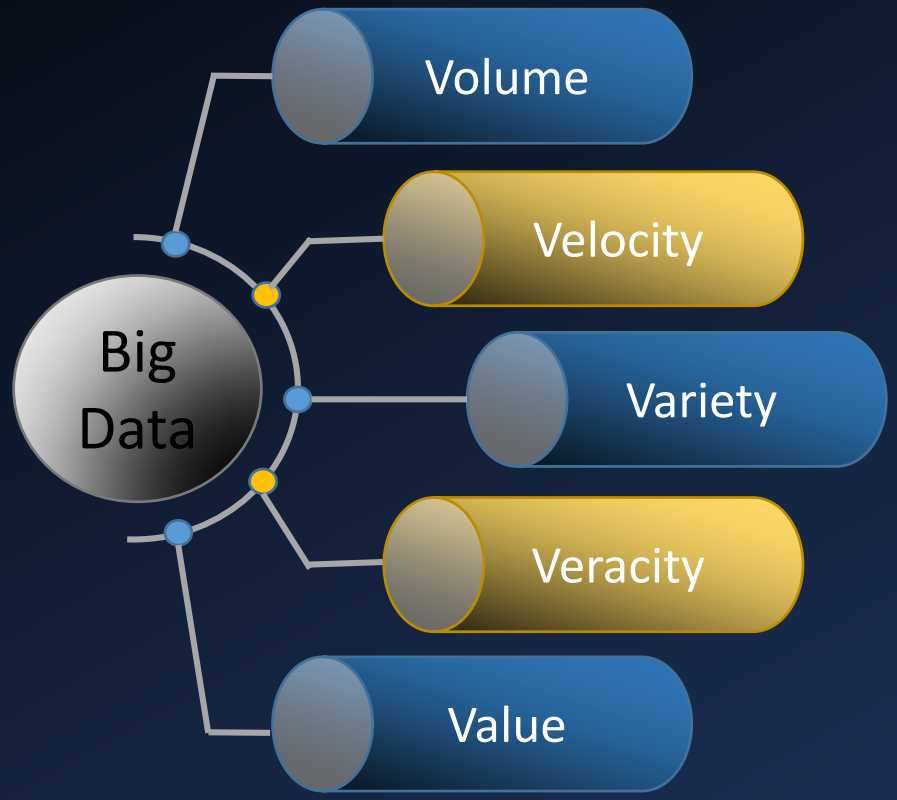
Big Data (MHI222956/MHI225101)

2.1 Big Data Architecture

Big Data Solutions



Big Data Challenges

- A Big Data solution addresses these challenges
- A type of IT system that combines the features and capabilities of several Big Data tools, or platforms, within a single solution, often in the form of a data pipeline
- Should have the capability to store the data, process the data, derive insights, and provide information in an acceptable time frame
- Should be scalable on relevant parameters, such as number of data sources/devices, messages, storage
- Scalability typically achieved by distributing storage and/or processing over many low powered computers (scaling out) rather than increasing the power of a single computer (scaling up)

Properties of distributed data store

- Consistency

  Strict consistency specifies that a data item written to one node will be seen instantaneously by all other nodes – in other words any read operation on any node will see the most recent value written to any node

- Availability

  Every request receives a (non-error) response -- without guarantee that it contains the newest information

- Partition-tolerance

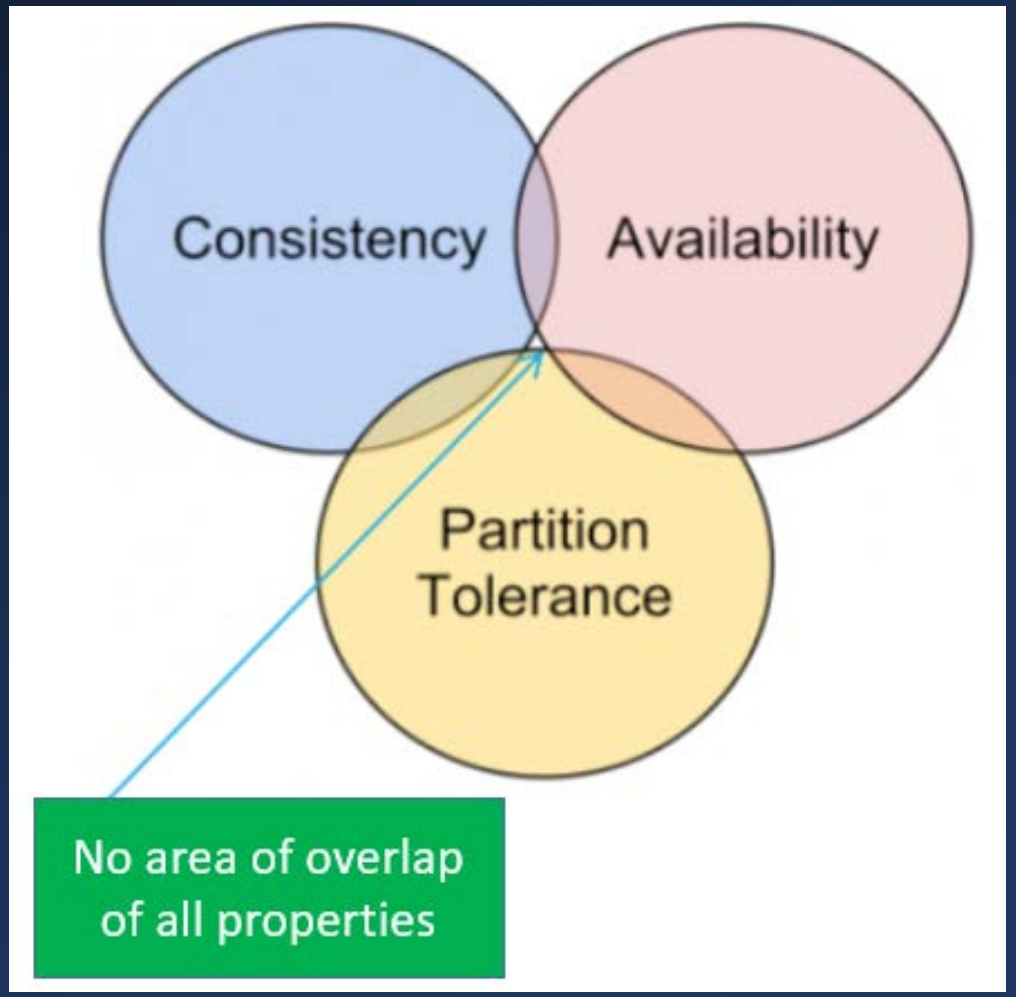
  The system continues to operate despite an arbitrary number of messages being dropped (or delayed) by the network between nodes

CAP theorem

- First described by Eric Brewer

(http://www.julianbrowne.com/article/brewers-captheorem)

- States that it is only possible to achieve 2 out of the three properties C, A, P



CAP theorem

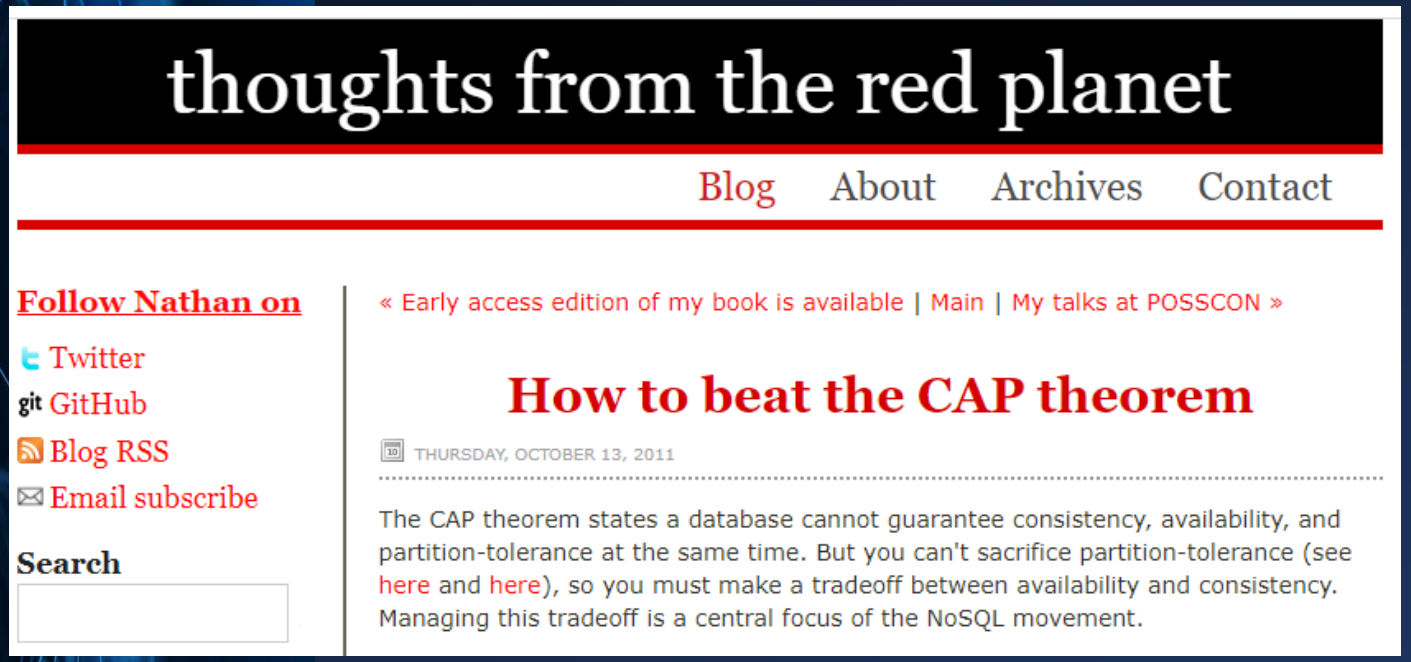
- For example, if a system is partition-tolerant, then it can deal with situations where messages don't get from one node to another. In this case a data item written to one node might not be able to be read from another node.

- If a system is partition-tolerant and available, then a node that you read from may not have the most up-to-date data so the system is not consistent.
- If a system is partition-tolerant and consistent, then it will only allow the most recent write to be read from any node, won't respond with a data item at all until the most recent one has been received at that node.
- If a system is to be available and consistent, then all nodes must be able to communicate instantaneously (or there must only be one node) so there can never be any partitions.

Architecture of Big Data solution

- The architecture of a Big Data solution may involve many components
- Optimum solution may not be a simple linear arrangement
- Software architecture is often described in terms of logical layers which offer a way to organize and reason about these components.
- The layers reflect components that perform specific functions within the overall architecture
- The layers are merely logical; they do not imply that the functions that support each layer are run on separate machines or separate processes
- A good example of this is the Lambda architecture

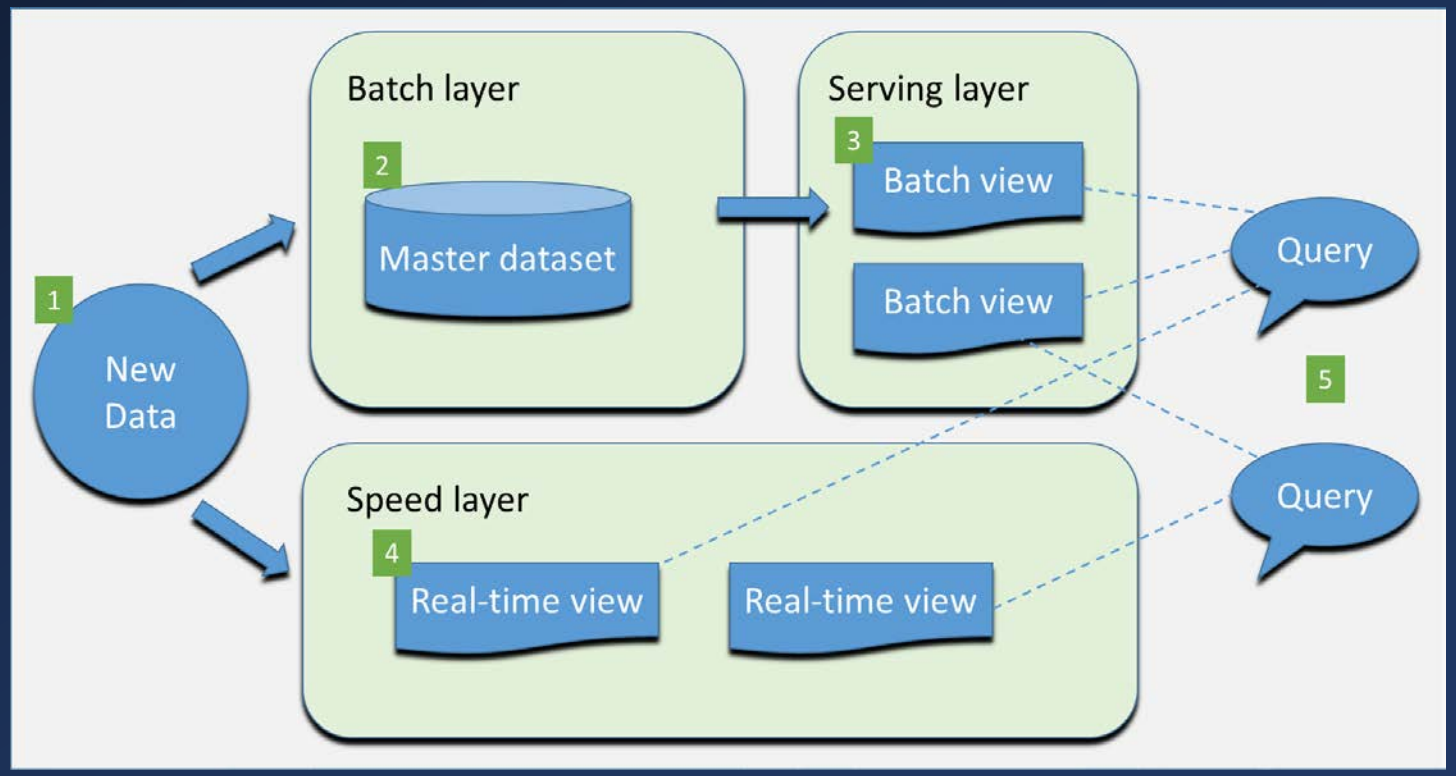
Lambda architecture



- Proposed by Nathan Marz in 2011
- Posted in his blog:

  http://nathanmarz.com/blog/how-to-beat-the-cap-theorem.html
- An important approach to tackling the limitations of the CAP theorem

1. All data entering the system is dispatched to both the batch layer and the speed layer for processing.
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.
3. The serving layer indexes the batch views so that they can be queried in low-latency, ad-hoc way.
4. The speed layer compensates for the high latency of updates to the serving layer and deals with recent data only.
5. Any incoming query can be answered by merging results from batch views and real-time views.



Lambda architecture

Batch and serving layers

- Scalable
  - Can be built with distributed batch processing platforms such as Hadoop, Spark, or distributed storage, can easily scale out
- Robust
  - These platforms handle failover when machines go down
- Human fault-tolerant
  - If incorrect data is introduced it won't overwrite existing data, can remove the corrupt data and recompute views from scratch

However…

- Low latency
  - NO, computation of server layer view may take significant time so view does not contain latest data

Speed layer

- Works with stream of incoming data
- Ensures that data since last batch layer recomputation started is available to query in real time
- Speed layer produces views based on recent data
- To reduce latency it works incrementally, updates realtime views as it receives new data rather than recomputing from scratch
- More complex than batch layer due to this incremental nature
- Once data makes its way through the batch layer into serving layer it can be discarded from the speed layer

- Complexity isolation – complexity is pushed to a layer working on small subset of data and whose results are temporary

Lambda architecture summary

batch view = function(all data)

real-time view = function(real-time view, new data)

query = function(batch view, real-time view)

Criticism of Lambda architecture

- Complexity of overall architecture
- Need to maintain code that needs to produce the same result in two complex distributed systems
- Alternative is to use a batch processing framework like MapReduce if you aren't latency sensitive, and use a stream processing framework if you are.
  - i.e., remove speed layer, or give up the batch layer and process everything in the speed layer
- May be possible to replicate the advantages of Lambda using modern stream processing platforms – "Kappa architecture"

Lambda architecture in use

- Yahoo

  For running analytics on its advertising data warehouse, Yahoo has taken a similar approach, also using Apache Storm, Apache Hadoop, and Druid[2].

- Netflix

  The Netflix Suro project is the backbone of Netflix's Data Pipeline that has separate processing paths for data but does not strictly follow lambda architecture since the paths may be intended to serve different purposes and not necessarily to provide the same type of views[3].

- LinkedIn

  Bridging offline and nearline computations with Apache Calcite.

Summary

- Big Data solution
- CAP theorem
- Architecture of Big Data solution
- Lambda architecture