Big Data Platforms (5 ECTS) DATA14003 Lecture 10

Keijo Heljanko

Department of Computer Science
University of Helsinki
keijo.heljanko@helsinki.fi

14.10-2021

Last Lecture

- Lecture 11 topic will be on our own research on Big Data processing
- Lecture 11 will not have any quizzes associated with it

Cloud and Open Source

- What kinds of open source projects are supporting the cloud in addition to the usual suspects such as Linux?
- Quite often successful open source projects start as clones of existing proprietary systems, so it makes sense to take a look at the proprietary industry leaders, and figuring out the open source alternatives to their software stacks

What is Inside Amazon IaaS Stack?

- Amazon Web Services is currently the leading commercial IaaS provider
- Many successful open source projects have started as clones of proprietary software
- What kinds of components are inside the Amazon Web Services IaaS stack?

Amazon Web Services - Top Level Categories

- Compute & Containers
- Databases
- Application Integration (Messaging)
- Security, Identity & Compliance, Crypto & PKI
- Machine Learning & Analytics
- Management & Governance
- Networking & Content Delivery, Media Services
- IoT, Web & Mobile
- Development Tools, Applications
- ... and many more ...

Amazon Compute and Open Source

- Many categories are not software but other services and hardware
- Some categories will have open source replacements, and many are currently under heavy development:
- Compute, virtual machines/Amazon EC2:
  - OpenStack Compute (`http://www.openstack.com/`)
- Compute, scalable batch processing, Apache Hadoop MapReduce (Google MapReduce clone) and Apache Spark

Amazon Storage and Open Source

- Storage, local network block storage/Amazon EBS:
  - Ceph Block Device (https://docs.ceph.com/en/latest/rbd/)
- Storage, global network object storage/Amazon S3:
  - Ceph Object Gateway (https://docs.ceph.com/en/latest/radosgw/)
  - OpenStack Object Storage (https://wiki.openstack.org/wiki/Swift)
- Event streaming Amazon Kinesis:
  - Apache Kafka (https://kafka.apache.org/)

Open Sourcing the Datacenter

- Open sourced datacenter design: Open Compute (`http://opencompute.org/`) originally from Facebook but now a very large consortium
  - Datacenter design based on heavy use of outside air cooling
  - Custom server designs for Intel and AMD processors
  - Designs of electrical systems and distributed UPSs
  - All designs fully open sourced

Google's Infrastructure

- Google is the leading SaaS cloud provider, what new components do they use, and what are the open source replacements?
  - Google Filesystem GFS (distributed filesystem) - Hadoop Distributed Filesystem HDFS
  - Google MapReduce (distributed batch processing) - Hadoop MapReduce - Apache Spark
  - Google Bigtable / Spanner / Dremel (distributed database) - Hadoop Database (HBase) / Apache Cassandra / Apache Hive / Cloudera Impala-Spark SQL-Facebook Presto
  - Google Chubby (distributed coordination service) / Apache Zookeeper / `etcd` (e.g., Kubernets)

Google's Infrastructure (cnt.)

- Many components have open source replacements developed by other Web companies: Facebook, LinkedIn, Twitter, . . .
- Google's systems implement a massively distributed fault tolerant PaaS infrastructure for the SaaS solutions from Google
- Runs on hardware made from commodity components
- Major focus on automated fault tolerance an multi-tenancy to minimize the number of administrators/computer
- Google has the "luxury of embarrassing parallelism": Running a single word processing application is inherently sequential but running a million word processing applications is embarrassingly parallel

Cloud Storage - Node Local Storage

There are many different kinds of storage systems employed in clouds:

- Node local hard disks
  - Often used for temporary data and binaries
  - High performance but quite often storage needs to be over-provisioned, as unused storage can not be easily shared to other nodes
  - Even if RAID is used for data redundancy, the server itself is a single point of failure

Cloud Storage - Network Block Device

- Network block devices
  - Examples: Amazon EBS, Ceph block device
  - Virtual block devices accessed over the network, often with RAID 1-like durability
  - Many systems only allow mounting each block device by one client. Can be made very scalable by serving different clients by different storage servers
  - Allows much more efficient usage of storage space, needs a (fault tolerant) management layer to manage the mapping of physical storage to virtual storage devices shown to clients
  - More traditional alternatives with less automatic management: Linux DRBD (Distributed Replicated Block Device), SAN storage over iSCSI

Cloud Storage - POSIX Filesystems

- POSIX filesystems
  - Examples: NFS, Lustre (almost POSIX)
  - Shared POSIX filesystems accessed by several clients
  - Often of quite limited scalability due to sharing the metadata of a distributed namespace between clients. Metadata updates are handled by a single server
  - Because of limited scalability, traditional POSIX filesystems are rarely used in cloud based applications

Distributed WORM FileSystems

- Distributed non-POSIX Write-once-read-many (WORM) filesystems
  - Examples: GFS, HDFS
  - Tailored for big files and write-once-read-many (WORM) workloads. Are very scalable in these usage scenarios
  - Both GFS and HDFS were originally limited by a single metadata server (NameNode in HDFS). This bottleneck is typically worked around by sharding the filesystem over several metadata servers
  - Approaches such as the HopsFS exist where the NameNode is replaced by a distributed Database based solution to scale up metadata handling
  - Windows Azure BLOB Storage is a proprietary storage system with HDFS compatible API

Object Storage

- Scalable object stores
  - Example: Amazon S3, Openstack Swift (developed by Rackspace), Ceph Object Storage
  - No nested filesystem directory hierarchy available: Objects are stored in buckets. Each stored item is accessed by a unique identifier
  - Extremely scalable as each object can be served by a different server based e.g., on a hash of the object identifier
  - Usually accessed with HTTP REST API, Amazon S3 API being the most common choice

Object Storage - Amazon S3

- S3 is a geographically replicated storage system, each data item is stored in at least two geographically remote datacenters by default
- Amazon S3 has been designed for 99.999999999% durability and 99.99% availability of objects over a given year.
- Amazon S3 is “strongly consistent” since Dec 2020:

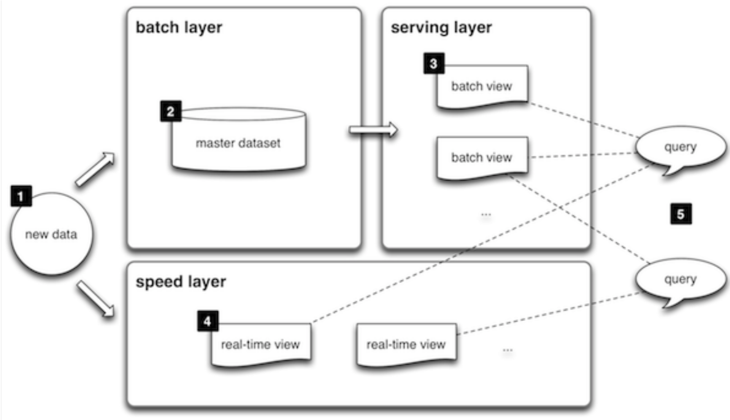
  (https://aws.amazon.com/blogs/aws/amazon-s3-update-strong-read-after-write-consistency/)
- However, two concurrent writes to S3 will still have “last write wins”-semantics based on timestamps associated with the writes and there is no locking available nor multi-key transactions.
- Thus it is tricky to implement a database directly on top of S3, and write-once-read-many is preferred

Lambda Architecture

- Batch computing systems like MapReduce and Spark in batch mode have a large latency in order of minutes to hours but have extremely good fault tolerance - each data item is processed exactly once (CP)
- Sometimes data needs to be processed in real time
- Stream processing systems such as Apache Storm allow for large scale real time processing without minimal fault tolerance - each data item is processed at least once (AP)
- These stream processing frameworks can provide latencies in the order of tens of milliseconds
- Lambda architecture is the combination of consistent / exact but high latency batch processing and available / approximate low latency stream processing

Lambda Architecture (cnt.)

- For more details, see e.g.,: `http://lambda-architecture.net/`



Lambda Architecture (cnt.)

- Batch layer - exact computation with high latency, maintaining master dataset, CP system
- Speed layer - approximate computation (minimal fault tolerance) with low latency, only maintaining approximations on recent data that is not yet processed by the batch layer, AP system
- Serving layer - serves the output of the batch layer and speed layer
- Lambda architecture allows for both low latency but also "eventual accuracy" of the approximate results, as these approximations are all eventually overwritten by exact results computed by the batch layer, combination of AP and CP systems

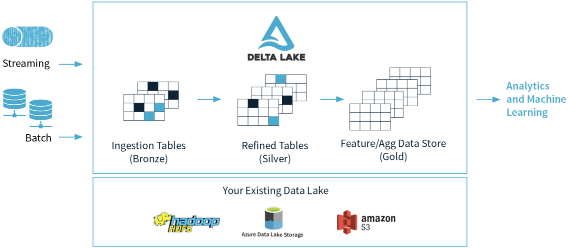
Stream Processing Frameworks

- One of the motivations for the Lambda architecture was the lack of fault tolerance in the Stream Layer
- New Streaming Frameworks have been introduced that have fault tolerance:
  - Spark Structured Streaming - Extension of Spark to stream processing
  - Apache Flink - Open source low latency streaming framework
  - Google Cloud Dataflow
  - Azure Stream Analytics
  - Apache Beam - Open source programming framework for batch and streaming, originally create by Google. Runners for: Google Cloud Dataflow, Spark, Flink, etc.
- Stream processing systems require a fault-tolerant pub-sub messaging system, e.g.: Apache Kafka (Open Source), Amazon Kinesis, Azure Event Hub

Apache Kafka

- An Open Source fault tolerant Publish/Subscribe messaging platform
- Uses Zookeeper (in the future: KRaft) for storing slowly changing management and configuration data (`etcd` does the same in Kubernetes)
- For implementing fault tolerance uses a custom replication algorithm similar in spirit to Paxos, Zab, and Raft but differing in technical details
- Uses a very large number of Pub/Sub queues called partitions in a topic. Each one of the queues has a separate leader, thus the leader bottleneck is avoided by using a very large number of leaders.
- Similar ideas are used in many sharded distributed databases, for example Apache Kudu, CockroachDB, and Google Spanner. This does not, however, solve the global transactions problem.

Data Warehouse - Delta Lake



- Three tier data curation process in a Data Warehouse
  - Bronze - Raw ingested data, including history
  - Silver - Filtered, cleaned, and augmented data
  - Gold - Business level aggregates and machine learning features
- Keeping historical data is important to fix any bugs in data cleaning or aggregation when found later on

Delta Lake

- Data Warehousing framework implementing ACID transactions
- Can implement Database tables with transactions on top of an object storage system: S3, Azure Blob Storage, Azure Data Lake Storage gen2, and also HDFS
- Can integrate with streaming systems, exact minimum latencies are based on the object storage system but are in the order of several seconds minimum
- Can be used to manage and lookup Table data partitions on S3 objects or HDFS files to minimize metadata load on these services
- Can be used also for providing GDPR data removal functionality and audit logs

Delta Lake and S3

- Needs an external locking service to be implemented when used with S3 to manage concurrent writes (dataloss!) of S3 objects, available from the Databrics commercial version
- See: “Warning! Concurrent writes to the same Delta table from multiple Spark drivers can lead to data loss.”:

  `https://docs.delta.io/latest/delta-storage.html#amazon-s3`
- Other supported storage systems do not have this problem

Approximation - Probabilistic Datastructures

- Computing exactly queries such as the distinct number of users visiting a Website requires processing all of the data
- If we have new data coming in frequently, keeping such statistics updated will require recomputation
- Probabilistic data structures can be used to compute approximate answers to such queries in an approximate fashion
- They can also be used to both parallelize and make the computation fully incremental

Probabilistic Datastructures in Spark

- The Spark `distinct().count()` dataframe transformation can be approximated with the `approx_count_distinct()` tranformation
- It internally uses the HyperLogLog++ probabilistic datastructure for to minimize the amount of data that needs to be shuffled over the network, and thus speeds up queries
- Other approximate computations with probabilistic data structures supported by Spark include: approximate quantiles, frequent items, frequency estimation with count-min-sketch, Bloom filters, and stratified sampling
- For documentation, see:

  `https://spark.apache.org/docs/latest/api/java/org/apache/spark/sql/DataFrameStatFunctions.html`

Incremental Computation - Algebird

- Algebird is a library for incremental computation implemented in Scala, and can be used in combination with Spark to implement incremental processing
- The basic idea is that the data can be partitioned in batches, and each batch can be summarized (either sequentially or in parallel) into a small probabilistic data structure called a sketch
- All of the sketches can be then combined into a single sketch for answering the query. It can be also incrementally updated
- This requires the operation with which the sketches are combined to summary sketches to be associative, and Algebird calls the suitable data representations Monoids
- See: `https://twitter.github.io/algebird/resources_for_learners.html`