Survey on Cloud Database and Its Applications

Shuihai Hu, Hong Zhang
What is Cloud Database?

**Definition:** A cloud database is a database that typically runs on a cloud computing platform.

1. **Virtual Machine (VM) Image:** Users purchase VMs for a limited time, and run a database on the VMs.
2. **Database as a service (DBaaS):** Cloud platforms offer options for using a database as a service.
Benefits of Cloud Database

- **Cost Saving**: Pay only for what you use and when you use it.

- **Scalability and Elasticity**: Scale up and down rapidly.

- **Reduced Administrative Burden**: No need to maintain expensive hardware full-time.
Cloud Database Around the World
Characteristics of Cloud Database

• **Very Large Data Size:** Companies like Google needs to process 20 PB data per day.

• **Partition:** A logical database are partitioned into distinct independent parts for manageability, performance or availability reasons.

• **Replication:** The same data is stored on multiple storage devices for high-availability.
Two Research Topics in Cloud Database

1. Reduce User-perceived Latency in Cloud Database Service

2. Facilitate Analytical Applications in the Cloud Database
Reduce User-perceived Latency in Cloud Database Service
Problem: High Latency Variance

Latency measurements of cloud database including Amazon S3 and Windows Azure, the results indicate high user-perceived latency variance:

- 120 PlanetLab sites as clients
- Upload and download 1KB objects
Latency Stages in Cloud Database

- **Stage 1**: Internet Latency
- **Stage 2**: Client Processing Latency.
- **Stage 3**: Data Center Network Latency.
- **Stage 4**: Storage Service Latency.
Latency variance in stage 1 (Internet Latency) is mainly caused by:

- There are multiple cloud data centers for a user to choose.
- There are multiple paths to send a request to a given data center.
Latency variance in stage 2 (Client Processing Latency) is mainly caused by:
- Bad load balancing among client nodes.
- Various processing capacity of client nodes (e.g., CPU, memory).
Latency variance in stage 3 (Data Center Network Latency) is mainly caused by:
  • Different routing paths have different instant network delays.
Latency variance in stage 4 (Storage Service Latency) is mainly caused by:

- Servers have various service rates.
- Bad load balance among database servers.
Related Works

1. Balance server load via replica selection:
   • C3: Cutting Tail Latency in Cloud Data Stores via Adaptive Replica Selection. NSDI’15

2. In-network Cache:
   • Be Fast, Cheap and in Control with SwitchKV. NSDI’16

3. Duplicating or reissuing requests:
   • Low Latency via Redundancy. CoNext’13
   • CosTLO: Cost-Effective Redundancy for Lower Latency Variance on Cloud Storage Services. NSDI’15
Solution-1: Balance load via replica selection

• Same data is stored on multiple database servers for high-availability.
• **Goal**: Reduce tail latency via a good replica selection algorithm
• **Challenge 1: Service-time variations.**
  - Servers have various number of queued requests to be processed.
  - Server processing time is not a constant.
Challenges in replica selection

• **Challenge 2:** Greedy strategy will lead to herd behavior and load oscillations.
  ➢ All servers send their requests to the fastest server.
Design of C3

• **Replica Ranking**: Based on the feedback from individual servers, clients rank and prefer servers according to a scoring function.

• **Distributed Rate Control**: Every client rate limits requests destined to each server, adapting these rates in a fully-distributed manner using a cubic rate control algorithm.
Replica Ranking

- Balance product of queue-size and service time: \( q \cdot \mu^{-1} \)

- Client will prefer the server with the smallest product of queue-size and service time.
Server-side Feedback

- Servers piggyback \( \{ q \} \) and \( \{ \mu^{-1} \} \) in every response to update the server information stored in clients.
A Client adjusts its sending rate each time a response is received.

- If the client’s sending rate is lower than the receive rate, it increases its rate according to a cubic function.
- If the receive-rate is lower than the sending-rate, the client decreases its sending-rate multiplicatively.
Solution-2: In-network Cache

- Data access is highly skewed and dynamic.
- **Goal**: handle highly skewed and dynamic workloads via in-network cache.

Access frequencies to different data.
Solution-2: In-network Cache

• How to efficiently serve queries with database servers and in-network cache?
• How to efficiently update the cache under dynamic workloads?

Store less-hot data in the database servers. Cache hottest queries in network.

less-hot queries, better-balanced loads

hottest queries
Design of SwitchKV: Content-aware Routing

Switches route requests directly to the appropriate nodes

- Latency can be minimized for all queries
- Throughput can scale out with # of backends
- Availability would not be affected by cache node failures
Goal: **react quickly** to workload changes with **minimal updates**

- Only cache the hottest k <key-value> items.
- Update switch rules periodically.
- Update bursty hot data immediately.
Solution-3: Duplicating or Reissuing Requests

- Idea: Duplicate or reissue requests at each latency stage, and adopt the earliest response.
• **Challenge**: duplicating every request increases system utilization, and consumes more bandwidth and computation resource. Request duplication will not always improve latency performance.
Two approaches to address this challenge:

1. Develop a queuing model to identify a server-side threshold load below which replication always improves latency (Low Latency via Redundancy. CoNext’13).

2. Build a system to estimate the latency of possible duplication strategies for making duplication choices (CosTLO, NSDI’15)
Facilitate Analytical Applications in Cloud Database
Outline

• **Q1:** How to perform analytical jobs (e.g., machine learning, content parsing, data analysis, et.al,) in cloud database?

• **Part I:** Process analytical jobs in cloud database using parallel computing frameworks;

• **Q2:** How to speed up these applications?

• **Part II:** Speedup analytical applications in cloud database by network scheduling
Process analytical applications in cloud database using parallel computing framework
Analytical jobs in the Cloud Database

• In class we have discussed how to do join or some sophisticated query in a relational database

• How to do it in a cloud database?
• Parallel relational database management systems
  • Gamma, Bubba, and Volcano
  • Supports SQL & perform automatic SQL query optimization

• Problem: SQL might not be enough for all cloud databases
SQL does not scale so well

- Cloud database is big and increasing
  - Facebook has 2.5 PB of user data + 15 TB/day (4/2009)
  - eBay has 6.5 PB of user data + 50 TB/day (5/2009)

- As a result, many cloud database are noSQL
  - Google’s BigTable,
  - Amazon’s Simple DB,
  - …
SQL is not general enough

- SQL is too limited for many applications
  - In order to support database requirements such as in-place updates and efficient transactions, SQL adopts a very restrictive type system.
  - The declarative “query-oriented” nature of SQL makes it difficult to express common programming patterns such as iteration.

- “Together, SQL is unsuitable for tasks such as machine learning, content parsing, and web-graph analysis that increasingly must be run on very large datasets.”

----DraydLINQ(OSDI’08)
How to perform analytic jobs in cloud database?

• Parallel relational database management systems (from the 80s):
  • Gamma, Bubba, and Volcano
  • Supports SQL & automatic SQL query optimization

• Parallel computing framework (OSDI’04):
  • Designed for large clusters
  • Data is accessed in “native format”
  • Supports many query languages
  • More flexible to program
Parallel computing framework--- components

- Parallel computing framework --- Hadoop ecosystem
  - Filesystem
    - HDFS (fault tolerance)
  - Database
    - Can be noSQL (MongoDB,Hbase)
  - Algorithm/pattern
    - Map-reduce(OSDI’04)
      ---- An algorithmic technique for the distributed processing of large amounts of data;
  - Higher level languages
    - Hive/Pig (SIGMOD’08)
How Hadoop processes a job?

**Input files**

**Map phase**

**Intermediate files (on local disk)**

**Reduce phase**

**Output files**
Hive and Pig

• Complex operations require multiple MapReduce jobs
  • Hive, Pig ---- higher-level language based on Hadoop

• Hive: data warehousing application in Hadoop
  • Query language is HQL, variant of SQL
  • Tables stored on HDFS as flat files
  • Developed by Facebook, now open source

• Pig: large-scale data processing system
  • Scripts are written in Pig Latin, a dataflow language
  • Developed by Yahoo!, now open source
  • Roughly 1/3 of all Yahoo! internal jobs

• Common idea:
  • Provide higher-level language to facilitate large-data processing
  • Higher-level language “compiles down” to Hadoop jobs
Pig (SIGMOD’08) Script in Hadoop

Load Visits → Map₁

Group by url → Reduce₁

Foreach url generate count

Load Url Info → Map₂

Join on url → Reduce₂

Group by category → Map₃

Foreach category generate top10(urls) → Reduce₃

Pig Slides adapted from Olston et al. (SIGMOD 2008)
Frameworks used in cloud database

• General purpose
  • Drayd (SIGMOD’09)
  • Spark (NSDI’12)
• ..... 

• Graph database
  • GraphX (OSDI’14)
  • Pregel (SIGMOD’10)
• ... 

• In one word, parallel computing framework is a powerful way of performing analytical applications in cloud databases;
Speedup analytical applications in cloud database by network scheduling
Data analytical job speedup

• Focus on speed up calculation stage (e.g., Tetris, Quincy)
• Focus on speed up communication stage (e.g., Varys, Baraat, D3)
Network scheduling important for data-parallel jobs

- Network-intensive stages (e.g., shuffle, join)
  - More than 50% time spent in network transfers*

- Network bandwidth in a datacenter is shared across apps
  - Nearly 50% used for background transfers**

*Efficient Coflow Scheduling with Varys, Sigcomm 2014.
**Leveraging Endpoint Flexibility in Data-Intensive Clusters, Sigcomm 2013
Network scheduling for application speedup

• Problem with traditional network scheduling
  • Application agnostic --- schedule each flow as a single entity
    • PDQ (SIGCOMM’13)
    • pFabirc (SIGCOMM’14)
    • PIAS (NSDI’15)
  • Minimizing average flow completion times of individual flows is not good enough for data parallel applications!

• Coflow(Hotnets’11)
  • Definition: A group of flows within the same application in the same stage;
  • The completion of a coflow depends on the completion of the last flow;
  • For analytic jobs, minimizing coflow completion time makes more sense!
The Concept of coflow (Hotnet’11)

• The key insight behind coflows is one simple observation: a communication stage in a distributed data-parallel application cannot complete until all its flows have completed.

• As a result, coflow captures this all-or-nothing characteristic of a collection of flows.
Techniques proposed for coflow scheduling

• **Clairvoyant coflow scheduling:**
  • Varys (SIGCOMM’13)
  • Rapier (Infocom’14)

• **Non-clairvoyant coflow scheduling:**
  • Baraat (SIGCOMM’13)
  • Aalo (SIGCOMM’15)
Problem formulation of Varys

• Datacenter as a big switch
  • Only bottlenecked by the ingress and egress capacities

• Coflows $C = \{C_1, C_2, ..., C_n\}$, with $C_i = \{f_{i1}, f_{i2}, ..., f_{ij}\}$. Each flow $f_{ij}$ has its $(src, dst)$, and size $s$;

• Goal: Minimizing the average coflow completion time

• NP-hard
Algorithm design in Varys

• Heuristics on how to prioritize coflows?
  • Width-based
  • Size-based
  • Length-based

• Idea: Prioritize coflows depending on their bottleneck
  • Smallest-Effective-Bottleneck-First (SEBF)
  • How to define bottleneck? --- The bottleneck of a coflow is defined as the time to finish if there are no other coflows;
Algorithm design in Varys

- SEBF – An example
Performance of Varys

• Speed up analytic jobs by up to 3 times;
Non-clairvoyant coflow scheduling

• Problem of clairvoyant coflow scheduling: size of coflows may not be known as a prior;

• **Non-clairvoyant coflow scheduling**: scheduling coflows with no prior knowledge on coflow size;
  • Baraat (SIGCOMM’13)
  • Aalo (SIGCOMM’15)
Non-clairvoyant coflow scheduling - Baraat

• Baraat --- FIFO-LM
  • Component 1 --- FIFO (first in first out): optimal if coflows have similar size, however bad if coflow sizes have high variance;
  • Component 2 --- LM (limited multiplexing): reduce the impact of large coflows;
  • Reduce tail task completion times by 60% for data analytics workloads

Figure 5: FIFO ordering can reduce tail completion times compared to fair sharing (FS).
Non-clairvoyant coflow scheduling - Baraat

• Problem with FIFO-LM
  • FIFO works well when coflows have similar sizes
  • Limited multiplexing is not good enough when flow sizes have high variance.

• Observation: LAS (least-attained-service) scheduling works well when coflow sizes have high variance;

• Idea: combine LAS scheduling and FIFO in an appropriate way!
Non-clairvoyant coflow scheduling - Aalo

- **Aalo** proposes DCLAS (Discretized Coflow-Aware Least-Attained Service)
- **Multilevel feedback queue structure**;
- K queues with increasing quantum and decreasing priority;
- A coflow degrades to the next queue if it uses up its quantum in the queue or complete;
- Within each queue, coflows are scheduled using FIFO;
Non-clairvoyant coflow scheduling - Aalo

• **Why it works?**
  
  • Among different queues, LAS is applied so that smaller coflows have higher priorities than large ones;
  
  • Within each queue, coflows having similar sent size. Thus FIFO among these coflows helps to minimize CCTs;
Performance of Aalo

- Speed up analytic jobs by up to 2 times;
Summery

• Part I: Process analytical applications in cloud database using parallel computing framework:
  • Traditional PRDBMS is not enough for cloud database;
  • Briefly introduce some parallel computing frameworks

• Part II Speedup analytical applications in cloud database by network scheduling
  • Optimizing network is key to optimize job execution;
  • Coflow instead of flow scheduling suites for analytical jobs;
  • Introduce some resent work coflow scheduling;