CS-E4640 - Big Data Platforms
Lecture 7

Keijo Heljanko

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

7.11-2018

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
- Content Delivery
- Database
- Deployment & Management
- E-Commerce
- Industry-specific Clouds
- Messaging
- Monitoring
- Networking
- Payments & Billing
- Storage
- Support
- Web Traffic
- Workforce

Amazon Compute and Open Source

- Many categories are not sofware 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:
  - Sheepdog project (`http://www.osrg.net/sheepdog/`)
  - Ceph Rados (`http://ceph.newdream.net/category/rados/`)
- Storage, global network object storage/Amazon S3:
  - OpenStack Object Storage (`http://www.openstack.com/`)

Open Sourcing the Datacenter

- Open sourced datacenter design: Open Compute (`http://opencompute.org/`) from Facebook
  - 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 / Megastore / Dremel (distributed database) - Hadoop Database (HBase) / Apache Cassandra / Apache Hive / Cloudera Impala-Spark SQL-Facebook Presto
  - Google Chubby (distributed coordination service) / Apache Zookeeper

Google's Infrastructure (cnt.)

- Many components have open source replacements developed by Google's competitors: Yahoo!, Facebook, 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 RBD (Rados block device), Sheepdog project
  - 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

Cloud Storage (cnt.)

- Distributed non-POSIX 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 are moving towards having more than one metadata server (GFS v2, YARN/MapReduce 2.0/Hadoop 0.23), as currently this it still is the bottleneck in large (1000+) node clusters

Cloud Storage (cnt.)

- Scalable object stores
  - Example: Amazon S3, Openstack Swift (developed by Rackspace), Ceph Object Storage
  - S3 is a geographically replicated storage system, each data item is stored in at least two geographically remote datacenters by default
  - 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
  - Often accessed with HTTP REST API, also used for storing Amazon EBS snapshots
  - Amazon S3 has been designed for 99.999999999% durability and 99.99% availability of objects over a given year. One needs geographic replication to achieve the durability goal

Cloud Datastores (Databases)

- Quite often the term Datastore is used for database systems developed for the cloud, as they often do not fully support all features of traditional relational databases (RDBMS)
- Other term used often is NoSQL databases, as the original systems did not support SQL. However, some SQL support is getting added also to cloud datastores, so the term is getting outdated quite fast
- The cloud datastores can be grouped by many characteristics

Scalable Cloud Data Store Features

A pointer to comparison of the different cloud datastores is the survey paper: “Rick Cattell: Scalable SQL and no-SQL Data Stores, SIGMOD Record, Volume 39, Number 4, December 2010”.

`http://www.sigmod.org/publications/sigmod-record/1012/pdfs/04.surveys.cattell.pdf`

Scalable Cloud Data Store Features (cnt.)

The Cattell paper lists some key features many of these new datastores include (no single system contains all of these!):

- The ability to horizontally scale “simple operation” throughput over many servers
- The ability to automatically replicate and to distribute (partition/shard) data over many servers
- A simple call level interface or protocol (vs SQL)

Scalable Cloud Data Store Features (cnt.)

- A weaker concurrency model than the ACID transactions of most relational (SQL) database systems
- Efficient use of distributed indexes and RAM for data storage
- The ability to dynamically add new attributes to data records (no fixed data schema)

A Grouping of Data Stores - Key-value Stores

- Key-value stores
  - Examples: Redis, Scalaris, Riak, Apache Cassandra (partially)
  - A unique primary key is used to access a data item that is usually a binary blob, but some systems also support more structured data
  - Quite often use peer-to-peer technology such as distributed hash table (DHT) and consistent hashing to shard data and allow elastic addition and removal of servers

A Grouping of Data Stores - Key-value Stores(cnt

- Key-value stores (cnt.)
  - Some systems use only RAM to store data (and are used as memcached replacements), others also persist data to disk and can be used as persistent database replacements
  - Most systems are AP systems but some (Scalaris) support local row level transactions
  - Cassandra is hard to categorize as it uses DHT, is an AP system, but has a very rich data model

A Grouping of Data Stores - Document Stores

- Document Stores
  - Examples: Amazon SimpleDB, CouchDB, MongoDB
  - For storing structured documents (think, e.g., XML)
  - Do not usually support transactions or ACID semantics
  - Usually allow very flexible indexing by many document fields
  - Main focus on programmer productivity, not ultimate data store scalability

A Grouping of Data Stores - Extensible Record Stores

- Extensible Record Stores (aka “BigTable clones”)
  - Examples: Google BigTable, Google Megastore, Apache HBase, Hypertable, Apache Cassandra (partially)
  - Google BigTable paper gave a new database design approach that the other systems have emulated
  - BigTable is based on a write optimized design: Read performance is sacrificed to obtain more write performance
  - Other notable features: Consistent and Partition tolerant (CP) design, Automatic sharding on primary key, and a flexible data model (more details later)

A Grouping of Data Stores - Scalable Relational Systems

- Scalable Relational Systems (aka Distributed Databases)
  - Examples: MySQL Cluster, VoltDB, Oracle Real Application Clusters (RAC)
  - Data sharded over a number of database servers
  - Usually automatic replication of data to several servers supported
  - Usually SQL database access + support for full ACID transactions
  - For scalability joins that span multiple database servers or global transactions should not be used by applications
  - Scalability to very large (100+) database servers not demonstrated yet but there is no inherent reason why this could not be done

Eventual Consistency / BASE

- The term “Eventually Consistent” was popularized by the authors of Amazon Dynamo, which is a datastore that is an AP system - accessible and partition tolerant
- Also the term BASE (basically available, soft state, eventually consistent) is a synonym for the same approach
- Basically these are datastores that value accessibility over data consistency
- Quite often they offer support to automatically resolve some of the inconsistencies using e.g., version numbering or CRDTs - Commutative Replicated Data Types
- Example systems: Amazon Dynamo, Riak, Apache Cassandra

RDBMS Supporter View

- RDBMS can do everything the scalable cloud data stores can do with similar scalability when used properly
- SQL is a convenient expressive declarative query language
- RDBMS have 30 years of engineering experience put into them and are highly tuned
- There are a very large number of specialized RDBMS for various application domains (interactive vs. analytics, RAM vs. disk based data, etc.)
- A standardization around relational schema and SQL brings benefits in design and training, where same set of skills can be effectively employed with another RDBMS system

RDBMS Opponent View

- The RDBMS vendors have not demonstrated scalability matching that of the newly architected scalable cloud data stores (e.g., BigTable)
- Primary key lookup is more easy to understand than SQL, and thus enables a much lower learning curve
- RDBMS force data schema on applications unnecessarily
- SQL makes it “too easy” to write expensive queries by accident such as complicated joins involving many tables and several servers. If these expensive operations are removed from the query language, the complexity of a query evaluation becomes obvious to the programmer
- RDBMS systems have become dinosaurs in their 30 year rule of the database market, and the (hardware) assumptions on which they have been built are obsolete

Scalable Data Stores - Future Speculation

- Long running serializable global transactions are very hard to implement efficiently, for scalability they should be avoided at all costs. Counterexamples: Google Percolator, Google Cloud Spanner
- The requirements of low latency and high availability make AP solutions attractive but they are more difficult to use for the programmer (need to provide application specific data inconsistency recovery routines!) than CP systems
- The time of “one size fits all”, where a RDBMS was a solution to all datastore problems has passed, and scalable cloud data stores are here to stay

Scalable Data Stores - Future Speculation (cnt.)

- ACID transactions are needed for, e.g., financial transactions, and there traditional RDBMS will be dominant
- Many of the new systems are not yet proven in production
- There will be a consolidation to a smaller number of data stores, once the “design space exploration” settles down

NewSQL - Google Cloud Spanner

- A beta release of Google's internal Spanner Database
- SQL support
- ACID transactions that can span multiple servers globally
- Fully consistent database with a single global database image
- Implementation based on Paxos for replication, Transaction commit priority made using physical timestamps
- All database server clocks syncronized very tightly using atomic clocks + GPS based time syncronization
- All database transactions have to wait two times the max clock drift holding write locks before committing a transaction
- Open source alternatives being developed: CockroachDB, Apache Kudu