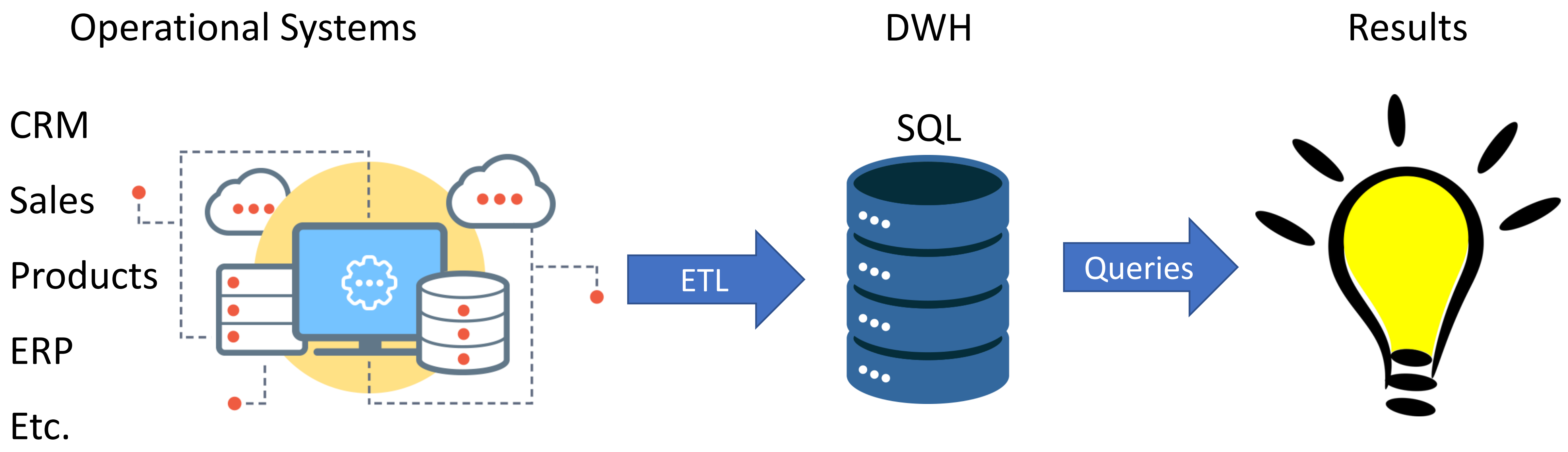
Introducing the Data Warehouse

Centralise operational data

to enable

Business Intelligence & decision making

Data Warehouse



Data Warehouse

Operational Systems

DWH

Results

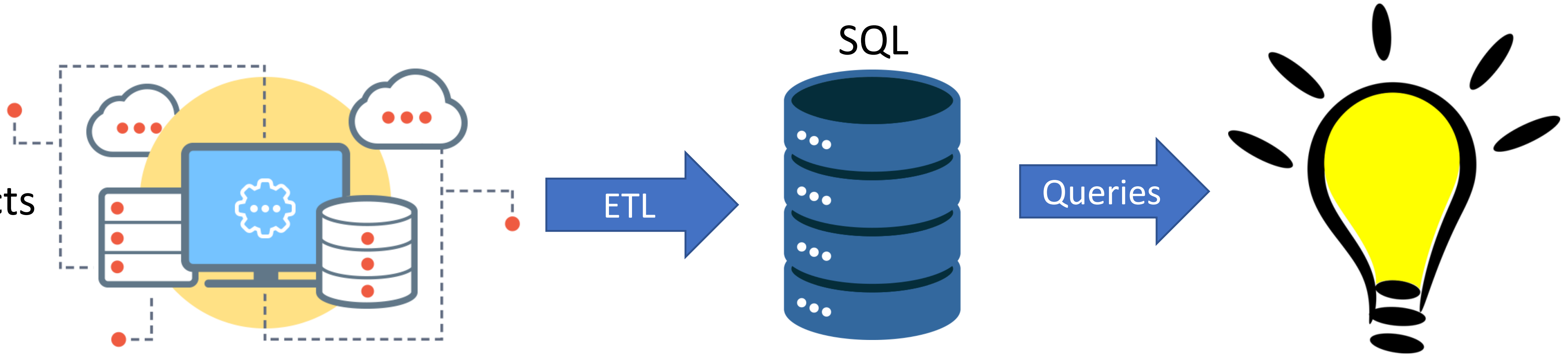
CRM

Sales

Products

ERP

Etc.



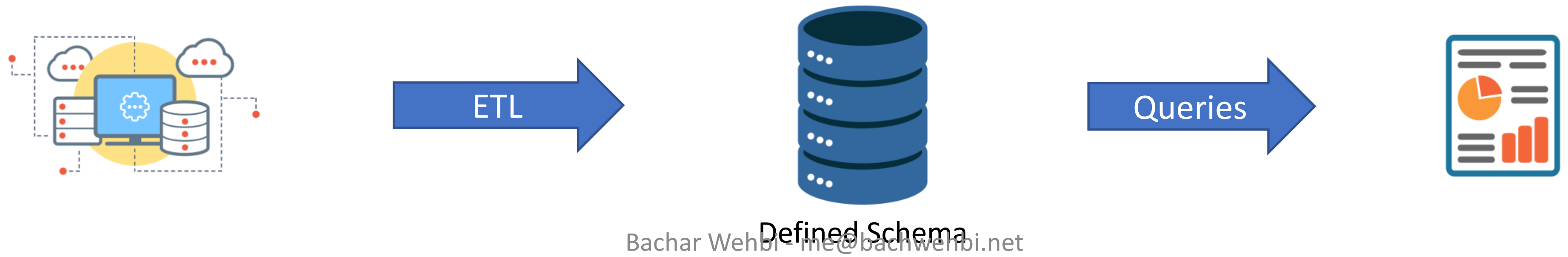
Structure data in RDB to address Data Location & Query Logic (SQL)

Load data from operational systems to central DWH and get Business Intelligence

Data Warehouse

Traditional process when building DWH

1. Start with use cases to identify required analysis and reports
2. Design DB schema and queries
3. Identify the required data sources
4. Create ETL pipelines to extract data from sources, transform it into target schema and load it in DWH (schema-on-write)
5. Query DWH for defined reports and analysis



Data Warehouse

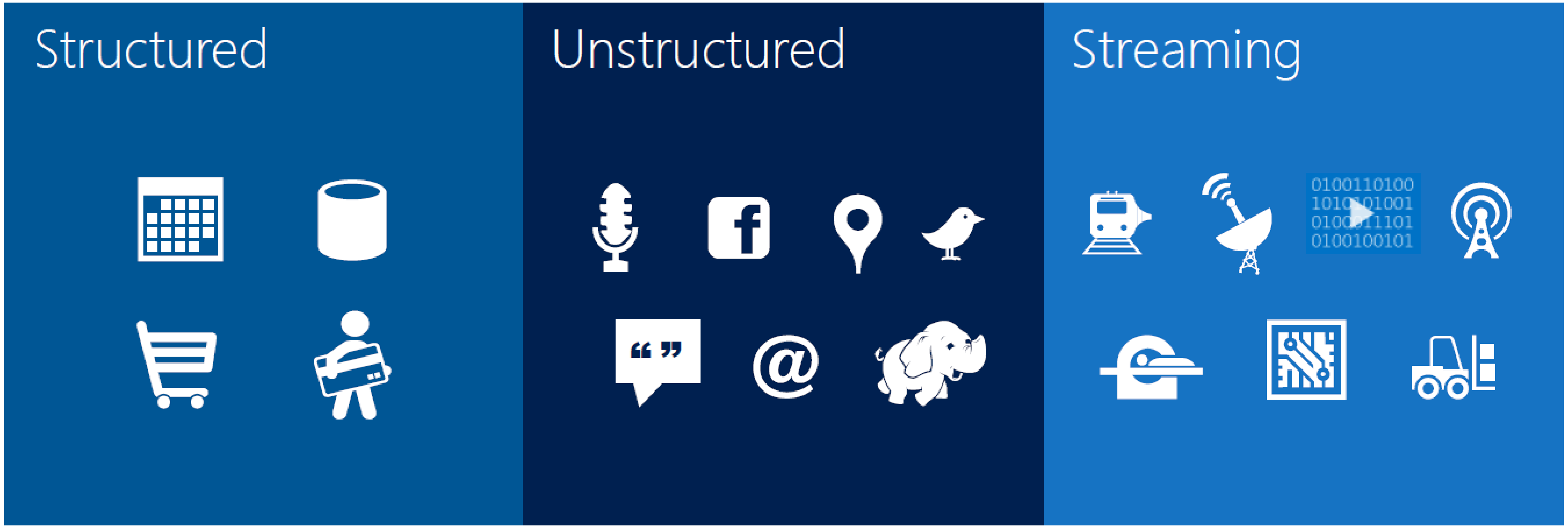
The Good

- Centralized and fresh data
- Fast queries
- Transactional

The Bad

- Hard to scale & non Elastic
- Mostly limited to internal structured data
- Not for advanced analytics & machine learning and
- Closed format (locked in)

Need to collect a variety of data



Challenge to store structured transactional data with less structured data

Do not necessary know in advance what use cases to make with the data

Big Data = All Data

So was born the Data Lake

Get <u>ALL DATA</u> in one scalable store

to enable

<u>Any future use case</u>

Data Lake

Internal + External Data

Results

CRM

Sales

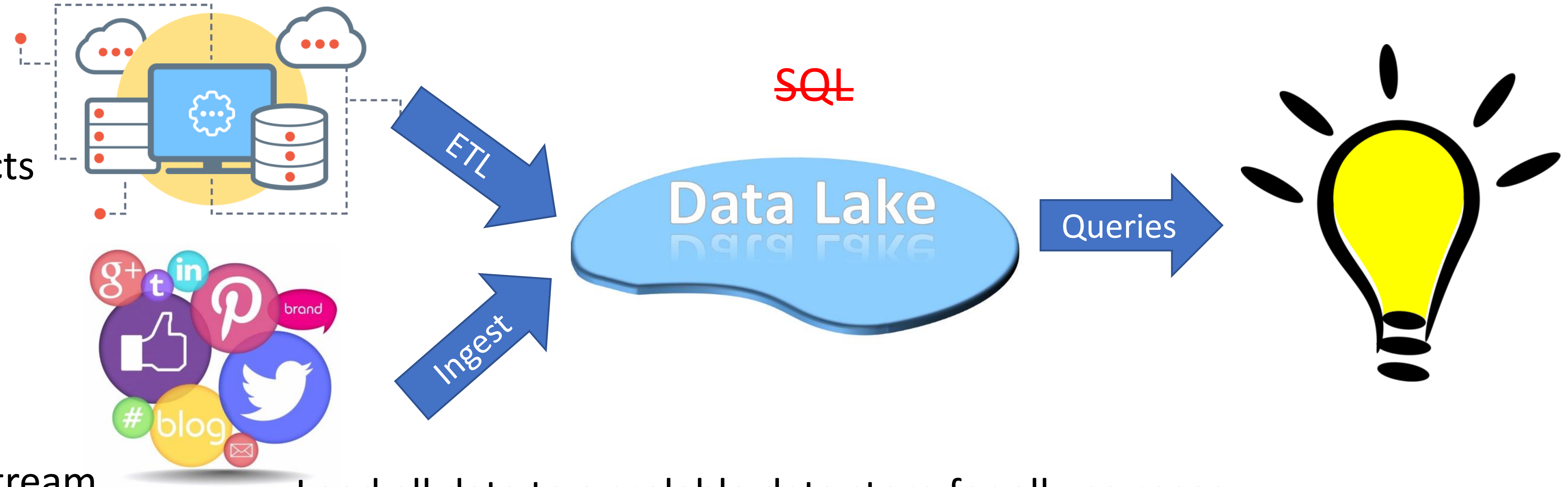
Products

ERP

Email

Social

Click Stream



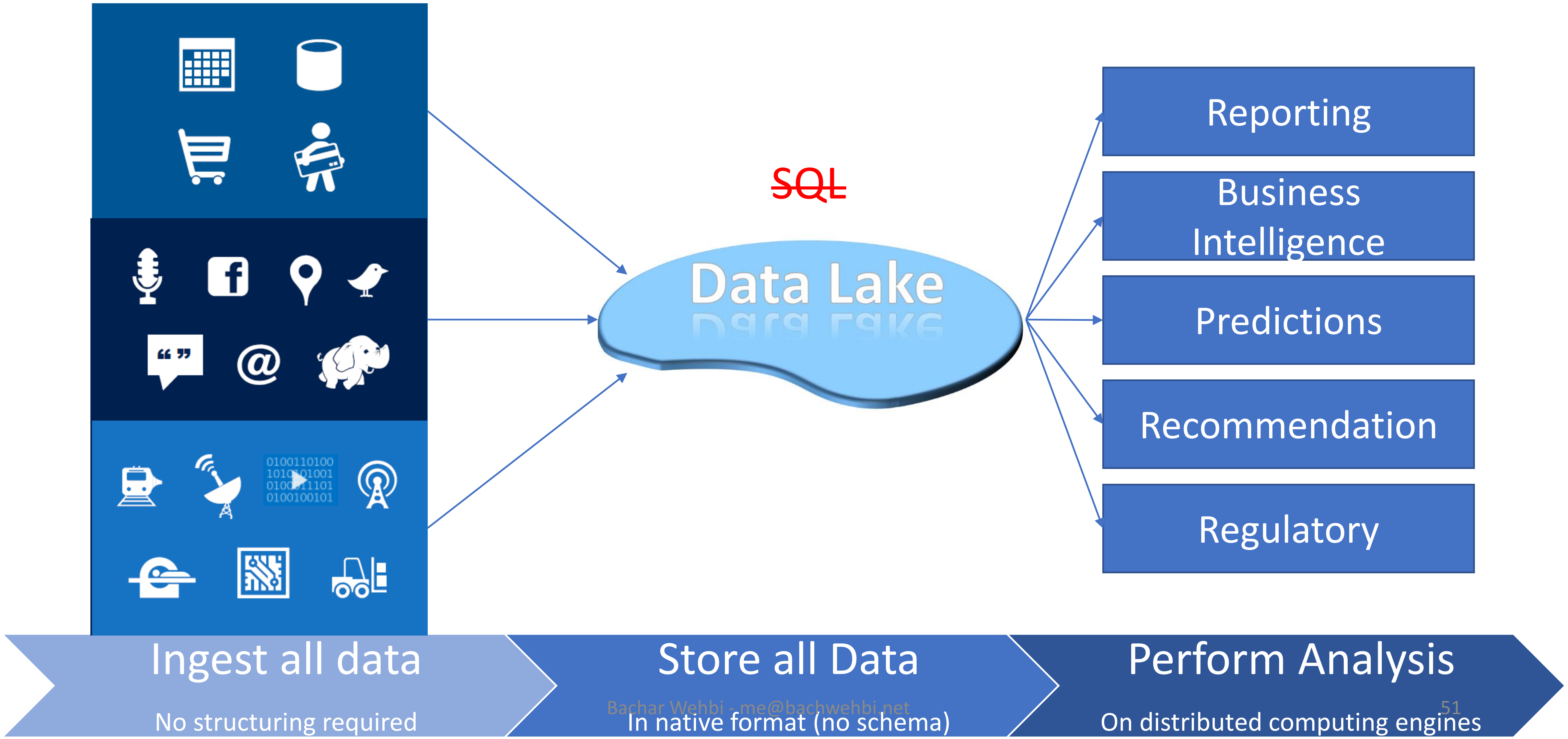
Load all data to a scalable data store for all use cases

Ingest all data
No structuring required

Store all Data
In native format (no schema)

Perform Analysis
On distributed computing engines

One Data Lake, Multiple use cases



Data Lake

The Good

- Massive scalability
- Relatively inexpensive (per TB)
- Built on open formats
  - Avro, ORC, Parquet
- Adapted for advanced analytics and machine learning
- Raw data (all data)

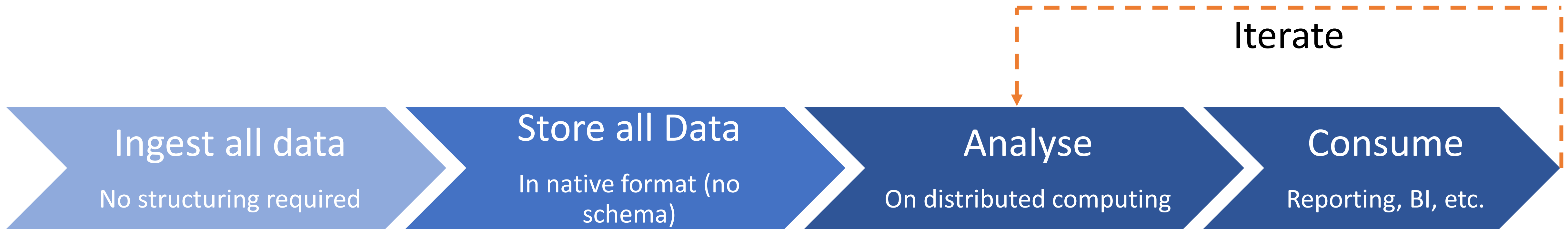
The Bad

- Raw data
  - Inconsistent
  - no schema
- Requires processing
  - Analytics
  - Reporting
- Performance

Data Lake

Modern approach for data systems

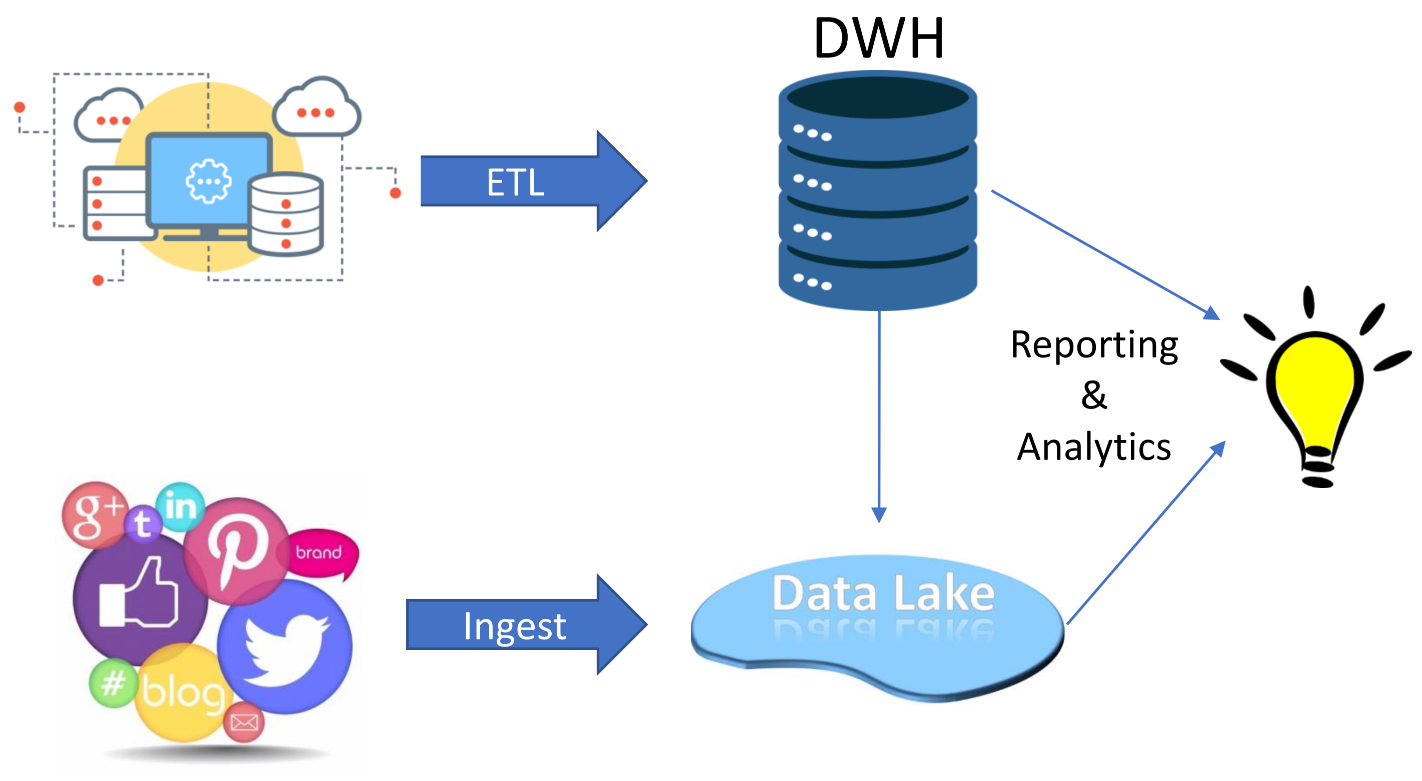
- Any data has a value ➔ Store everything
- Use cases may be defined in the future ➔ store data in native format (raw data), do not impose a rigid schema
- Schema is imposed on data read (queries, analysis, etc.)
- schema-on-read



Building a Data Lake is Complex

And there is no ONE way to do it

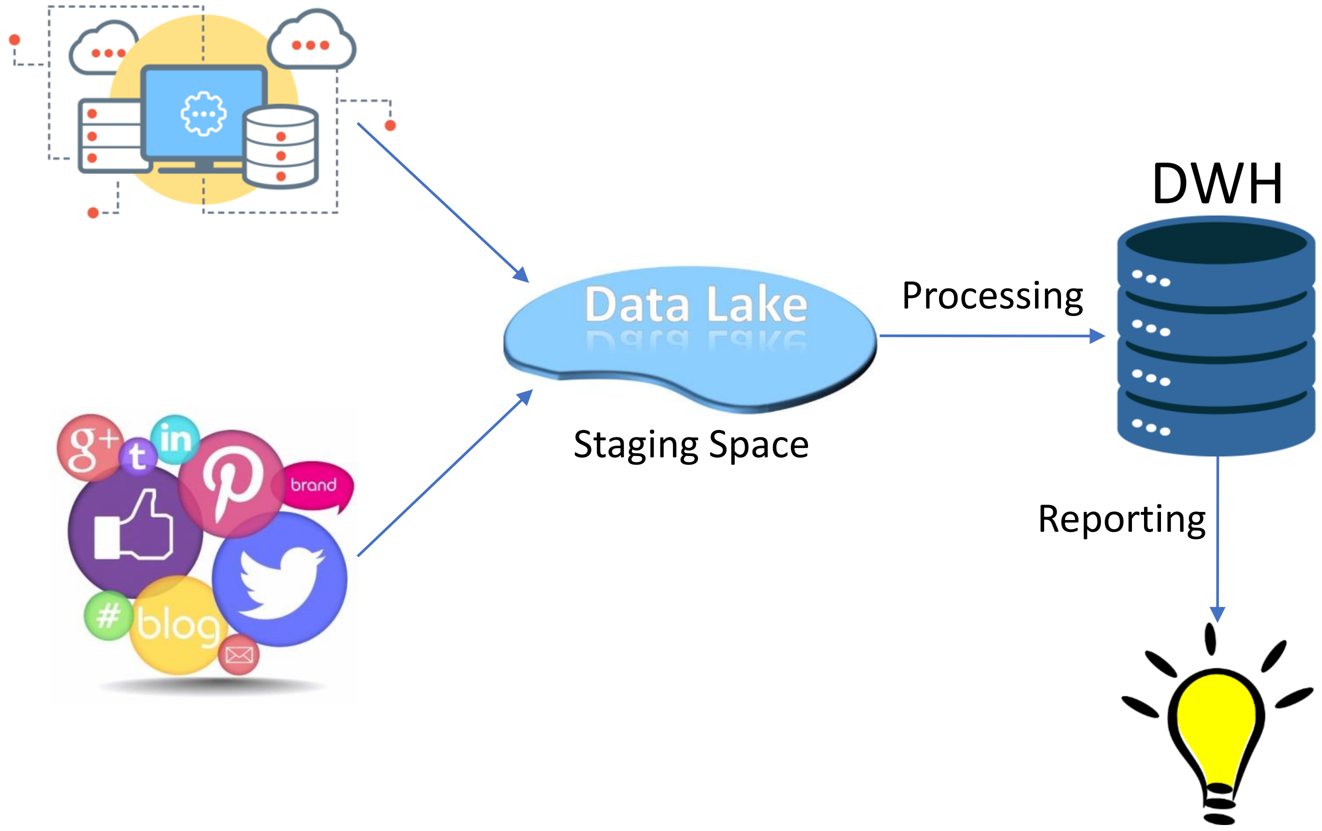
Augmented Data Warehouse



- When DWH already exists
- DWH incapable of handling new data
  - New data goes directly to Data lake

- Cons
  - Complex architecture: two systems
  - Complex ingestion (3 ways)
  - Data with different maturity

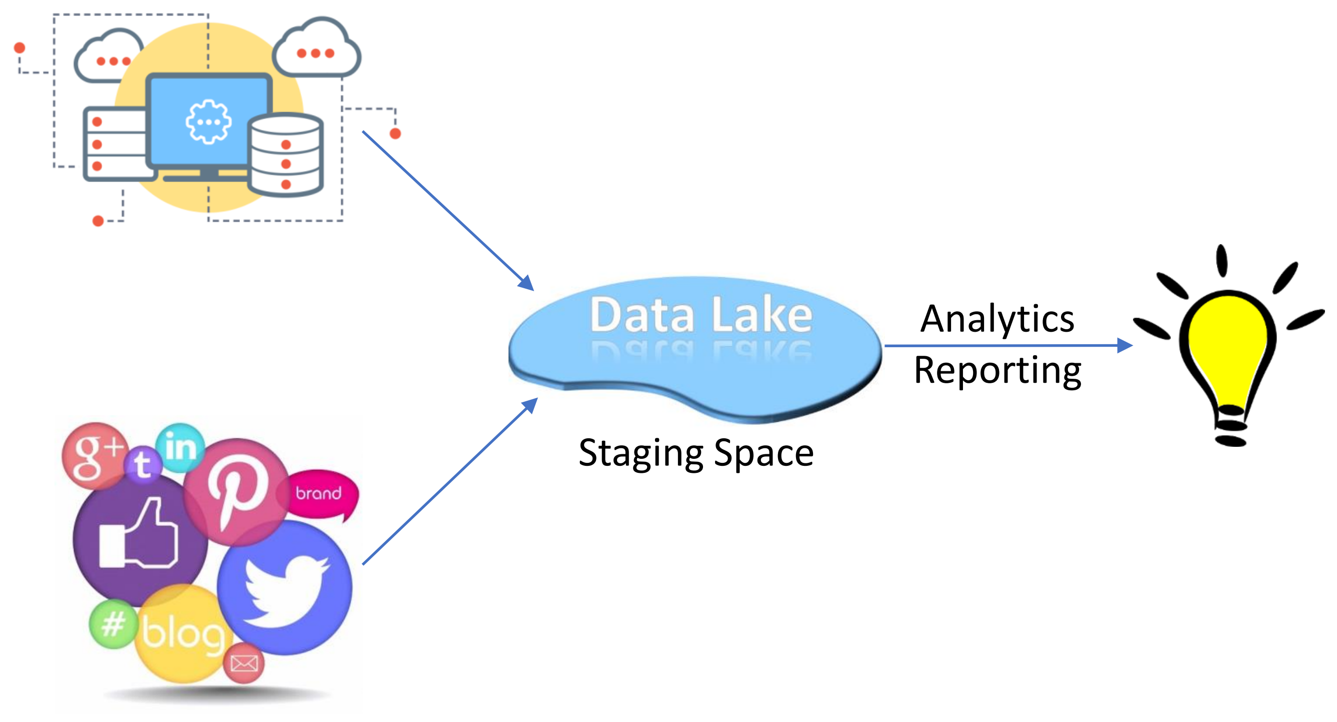
Data Lake & Data Warehouse



- Data Lake centralizes raw data in one place
- Data processed and moved to DWH
  - Batch processing

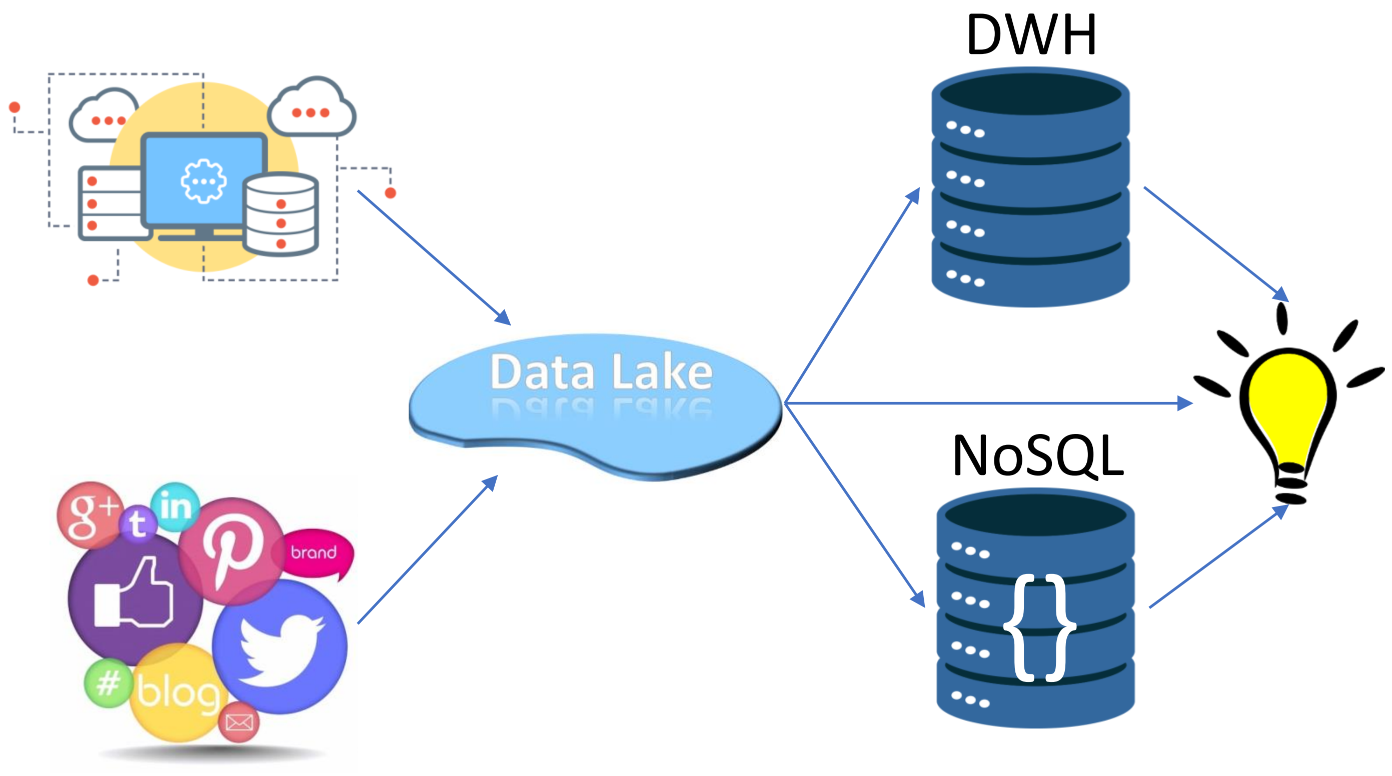
- Cons
  - Not using all the potential of Data Lake
  - Not suited for any data use case (advanced analytics and machine learning)

All in one Data Lake



- When there is no existing DWH
- Data Lake for all usages
  - Staging, processed data, reporting, analytics
- Cons
  - Performance (especially for interactive reporting)
  - Can’t use existing BI tools (or requires major adaptation)

Target Architecture



- Use the complete Data ecosystem (Data Lake, DWH, NoSQL)
- Data centralized in the Data Lake
- Data harmonized in the Data Lake
- Processing results copied to DWH or NoSQL DB as required
    - Read Only
- Future proof: adapted for today and future use cases

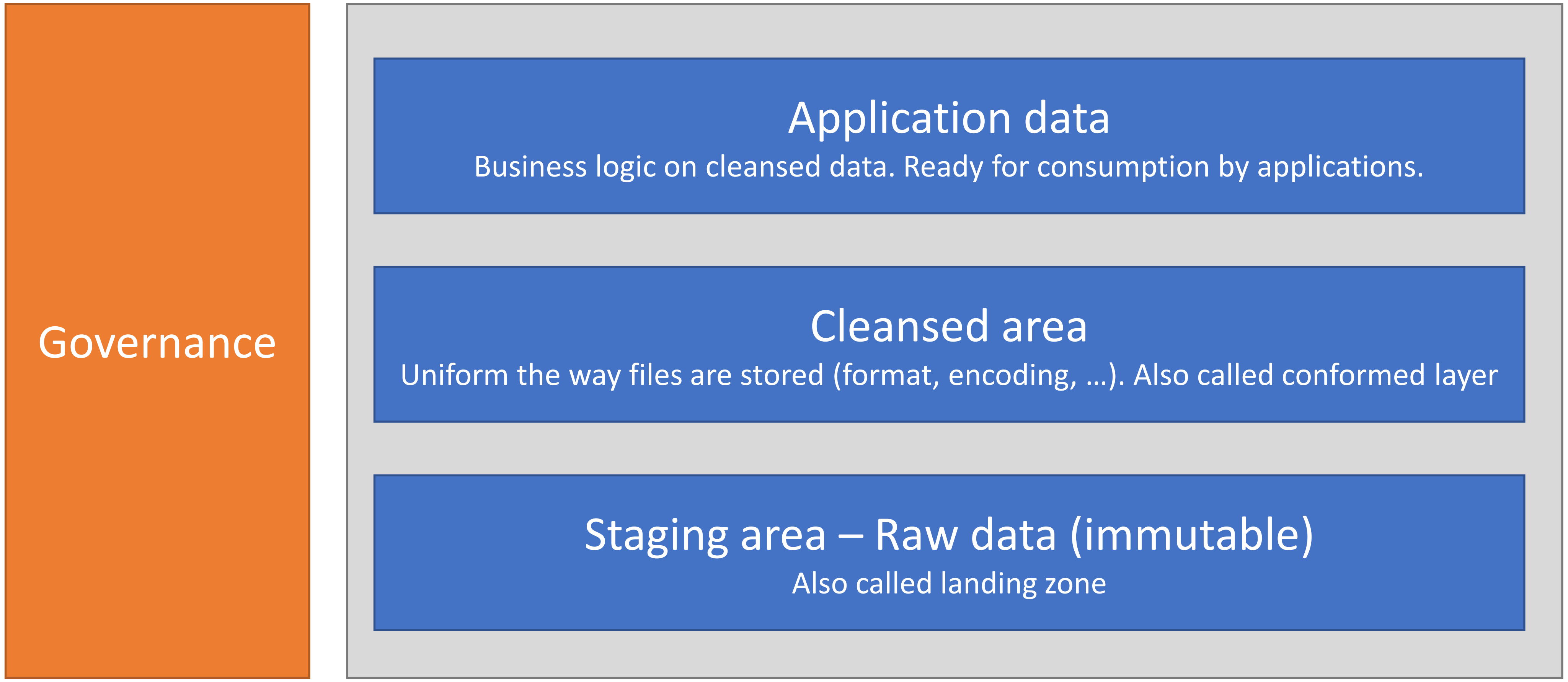
Data Lake

Paradigm shift

Before: Structure ➔ Ingest ➔ Analyse

Now: Ingest ➔ Analyse ➔ Structure

Data Lake


Governance
Application data
Business logic on cleansed data. Ready for consumption by applications.
Cleansed area
Uniform the way files are stored (format, encoding, …). Also called conformed layer
Staging area – Raw data (immutable)
Also called landing zone

Lambda Architecture

Lambda architecture is a **data-processing architecture** designed to

handle **massive quantities** of data by taking advantage of both **batch** and **stream-processing** methods.” (Wikipedia)

Nathan marz

- Ex- Twitter Engineer
- Creator of Apache Storm

http://nathanmarz.com/blog/how-to-beat-the-cap-theorem.html

Lambda Architecture: Why

- Fault-tolerance
  - against hardware failures
  - and against human errors (which occur more frequently: bugs, errors, etc)
- Support for a variety of use cases
  - Low latency querying as well as updates
- Linear scale-out capabilities
  - Adding more machines ➔ speed up the whole process
- Extensibility to support new use cases easily

Lambda Architecture: Properties

- Scalable data store
  - Easily store and scale immutable, constantly growing dataset
- Append only data store
  - The primary write operation is adding new immutable data
  - Updates and deletes are no different than creating new data
- Avoid complexity of the CAP theorem (and human errors)
  - recomputing queries from raw data
- Incremental algorithms
  - Lower latency of queries to an acceptable level

Lambda Architecture: Master Dataset

Master Dataset: the source of truth in the Lambda Architecture.

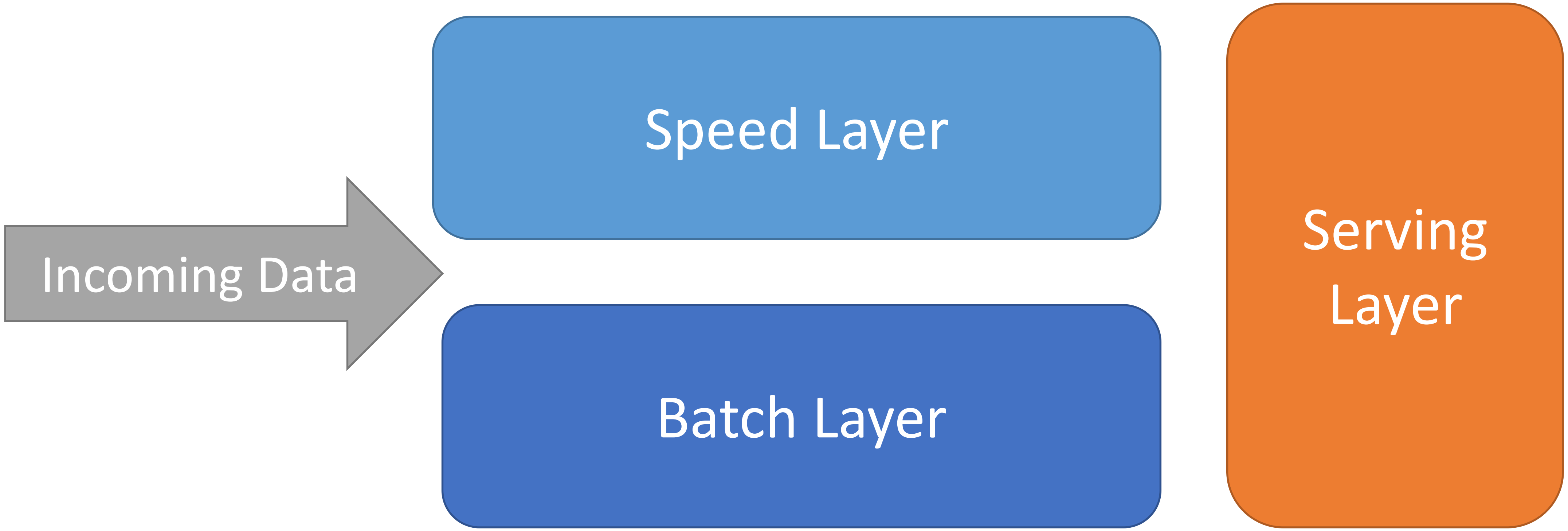
- If we loose the computed views (failure) we could reconstruct the application from the master dataset.

Data in master dataset is :

- Raw in its original format: append only
- Immutable: write once read many
- Eternally true

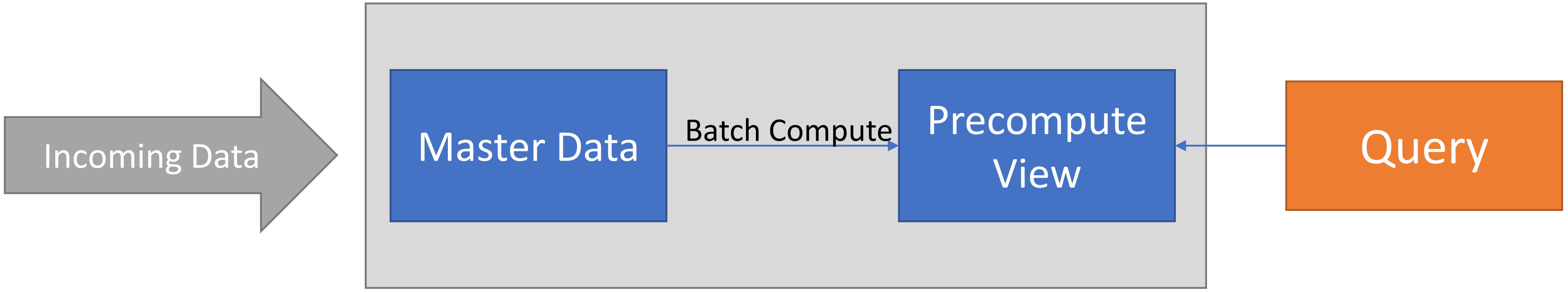
Lambda Architecture

Lambda is a layered architecture composed of 3 components



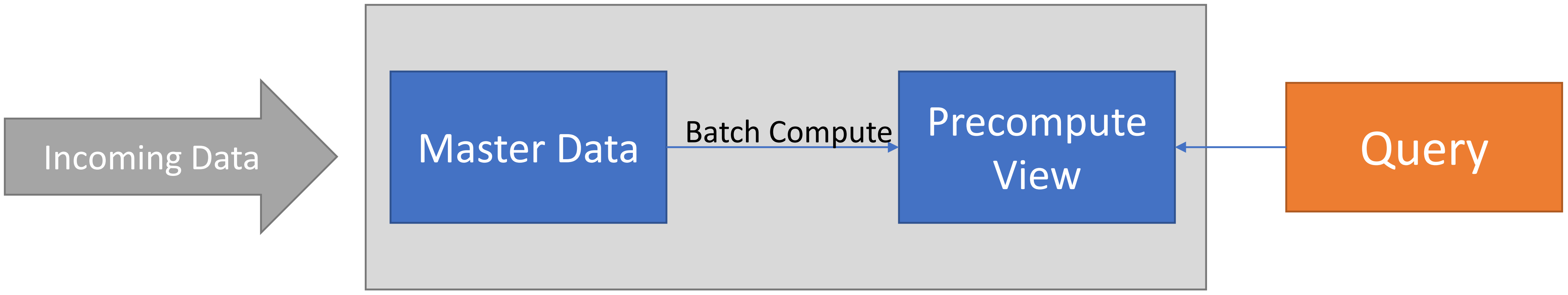
Lambda Architecture: Batch Layer

- Batch layer precomputes the master dataset into batch views
- Queries can be resolved with low latency
- Unrestrained computation: can compute anything given enough time



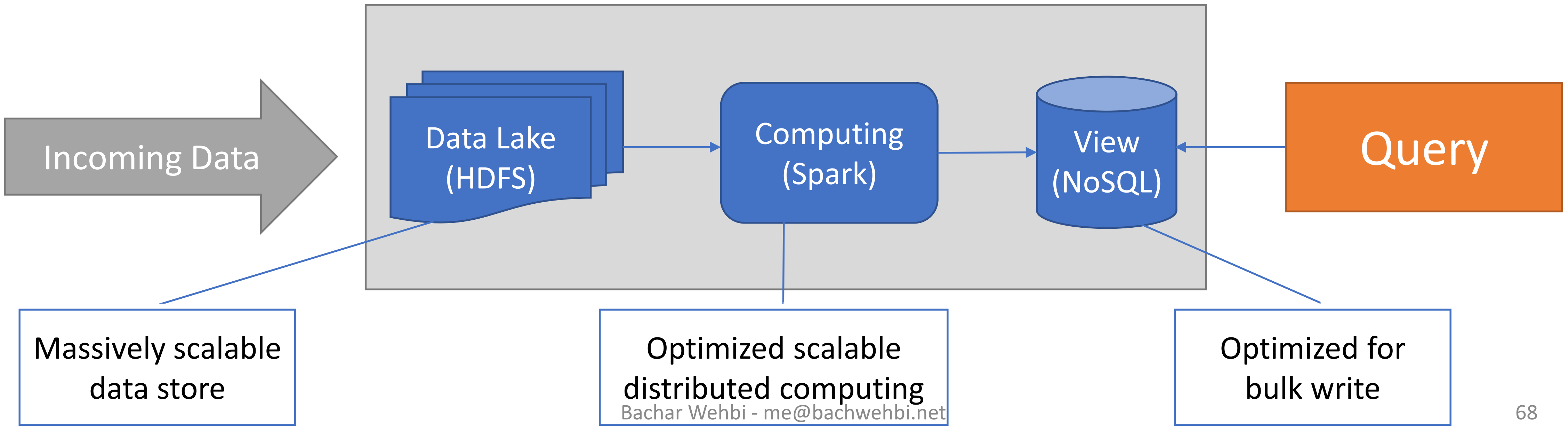
Lambda Architecture: Batch Layer

(Re)-Computation based on immutable dataset is idempotent
➔ Recomputing will yield to the same result



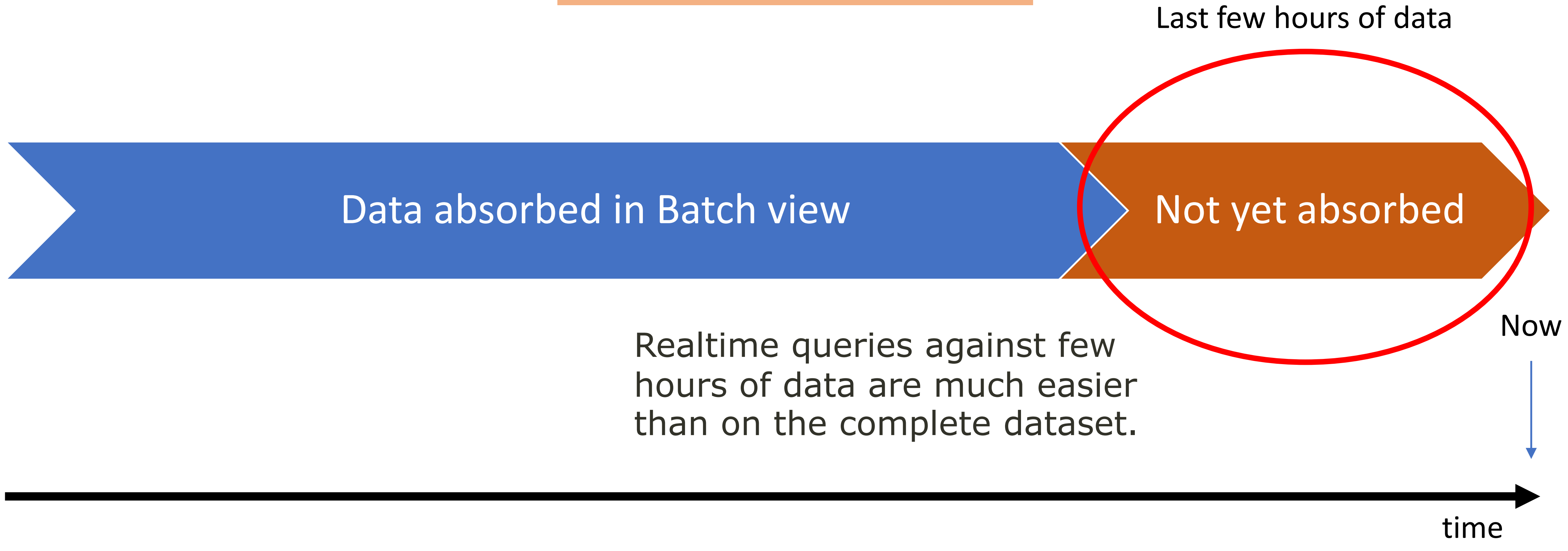
Lambda Architecture: Batch Layer

(Re)-<u>Computation</u> based on <u>immutable</u> dataset is <u>idempotent</u>
➔ Recomputing will yield to the same result



Lambda Architecture: Batch Layer

We are almost there!



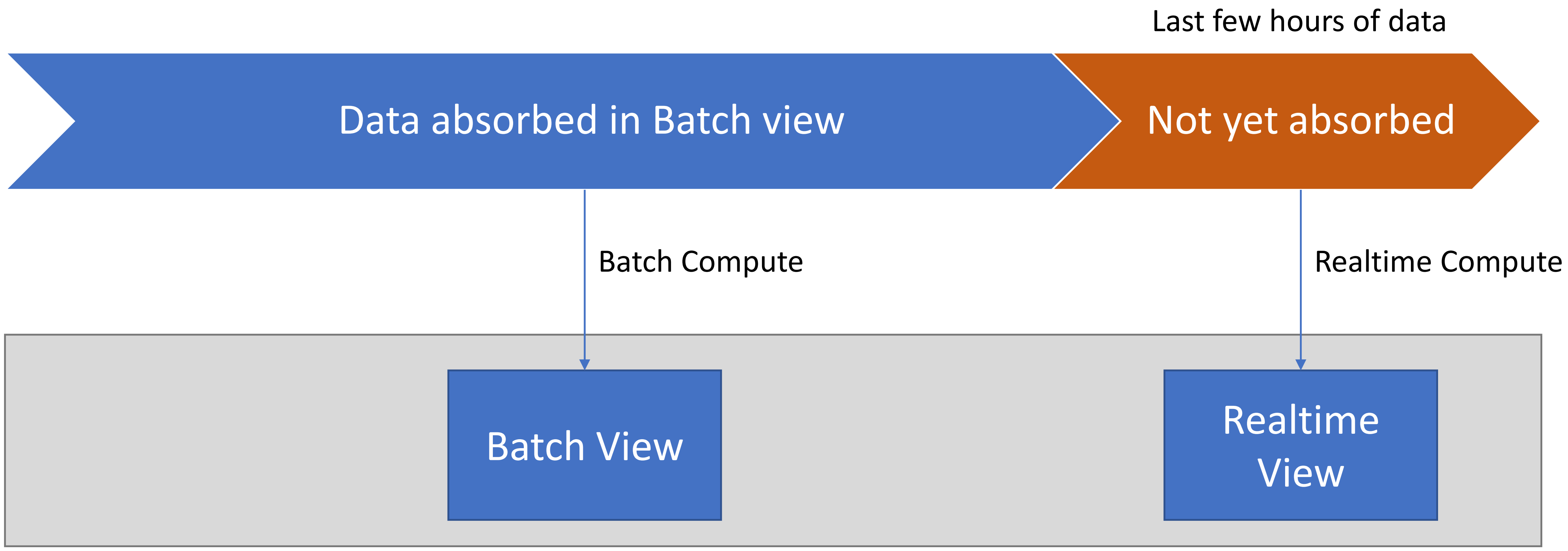
Realtime queries against few hours of data are much easier than on the complete dataset.

Speed Layer

The Batch layer processes data up to few hours precision.

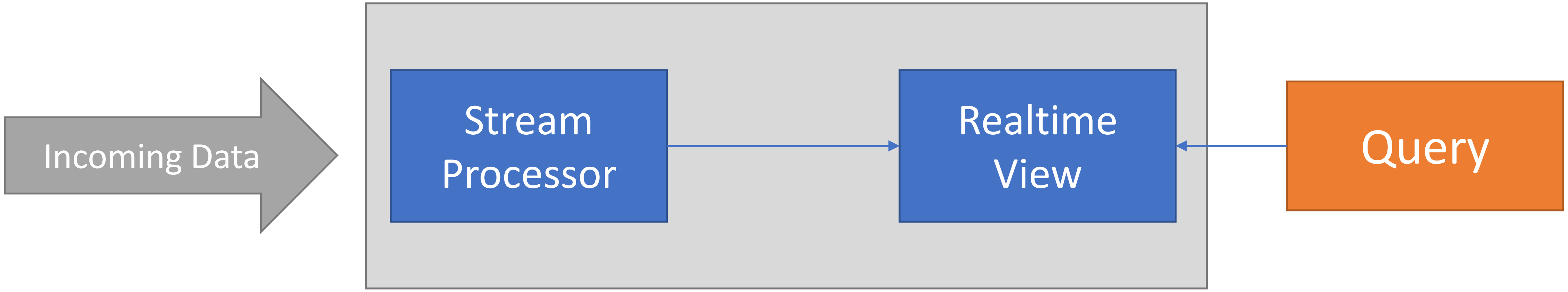
The Speed layer’s objective is to compensate for the last few hours of data

Lambda Architecture: Speed Layer



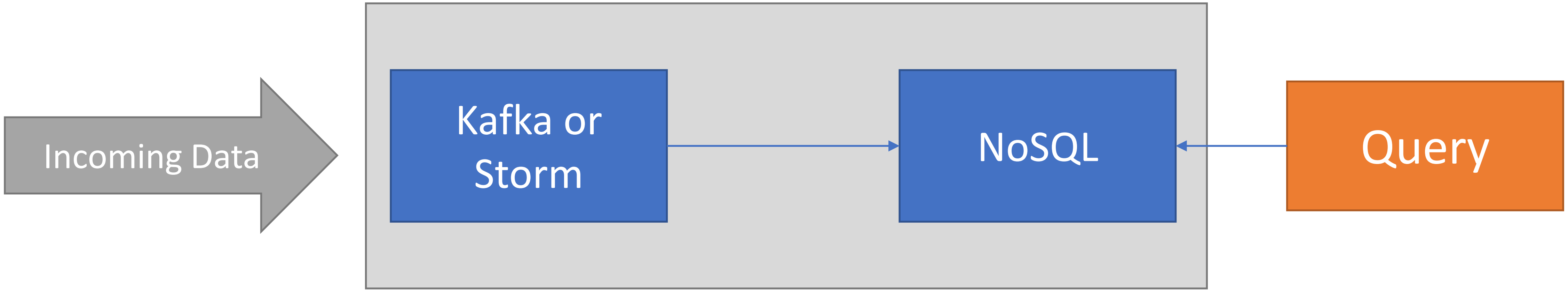
Lambda Architecture: Speed Layer

- Two main tasks for the Speed Layer:
  - Continuously process incoming data stream
  - Store the results of the real-time views
- Stores a limited window data (to compensate data not considered in Batch layer)
- If anything goes wrong, it is autocorrected in the next Batch View



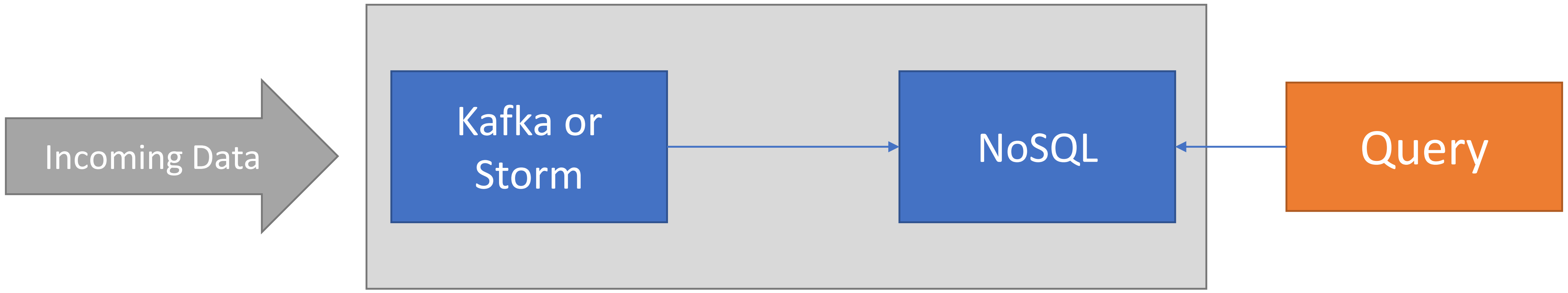
Lambda Architecture: Speed Layer

- Stream Processing
  - PubSub Architecture is usually fine
  - Can retain (store) data for a relatively short period
  - Realtime processing capability of data as it comes
  - Incremental algorithms



Lambda Architecture: Speed Layer

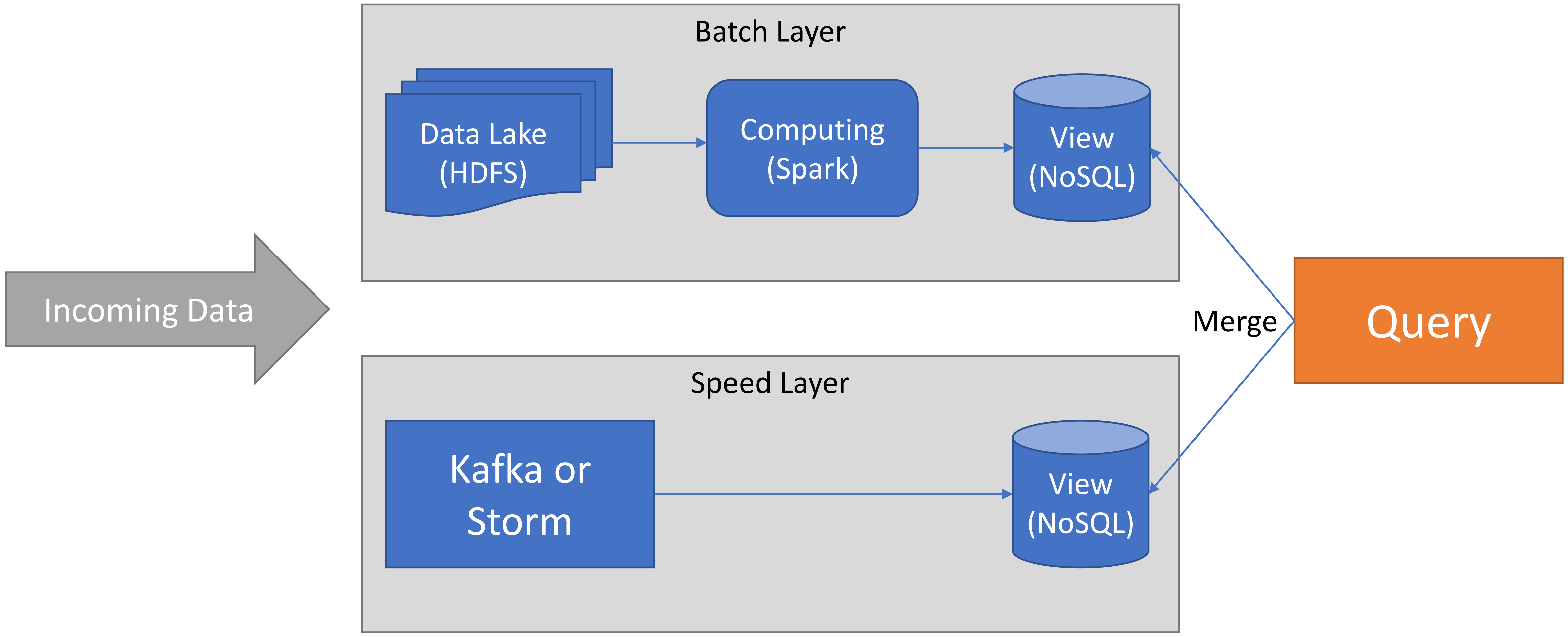
- Realtime views
    - Stored in random write database (incremental calculations)
    - Should support random read (to serve queries)
    - Performance should account large amount of random read & write operations



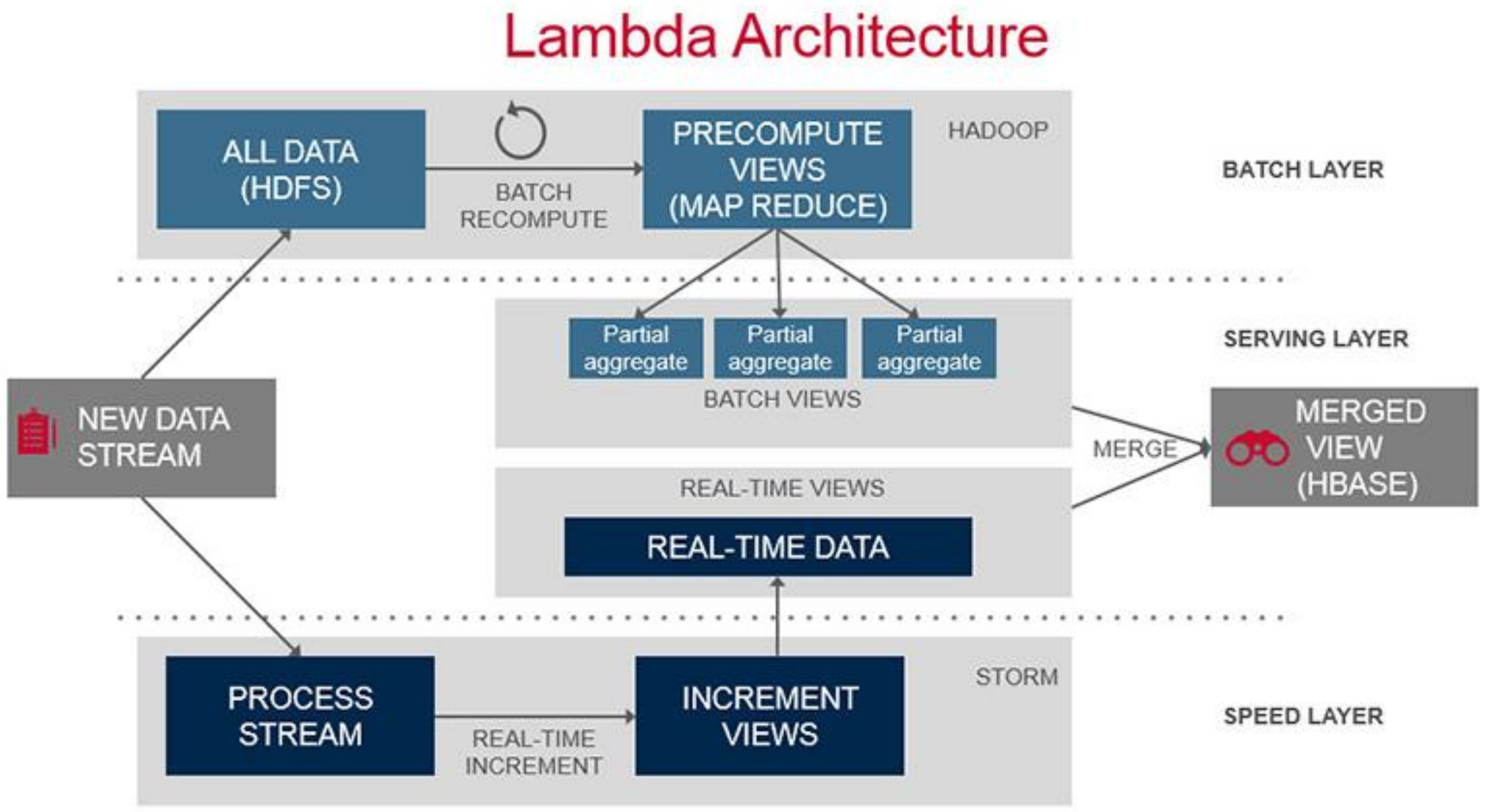
Serving Layer

Queries the Batch and Speed layers and merges their results

Lambda Architecture: Serving Layer



Lambda Architecture



https://mapr.com/developercentral/lambda-architecture/

Lambda Architecture: Advantages

- Data is immutable
  - Simplifies reasoning about data (no updates, no deletes)
  - Modelling data transformations from source data
  - Transformations are idempotent, repeat them any number of times leads to the same result
- Complexity isolated in the Realtime layer
  - Batch layer corrects any coherence issues in Realtime views
- Fault-tolerance (Human & System)
  - Discard views (Batch and Realtime) and recompute everything
  - Takes into account reprocessing data

Lambda Architecture: Cons

- Two different layers to maintain
  - 2 codes one for Batch Layer one for Speed Layer
  - They need to produce the same result
  - More technologies to deal with
  - Merging views is not easy and brings additional complexity of maintaining potentially 2 databases for the serving layer
- Two different and diverging programming paradigms to build a Lambda architecture.
- Requires multiple competences to put it to work.

KAPPA Architecture

“The Lambda Architecture has its merits, but alternatives are worth exploring.” (Jey Kreps)

Jey Kreps

- Ex- LinkedIn
- Co-Creator of Apache Kafka
- CEO of Confluent

https://www.oreilly.com/ideas/questioning-the-lambda-architecture

Kappa Architecture: Why

- Why can't the stream processing system be improved to handle the full problem of maintaining two systems in Lambda Architecture?

- If you have a sufficiently powerful stream processing system you can do most of the view building (transformation & computing) directly in the stream processing system.

➢Simplify the Lambda Architecture by removing the batch layer.

Kappa Architecture: Properties

- The datastore is retained in an append only Log (Write Ahead Log)
  - Immutable data similar to Lambda Architecture
  - Stores data in the stream system as a Log instead of an object store
- Processing results are stored in real-time views (usually NoSQL)
  - Processing data is idempotent as in Lambda Architecture
- Reprocessing can be done by consuming the entire history of data
- Fault Tolerant: throw the view are recompute the views!

Kappa Architecture: How (1/2)

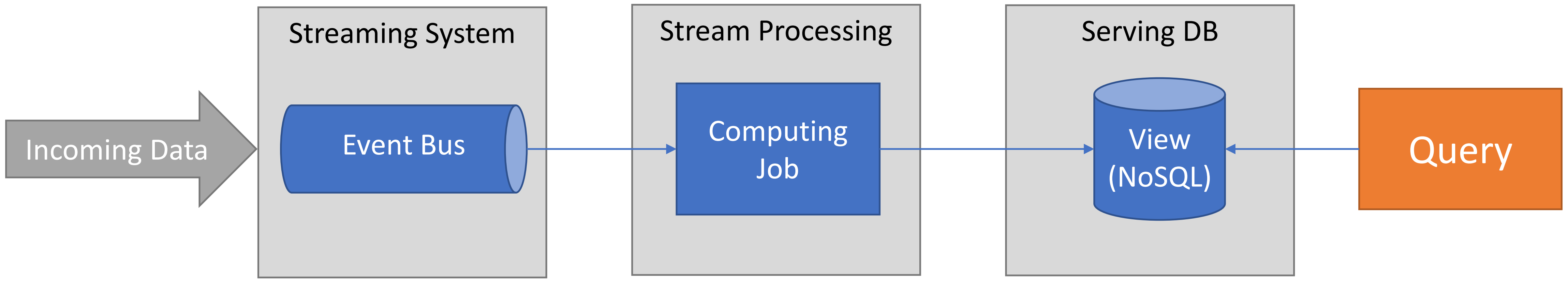
- Use a Stream system as Apache Kafka
  - This is a publish subscribe paradigm
  - System should allow multiple subscribers
- Retain data for as long as needed
  - example 30 days, 1 year, … This depends on the application
  - Data can be retained with unlimited history as well
    - Directly in the stream system (Kafka can handle Peta Bytes) or in an massively scalable object store like HDFS

Kappa Architecture: How (2/2)

- Start a computing job by subscribing to the data topic
- Write the results in a view (usually NoSQL)
  - Random read, Random write database
- Reprocessing can simply be done:
  - start a new instance of the stream processing
  - Point to the start of the retained data
  - Store results in a new destination
  - When the result is ready, switch from old job to new job

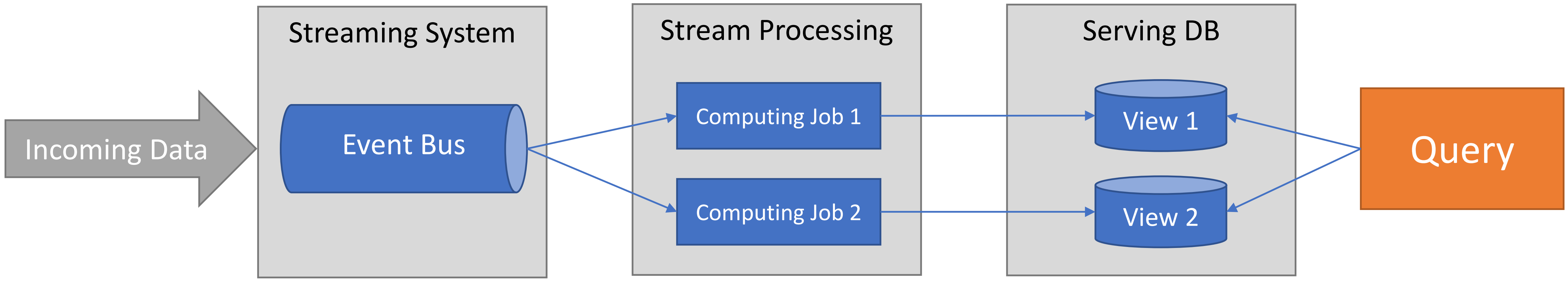
Kappa Architecture

Kappa architecture high level conceptual architecture



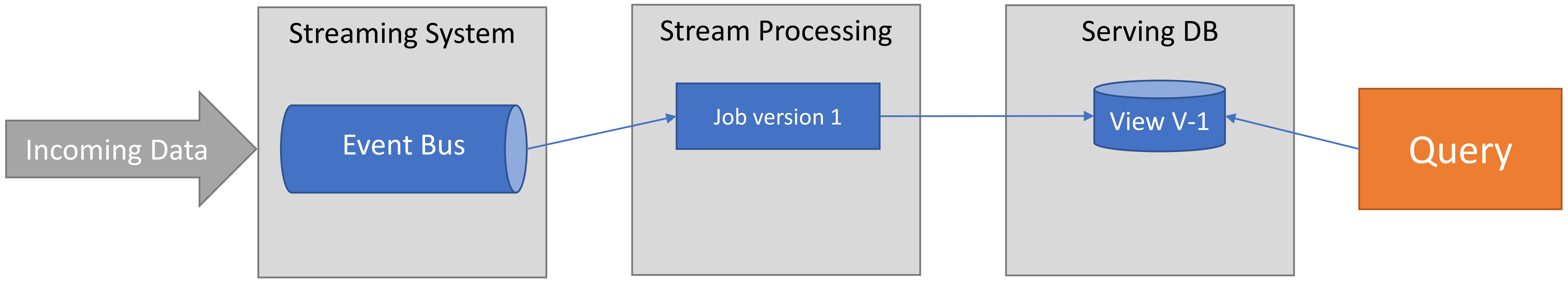
Kappa Architecture

One data stream – multiple processing jobs



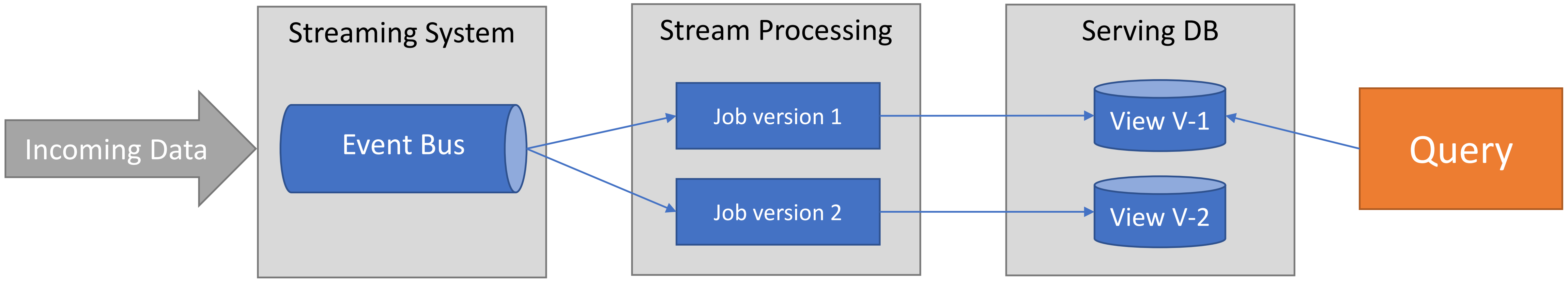
Kappa Architecture

How reprocessing works?



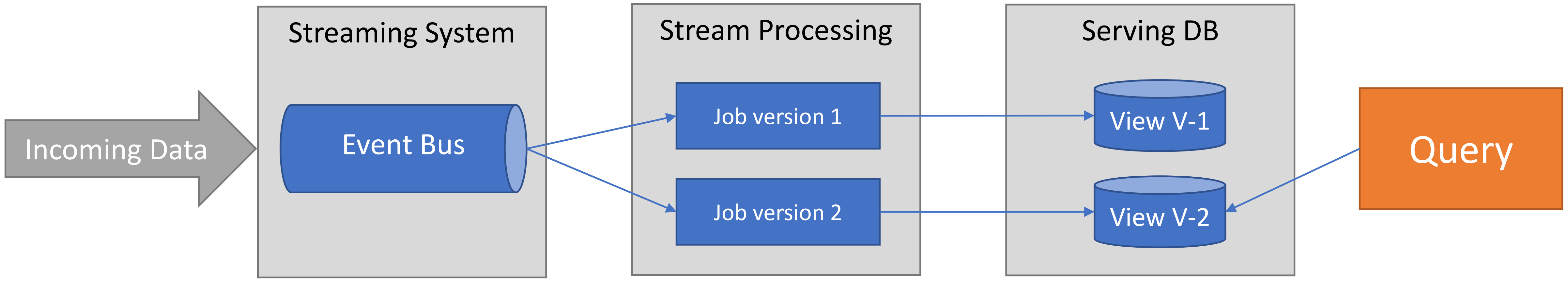
Kappa Architecture

Reprocessing data gets very simple: change your code and run it against retained data



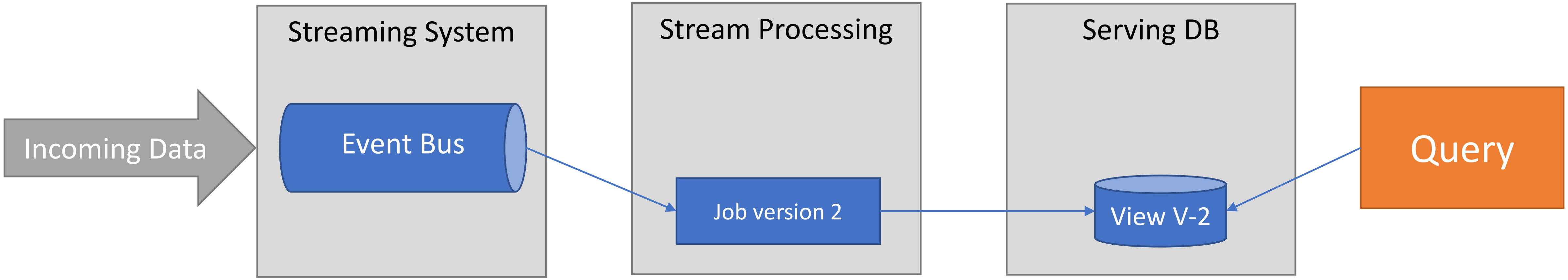
Kappa Architecture

Reprocessing data gets very simple: change your code and run it against retained data
When the second job is ready, read from new table (view V-2)



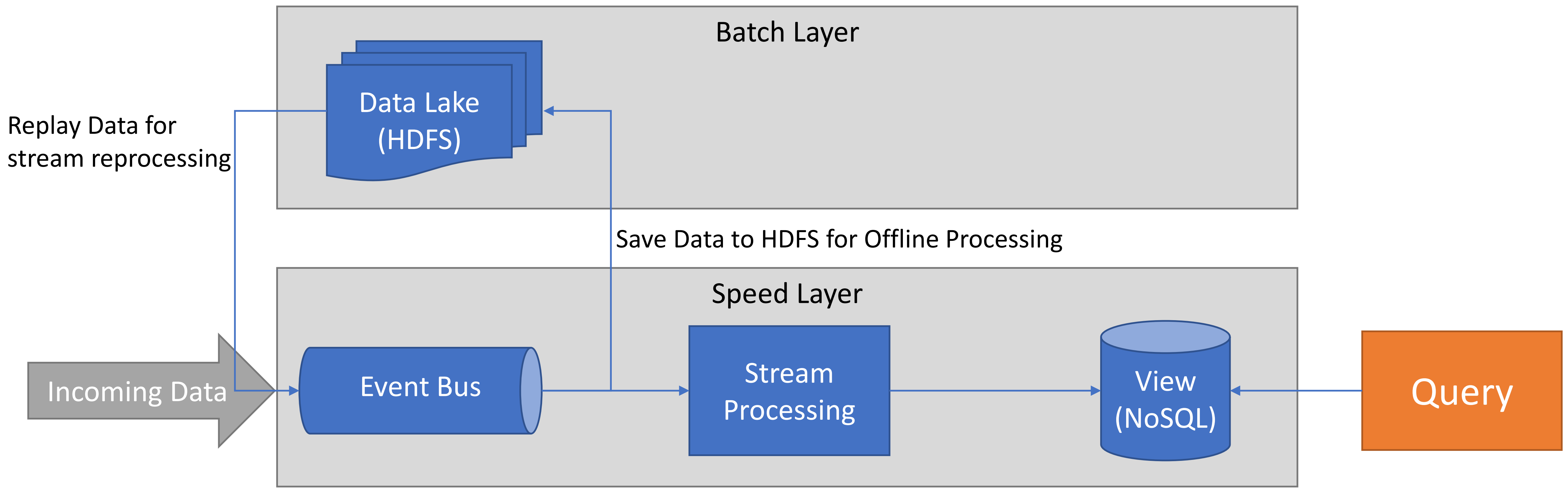
Kappa Architecture

Reprocessing data gets very simple: change your code and run it against retained data
When the second job is ready, read from new table (view V-2)
Stop the old version and delete its results



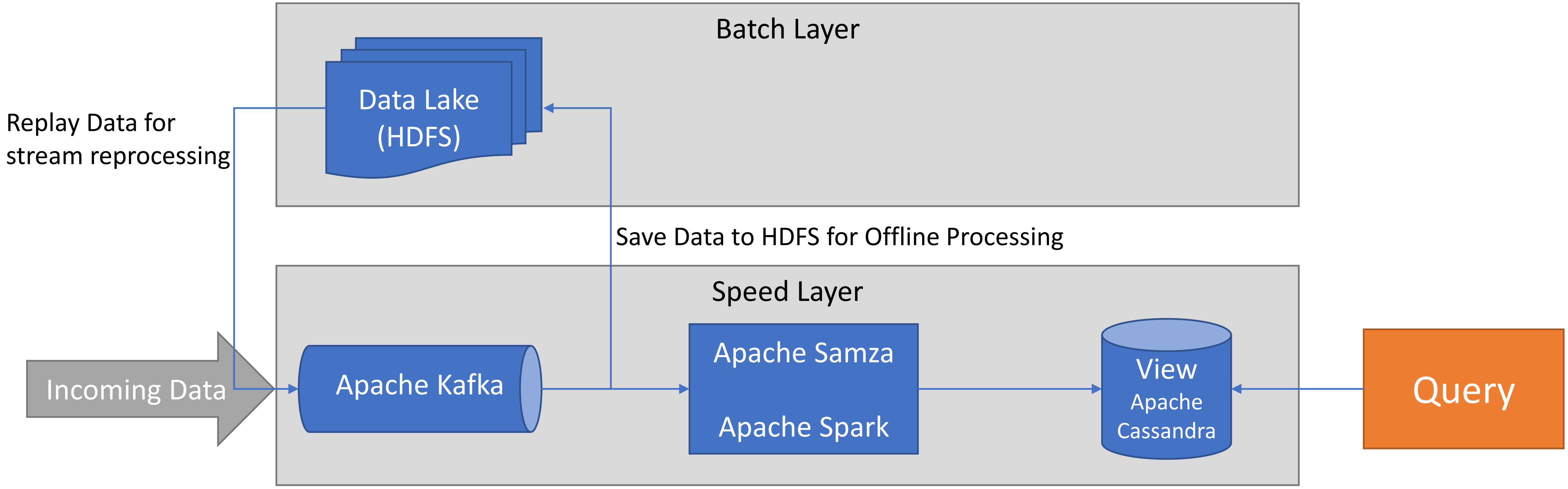
Kappa Architecture

Kappa architecture as a simplified view of Lambda architecture



Kappa Architecture

Kappa architecture as a simplified view of Lambda architecture



Kappa Architecture: Pros

- Simplicity: single processing platform
  - Only one paradigm and one code to maintain
- Latency: well adapted for real-time and time sensitive applications
- Configurable data retention and possibility to mirror data to HDFS to keep a full history
  - Data in both cases is immutable
- No periodic reprocessing as in Lambda
  - Reprocess data only when the code changes
- Run in parallel multiple versions of the processing job
  - Good for A/B testing

Kappa Architecture: Cons

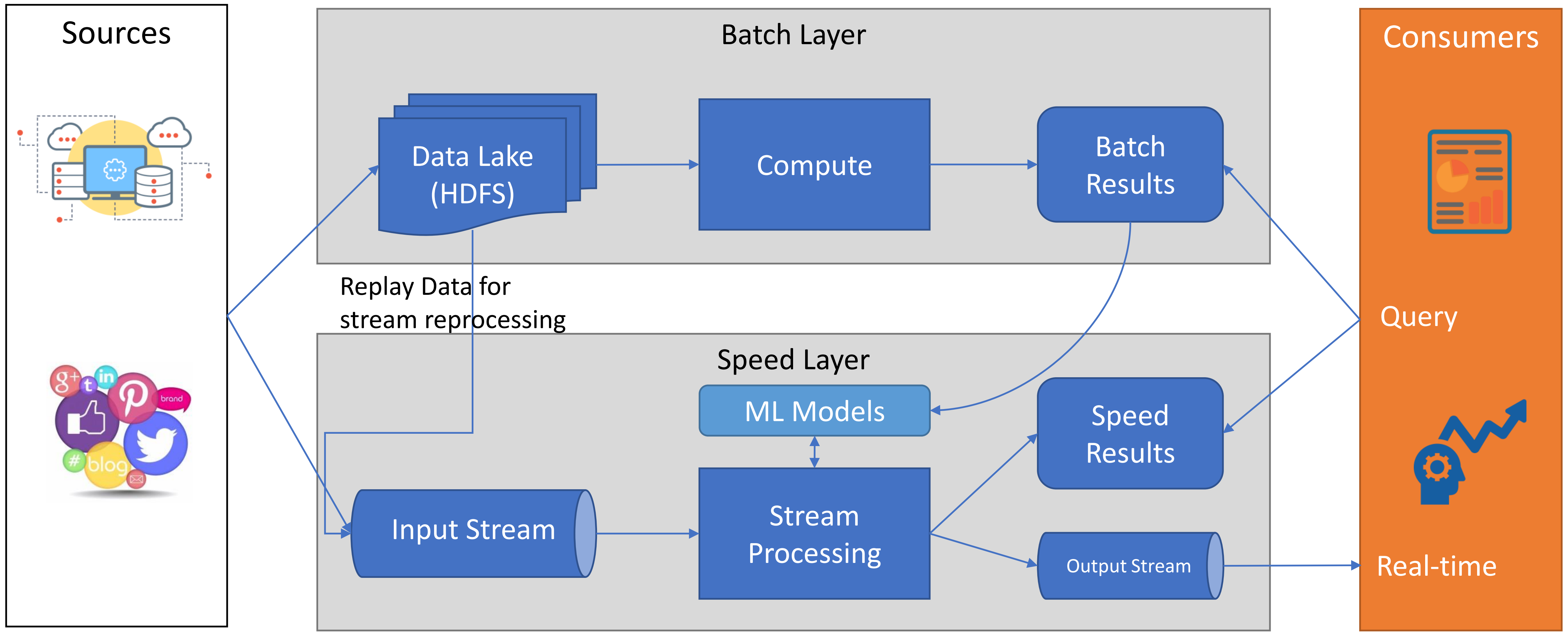
- Effort to adapt processing for real time

- Not a silver bullet for every Big Data project
    - Not meant for machine learning projects
    - More adapted for event based and streaming projects

Unified Architecture

Combines the advantages of both Lambda and Kappa architectures

Unified Architecture



Data Systems are Complex

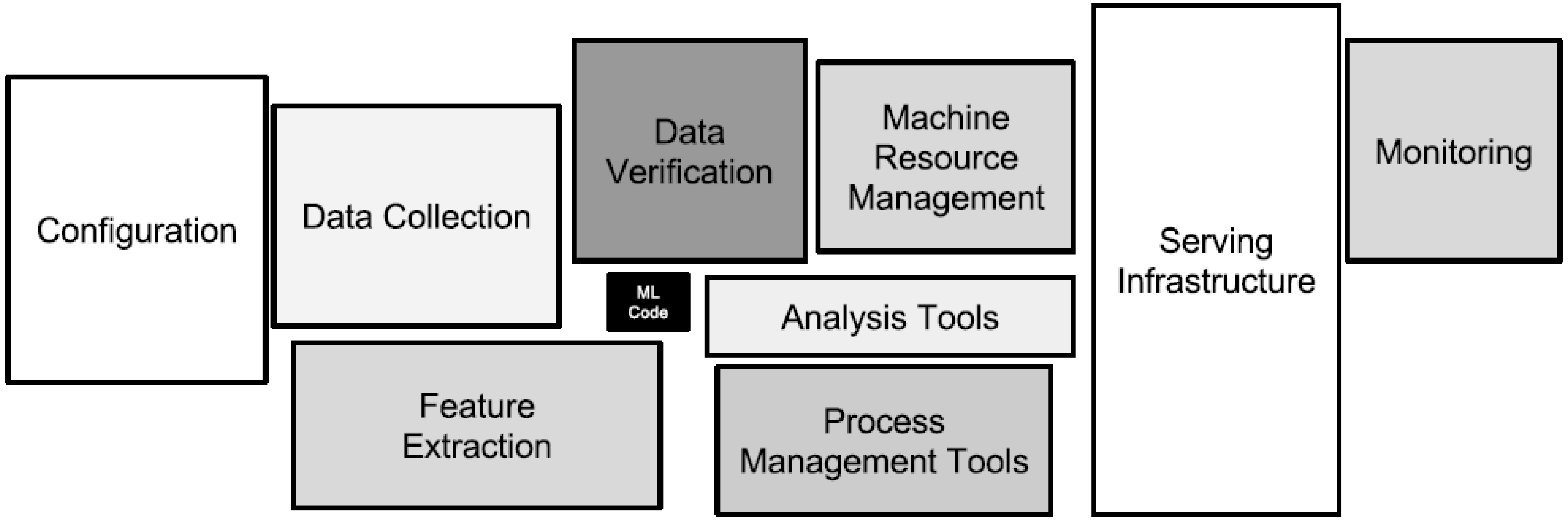


Figure 1: Only a small fraction of real-world ML systems is composed of the ML code, as shown by the small black box in the middle. The required surrounding infrastructure is vast and complex.

Big Data is hard, really hard!

"Hidden Technical Debt in Machine Learning Systems", Google NIPS 2015

References

- http://milinda.pathirage.org/kappa-architecture.com/
- https://www.youtube.com/watch?v=fU9hR3kiOK0
- https://www.youtube.com/watch?v=aJuo_bLSW6s&feature=youtu.be
- https://www.oreilly.com/ideas/questioning-the-lambda-architecture
- https://engineering.linkedin.com/distributed-systems/log-what-every-software-engineer-should-know-about-real-time-datas-unifying
- https://ferd.ca/beating-the-cap-theorem-checklist.html
- https://martinfowler.com/eaaDev/EventSourcing.html
- https://martinfowler.com/bliki/CQRS.html
- http://lambda-architecture.net/stories/
- http://nathanmarz.com/blog/how-to-beat-the-cap-theorem.html
- https://codahale.com/you-cant-sacrifice-partition-tolerance/
- https://www.voltdb.com/blog/2010/10/21/clarifications-cap-theorem-data-related-errors/
- http://blog.cloudera.com/blog/2010/04/cap-confusion-problems-with-partition-tolerance/
- https://docs.microsoft.com/en-us/azure/architecture/patterns/event-sourcing
- https://mapr.com/developercentral/lambda-architecture/
- http://cs-www.cs.yale.edu/homes/dna/papers/abadi-pacelc.pdf
- https://www.infoq.com/articles/cap-twelve-years-later-how-the-rules-have-changed

References

- https://databricks.com/blog/2015/04/21/analyzing-apache-access-logs-with-databricks-cloud.html
- https://www.oreilly.com/ideas/applying-the-kappa-architecture-in-the-telco-industry
- http://samza.apache.org/learn/documentation/0.7.0/jobs/reprocessing.html

References - Data

- https://www.ibm.com/developerworks/library/wa-introhdfs/
- https://cdn.oreillystatic.com/en/assets/1/event/160/Parquet%20performance%20tuning_%20The%20missing%20guide%20Presentation.pdf
- https://techmagie.wordpress.com/category/big-data/avro-parquet/
- https://www.dataengineeringpodcast.com/data-serialization-with-doug-cutting-and-julien-le-dem-episode-8/
- http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html#Data_Replication
- http://hadooptalk.com/awesome-introduction-of-parquet/
- https://akiselev87.wordpress.com/2016/10/26/spark-and-data-formats-introduction/