Big Data

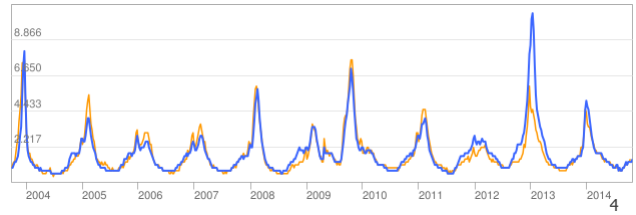
Introduction to Big Data

Big data

3

Google Flu trends

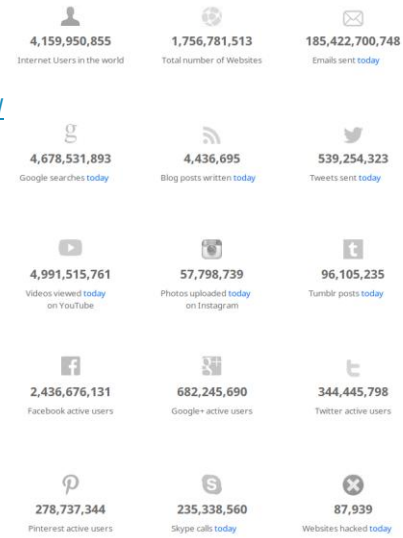
- February 2010
  - Google detected flu outbreak two weeks ahead of CDC data (Centers for Disease Control and Prevention – U.S.A)
  - Based on the analysis of Google search queries



4

Data on the Internet…

- Internet live stats
  - http://www.internetlivestats.com/



Who generates big data?

- User Generated Content (Web & Mobile)
  - E.g., Facebook, Instagram, Yelp, TripAdvisor, Twitter, YouTube

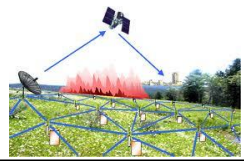
- Health and scientific computing

6

Who generates big data?

- Log files
  - Web server log files, machine system log files

- Internet Of Things (IoT)
  - Sensor networks, RFID, smart meters



7

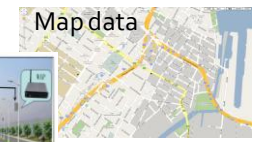
An example of Big data at work

- Crowdsourcing

Computing

Sensing

Map data



Real time traffic info

8

What is big data?

- Many different definitions
  - "Data whose scale, diversity and complexity require new architectures, techniques, algorithms and analytics to manage it and extract value and hidden knowledge from it"

9

What is big data?

- Many different definitions
  - "Data whose **scale**, **diversity** and **complexity** require new architectures, techniques, algorithms and analytics to manage it and extract value and hidden knowledge from it"

10

What is big data?

- Many different definitions
  - "Data whose scale, diversity and complexity require new **architectures**, **techniques**, **algorithms** and **analytics** to manage it and extract value and hidden knowledge from it"

11

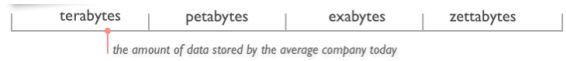
The Vs of big data

- The 3Vs of big data
  - **V**olume: scale of data
  - **V**ariety: different forms of data
  - **V**elocity: analysis of streaming data
- … but also
  - **V**eracity: uncertainty of data
  - **V**alue: exploit information provided by data

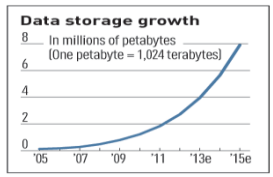
12

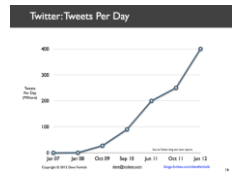
The Vs of big data

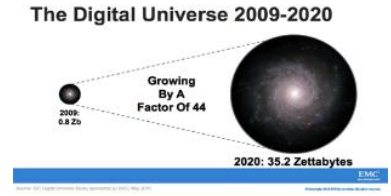
- Volume



  - Data volume increases exponentially over time
  - 44x increase from 2009 to 2020
    - Digital data 35 ZB in 2020



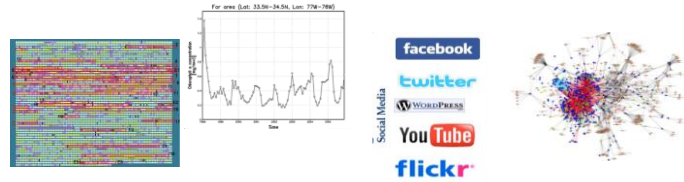




13

The Vs of big data

- Variety
  - Various formats, types and structures
    - Numerical data, image data, audio, video, text, time series



  - A single application may generate many different formats
    - Heterogeneous data
    - Complex data integration problem

14

The Vs of big data

- Velocity
  - Fast data generation rate
    - Streaming data
  - Very fast data processing to ensure timeliness

15

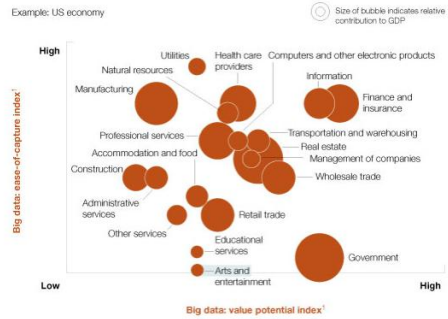
The Vs of big data

- Veracity
  - Data quality

16

The Vs of big data

- Value
  - Translate data into business advantage



[1]For detailed explication of metrics, see appendix in McKinsey Global Institute full report *Big data: The next frontier for innovation, competition, and productivity*, available free of charge online at mckinsey.com/mgi.
Source: US Bureau of Labor Statistics; McKinsey Global Institute analysis

17

Big data value chain

Generation → Acquisition → Storage → Analysis

- Generation
  - Passive recording
    - Typically structured data
    - Bank trading transactions, shopping records, government sector archives
  - Active generation
    - Semistructured or unstructured data
    - User-generated content, e.g., social networks
  - Automatic production
    - Location-aware, context-dependent, highly mobile data
    - Sensor-based Internet-enabled devices

18

Big data value chain

- Acquisition
  - Collection
    - Pull-based, e.g., web crawler
    - Push-based, e.g., video surveillance, click stream
  - Transmission
    - Transfer to data center over high capacity links
  - Preprocessing
    - Integration, cleaning, redundancy elimination

19

Big data value chain

- Storage
  - Storage infrastructure
    - Storage technology, e.g., HDD, SSD
    - Networking architecture, e.g., DAS, NAS, SAN
  - Data management
    - File systems (HDFS), key-value stores (Memcached), column-oriented databases (Cassandra), document databases (MongoDB)
  - Programming models
    - Map reduce, stream processing, graph processing

20

Big data value chain

- Analysis
  - Objectives
    - Descriptive analytics, predictive analytics, prescriptive analytics
  - Methods
    - Statistical analysis, data mining, text mining, network and graph data mining
    - Clustering, classification and regression, association analysis
  - Diverse domains call for customized techniques

21

Big data challenges

- Technology and infrastructure
  - New architectures, programming paradigms and techniques are needed
- Data management and analysis
  - New emphasis on "data"
  - ➡ **Data science**

22

Comparison with "traditional" data processing and distributed programming

Large scale data processing

- Traditional approach
  - Database and data warehousing systems
  - Well-defined structure
  - Small enough data
- Big data
  - Data sets not suitable for (traditional relational) databases
    - E.g., Internet data crawled by Google, Yahoo!, Facebook, …
  - May need near real-time (streaming) analysis
    - Different from data warehousing
  - Different programming paradigms

24

Large scale data processing

- Traditional computation is **processor bound**
  - Small dataset
  - Complex processing
- How to increase performance?
  - New and faster processor
  - More RAM

25

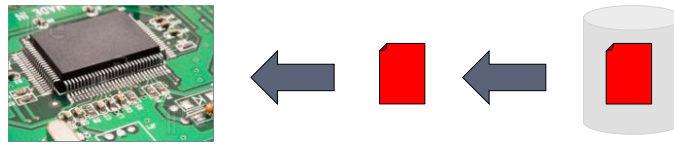
Large scale data processing

- Traditional data storage
  - On large SANs
  - Data transferred to processing nodes on demand at computing time
- Traditional distributed computing
  - Multiple machines, single job
  - Complex systems
    - E.g., MPI
  - Programmers need to manage data transfer synchronization, system failure, dependencies

26

The bottleneck

- Processors process data
- Hard drives store data
- We need to transfer data from the disk to the processor



27

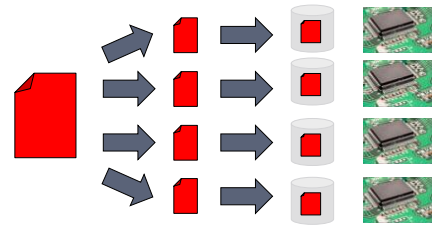
The bottleneck

- Hard drives evolution
  - Storage capacity increased fast in recent decades
    - E.g., from 1GB to 1TB
  - The transfer rate increased less
    - E.g., from 5MB/s to 100MB/s
- Transfer of disk content in memory
  - Few years ago: 3.33 min.
  - Now: 2.7 hours (if you have enough RAM)
- Problem: **data transfer from disk to processors**

28

The solution

- Transfer the processing power to the data
- Multiple distributed disks
  - Each one holding a portion of a large dataset
- Process in parallel different file portions from different disks



29

Issues

- Need to manage
  - Process synchronization
  - Hardware failures
  - Data loss
  - Joining data from different disks
  - Scalability
- Managed by new distributed architectures
  - E.g., Hadoop

30

Apache Hadoop

- Open source project by the Apache Foundation
- Based on 2 Google papers
  - Google File System (GFS), published in 2003
  - Map Reduce, published in 2004
- Reliable storage and processing system based on YARN (Yet Another Resource Negotiator)
  - Storage provided by HDFS
    - Distributed file system
  - Different processing models
    - E.g., MapReduce, Spark-RDDs, Spark streaming, Hive, Giraph

31

Motivations of Hadoop-like frameworks

Data volumes

- The amount of data increases every day
- Some numbers (~ 2012):
  - Data processed by Google every day: 100+ PB
  - Data processed by Facebook every day: 10+ PB
- To analyze them, systems that scale with respect to the data volume are needed

33

Data volumes: Google Example

- Analyze 10 billion web pages
- Average size of a webpage: 20KB
- Size of the collection: 10 billion x 20KB = 200TB
- Hard disk read bandwidth: 100MB/sec
- Time needed to read all web pages (without analyzing them): 2 million seconds = more than **24 days**
- A single node architecture is not adequate

34

Failures

- Failures are part of everyday life, especially in data center
  - A single server stays up for 3 years (~1000 days)
    - 10 servers → 1 failure every 100 days (~3 months)
    - 100 servers → 1 failure every 10 days
    - 1000 servers → 1 failure/day
- Sources of failures
  - Hardware/Software
  - Electrical, Cooling, …
  - Unavailability of a resource due to overload

35

Failures

- LALN data [DSN 2006]
  - Data for 5000 machines, for 9 years
  - Hardware failures: 60%, Software: 20%, Network 5%
- DRAM error analysis [Sigmetrics 2009]
  - Data for 2.5 years
  - 8% of DIMMs affected by errors
- Disk drive failure analysis [FAST 2007]
  - Utilization and temperature major causes of failures

36

Failures

- Failure types
  - Permanent
    - E.g., Broken motherboard
  - Transient
    - E.g., Unavailability of a resource due to overload

37

Network bandwidth

- Network becomes the bottleneck if big amounts of data need to be exchanged between nodes/servers
  - Network bandwidth: 1Gbps
  - Moving 10 TB from one server to another takes 1 day
    → Data should be moved across nodes only when it is indispensable
  - Usually, codes/programs are small (few MBs)
    → Move code/program and computation to data

38

Network bandwidth

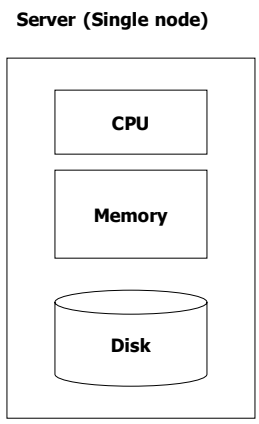
- Network becomes the bottleneck if big amounts of data need to be exchanged between nodes/servers
  - Network bandwidth: 1Gbps
  - Moving 10 TB from one server to another takes 1 day
    → Data should be moved across nodes only when it is indispensable
  - Usually, codes/programs are small (few MBs)
    → Move code/program and computation to data

⇩

Data locality

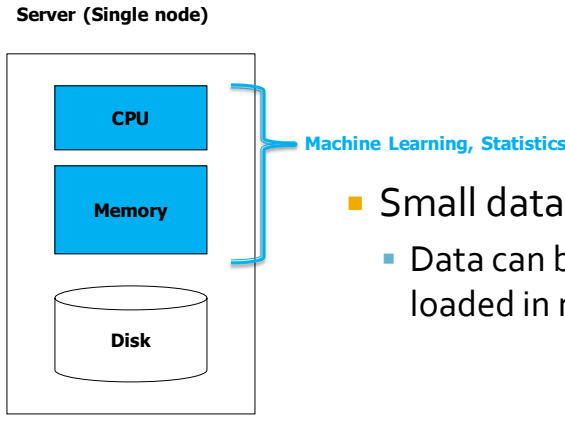
39

Single-node architecture



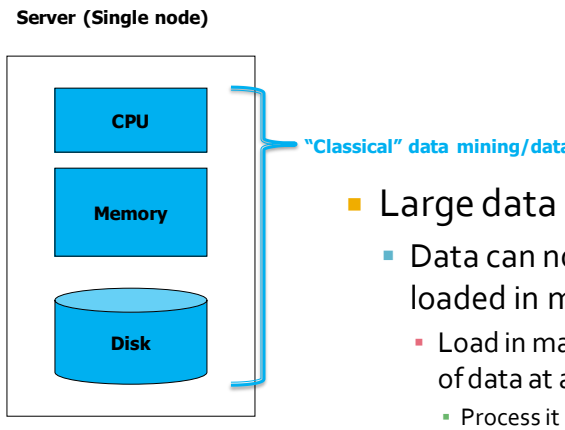
40

Single-node architecture



- Small data
  - Data can be completely loaded in main memory

Single-node architecture



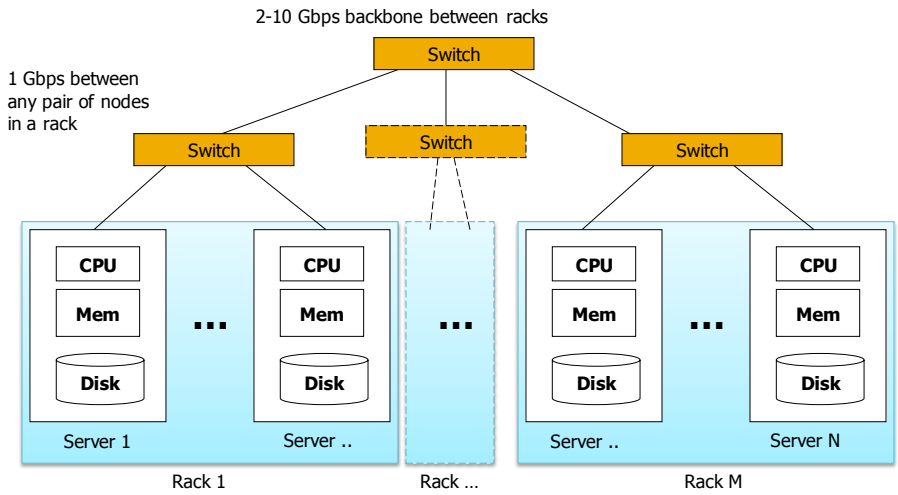
- Large data
  - Data can not be completely loaded in main memory
    - Load in main memory one chunk of data at a time
      - Process it and store some statistics
    - Combine statistics to compute the final result

Cluster Architecture

- Cluster of servers (data center)
  - Computation is distributed across servers
  - Data are stored/distributed across servers
- Standard architecture in the Big data context
  - Cluster of commodity Linux nodes/servers
    - 32 GB of main memory per node
  - Gigabit Ethernet interconnection

43

Commodity Cluster Architecture



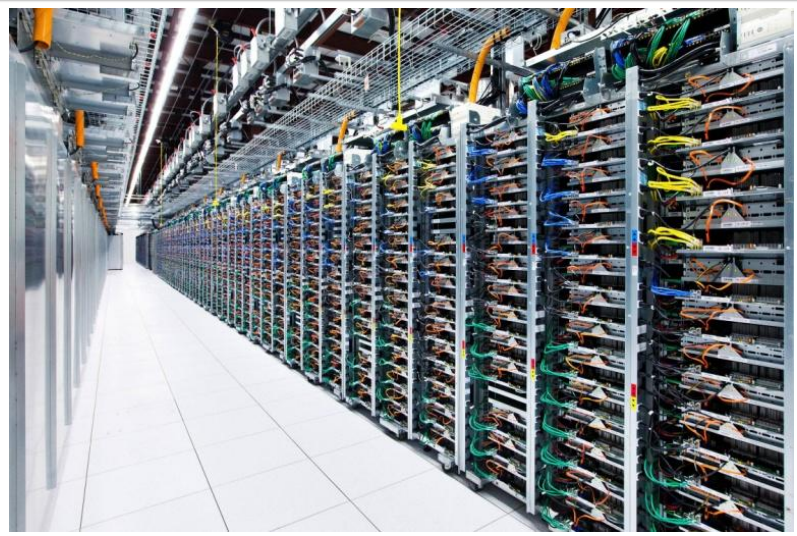
Each rack contains 16-64 nodes

44

Data center

45

Data center



46

Scalability

- Current systems must scale to address
  - The increasing amount of data to analyze
  - The increasing number of users to serve
  - The increasing complexity of the problems
- Two approaches are usually used to address scalability issues
  - Vertical scalability (scale up)
  - Horizontal scalability (scale out)

47

Scale up vs. Scale out

- Vertical scalability (scale up)
  - Add more power/resources (main memory, CPUs) to a single node (high-performing server)
    - Cost of super-computers is not linear with respect to their resources
- Horizontal scalability (scale out)
  - Add more nodes (commodity servers) to a system
    - The cost scales approximately linearly with respect to the number of added nodes
    - But data center efficiency is a difficult problem to solve

48

Scale up vs. Scale out

- For data-intensive workloads, a large number of commodity servers is preferred over a small number of high-performing servers
  - At the same cost, we can deploy a system that processes data more efficiently and is more fault-tolerant
- Horizontal scalability (scale out) is preferred for big data applications
  - But distributed computing is hard

    →New systems hiding the complexity of the distributed part of the problem to developers are needed

49

Cluster computing challenges

- Distributed programming is hard
  - Problem decomposition and parallelization
  - Task synchronization
- Task scheduling of distributed applications is critical
  - Assign tasks to nodes by trying to
    - Speed up the execution of the application
    - Exploit (almost) all the available resources
    - Reduce the impact of node failures

50

Cluster computing challenges

- Distributed data storage
  - How do we store data persistently on disk and keep it available if nodes can fail?
    - Redundancy is the solution, but it increases the complexity of the system
- Network bottleneck
  - Reduce the amount of data send through the network
    - Move computation (and code) to data

51

Cluster computing challenges

- Distributed computing is not a new topic
  - HPC (High-performance computing) ~1960
  - Grid computing ~1990
  - Distributed databases ~1990
- Hence, many solutions to the mentioned challenges are already available
- But we are now facing big data driven-problems

  → The former solutions are not adequate to address big data volumes

52

Typical Big Data Problem

- Typical Big Data Problem
  - Iterate over a large number of records/objects
  - Extract something of interest from each
  - Aggregate intermediate results
  - Generate final output
- The challenges:
  - Parallelization
  - Distributed storage of large data sets (Terabytes, Petabytes)
  - Node Failure management
  - Network bottleneck
  - Diverse input format (data diversity & heterogeneity)

53

Lambda architecture

Some definitions: v1

- *Source: Nathan Marz*
  "The past decade has seen a huge amount of innovation in scalable data systems. These include large-scale computation systems like **Hadoop** and **databases such as Cassandra** and Riak. These systems can handle very large amounts of data, but with serious trade-offs.

55

Some definitions: v1

Hadoop, for example, can **parallelize large-scale batch computations** on very large amounts of data, but the computations have **high latency**. You don't use Hadoop for anything where you need low-latency results.

56

Some definitions: v1

NoSQL databases like Cassandra achieve their **scalability** by offering you a **much more limited data model** than you're used to with something like SQL. Squeezing your application into these limited data models can be very complex. And because the databases are mutable, they're not human-fault tolerant.

57

Some definitions: v1

These tools on their own are not a panacea. But when **intelligently used in conjunction with one another,** you **can produce scalable systems for arbitrary data problems** with human-fault tolerance and a minimum of complexity. This is the **Lambda Architecture** you'll learn throughout the book. "

58

Some definitions: v2

- *Source: Databricks*
  "**Lambda architecture** is a way of processing massive quantities of data (i.e. "**Big Data**") that provides access to **batch-processing** and **stream-processing** methods with a hybrid approach.

  Lambda architecture is used to solve the problem of **computing arbitrary functions**."

59

Lambda architecture: requirements

- Fault-tolerant against both hardware failures and human errors
- Support variety of use cases that include low latency querying as well as updates
- Linear scale-out capabilities
- Extensible, so that the system is manageable and can accommodate newer features easily

60

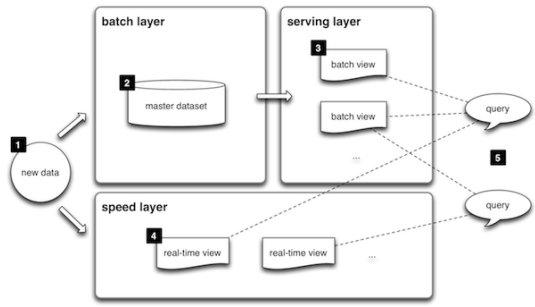
Lambda architecture: queries

query = function(all data)

- Latency: the time it takes to run a query
- Timeliness: how up to date the query results are (freshness and consistency)
- Accuracy: tradeoff between performance and scalability (approximations)

61

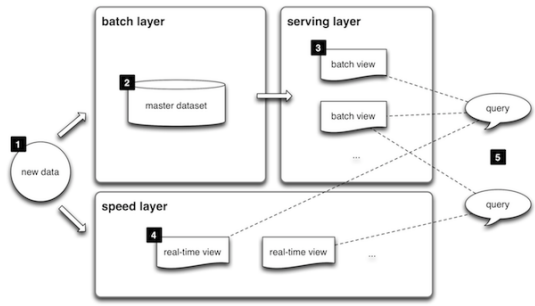
Lambda architecture



From http://lambda-architecture.net/

62

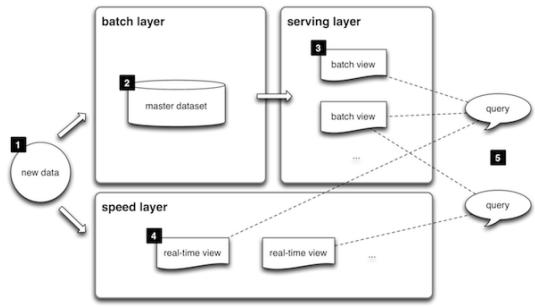
Lambda architecture



1. **All data** entering the system is **dispatched to both** the **batch layer** and the **speed layer** for processing

From http://lambda-architecture.net/ 63

Lambda architecture



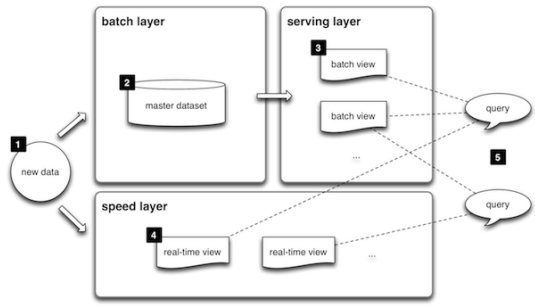
2. The batch layer has two functions:

   (i) managing the **master dataset** (an **immutable, append-only** set of **raw data**), and

   (ii) to **pre-compute** the **batch views**

From http://lambda-architecture.net/ 64

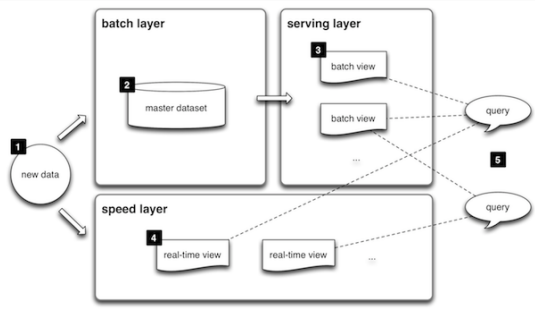
Lambda architecture



3. The serving layer **indexes** the batch views so that they can be **queried in low-latency,** ad-hoc way

From http://lambda-architecture.net/

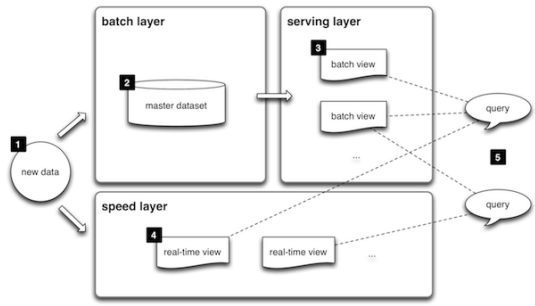
Lambda architecture



4. The speed layer compensates for the high latency of updates to the serving layer and **deals with recent data only**

From http://lambda-architecture.net/

Lambda architecture

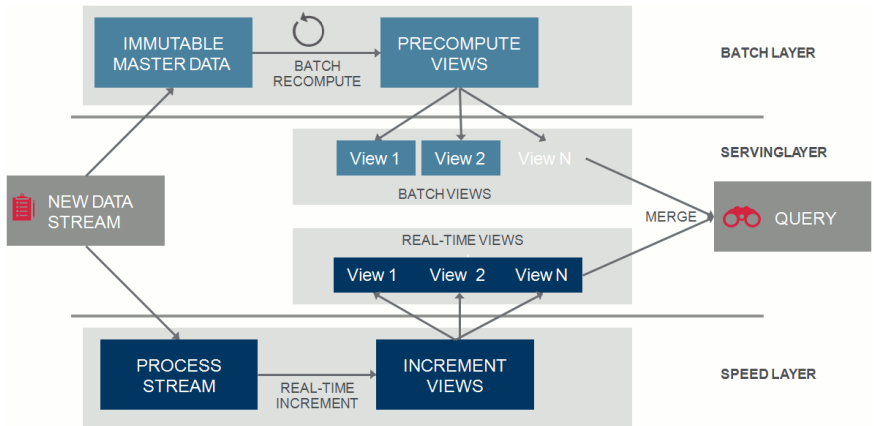


5. Any incoming **query** can be answered by **merging** results from **batch views** and **real-time views**

From http://lambda-architecture.net/

67

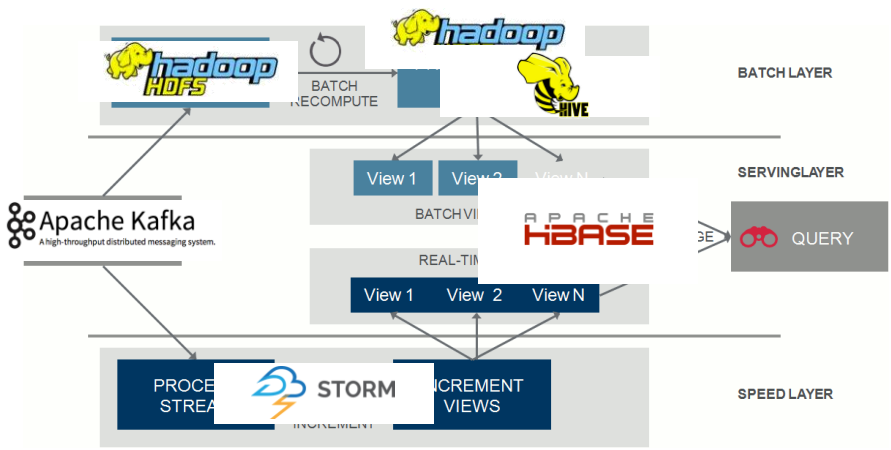
Lambda architecture v2



From https://speakerdeck.com/mhausenblas/lambda-architecture-with-apache-spark

68

Lambda architecture: possible implementation



From https://speakerdeck.com/mhausenblas/lambda-architecture-with-apache-spark

69