RAMCloud

Scalable High-Performance Storage Entirely in DRAM

2009

by John Ousterhout et al.
Stanford University

presented by Slavik Derevyanko

Outline

- RAMCloud project overview
- Motivation for RAMCloud storage: advantages
- Evolvement of Big Data systems within Google
- Alternatives to keeping data in RAM
- Challenges faced by RAMCloud

Overview

- RAMCloud is a key-value storage system that provides **low-latency access** to large-scale datasets (up to x1000 faster access)
- Main idea: **information is kept entirely in DRAM** of cluster machines at all times
- Not a novel idea (cache, etc), but the cost was always a showstopper to keep all of the data in RAM.
- Paper from 2009, since then the storage system has been implemented, and the project is ongoing: https://ramcloud.atlassian.net/wiki/display/RAM/Setting+Up+a+RAMCloud+Cluster

RAMCloud advantages

- Because all data is in DRAM at all times, a **RAMCloud can provide 100-1000x lower latency** than disk-based systems and **100-1000x greater throughput**.
- Better latency
  - **Access latencies of 5-10 microseconds** (to read a few hundred bytes of data from a single record in a single storage server in the same datacenter).
  - In comparison, **disk-based systems** offer access times over the network ranging from **5-10ms** (x1000, if disk I/O is required) down to **several hundred microseconds (for data cached in memory).**
  - **Why important:** generating a response in queries to Amazon, Facebook, Google hits 100s of services
- Better throughput
  - A single multi-core storage server should be able to service at least **1,000,000 small requests/sec**
  - In comparison, a **disk-based system** running on a comparable machine with a few disks can service **1000-10000 requests per second**, depending on cache hit rates.

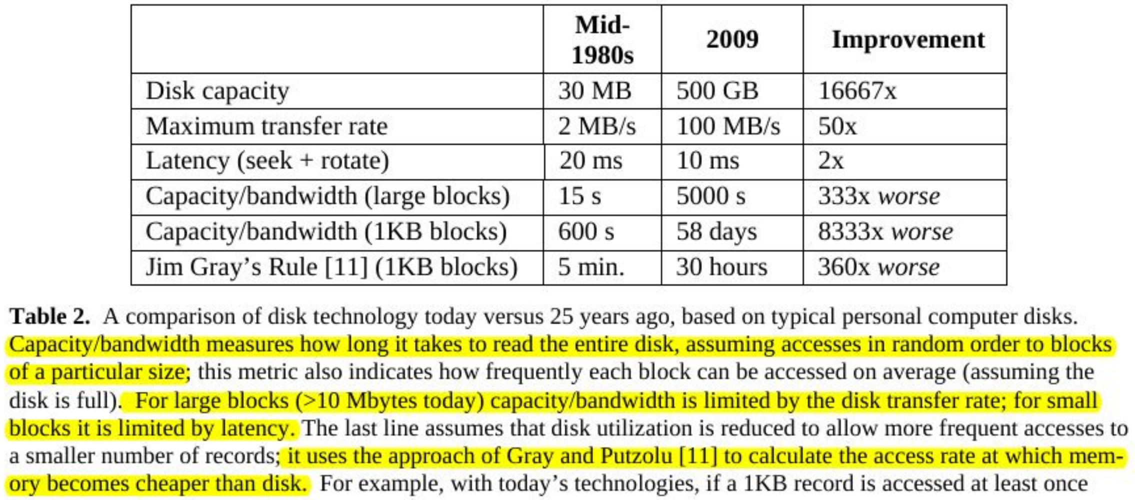
RAMCloud advantages

- Better scalability
  - Distributed transactions are extremely fast - less conflicts on updates
  - Using RAMCloud will **simplify the development of large-scale Web applications** by eliminating many of the scalability issues.

- Example: Facebook storage system (August 2009)
  - 4000 MySQL servers (these days also Hive, Cassandra, Giraph, HBase...)
  - Data is sharded: **distribution of data** across the instances and **consistency** between the instances are **handled explicitly by Facebook application code**
  - Even so, the **database servers are incapable of meeting Facebook's throughput** requirements by themselves, so Facebook also **employs 2000 memcached servers** (cache recently used query results in key-value stores kept in main memory)

HDD technology evolution

- Disk capacity has increased more than 10000-fold over the last 25 years
- The access rate to information on disk has improved much more slowly: seek time and rotational latency have only improved by a factor of two.
- **It simply isn't possible to access information on disk very frequently**
- The role of disks must inevitably become more archival



	Mid-1980s	**2009**	**Improvement**
Disk capacity	30 MB	500 GB	16667x
Maximum transfer rate	2 MB/s	100 MB/s	50x
Latency (seek + rotate)	20 ms	10 ms	2x
Capacity/bandwidth (large blocks)	15 s	5000 s	333x *worse*
Capacity/bandwidth (1KB blocks)	600 s	58 days	8333x *worse*
Jim Gray's Rule [11] (1KB blocks)	5 min.	30 hours	360x *worse*

Table 2. A comparison of disk technology today versus 25 years ago, based on typical personal computer disks. Capacity/bandwidth measures how long it takes to read the entire disk, assuming accesses in random order to blocks of a particular size; this metric also indicates how frequently each block can be accessed on average (assuming the disk is full). For large blocks (>10 Mbytes today) capacity/bandwidth is limited by the disk transfer rate; for small blocks it is limited by latency. The last line assumes that disk utilization is reduced to allow more frequent accesses to a smaller number of records; it uses the approach of Gray and Putzolu [11] to calculate the access rate at which memory becomes cheaper than disk. For example, with today's technologies, if a 1KB record is accessed at least once

Evolvement of Big Data systems within Google

Evolvement of Big Data systems within Google

“We don’t really use MapReduce anymore. The company stopped using the system years ago.”

Urs Hölzle, senior vice president of technical infrastructure at Google, 2014 Google I/O conference in San Francisco.

Evolvement of Big Data systems within Google

- MapReduce is inefficient in handling iterative data processing jobs
- Mostly suitable for offline, batch processing, not suited for streaming data processing.
- A new hyper-scale system, DataFlow is considered as its successor.
- Besides DataFlow, Google developed a series of big data systems, such as Dremel (2010), Spanner (2013) and Pregel (2010), to replace the original two, MapReduce(2004) and BigTable(2006).
- 2007 - Initial release of Apache Hadoop, open-source MapReduce implementation.

Similar evolution in open source Big Data systems

- Hadoop is found inefficient in processing iterative jobs
- Nowadays, a computing node can be equipped with very large amounts of memory, so that data can be fully maintained in the distributed memory of a cluster
- This observation motivates the development of in-memory based processing system - Apache Spark
- Using main memory to hold intermediate results, can run jobs 100 times faster than Hadoop

Present-day alternatives to RAM storage

UNIVERSITY OF WATERLOO

In-memory caching

- **Caching** - achieving high performance by keeping the most frequently accessed blocks in DRAM (If most accesses are made to a small subset of the disk blocks)
- Jim Gray's rule: diluting the benefits of caching by **requiring a larger and larger fraction of data to be kept in DRAM**
- Large-scale web-applications such as Facebook appear to have little or no locality, due to complex linkages between data (e.g., friendships)
    - **25% of all the online data for Facebook** is kept on memcached servers (**hit rate of 96.5%**).
    - Counting database server caches - approx. **75% of data is in main memory** at any point in time (excluding images)
- **Cache miss penalties**: even a 1% miss ratio for a DRAM cache costs a factor of 10x in performance
- Caches in the future **will have to be so large** that they will **provide little cost benefit** while still introducing **significant performance risk**

Flash drives

- The primary advantage of DRAM over flash memory is latency
- Flash devices have **read latencies as low as 20-50 µs**, but they are typically **packaged as I/O devices**, which adds additional latency for device drivers and interrupt handlers.
- **Write latencies** for flash devices **are 200 µs** or more.
- Overall, a **RAMCloud is likely to have latency 5-10x lower than a FlashCloud**
- RAMCloud encourages a more aggressive attack on latency in the rest of the system – RPC (routers, TCP), software stack (OS)
- Most RAMCloud techniques will apply to other technologies – flash drives, etc.

RAMCloud challenges

Cost

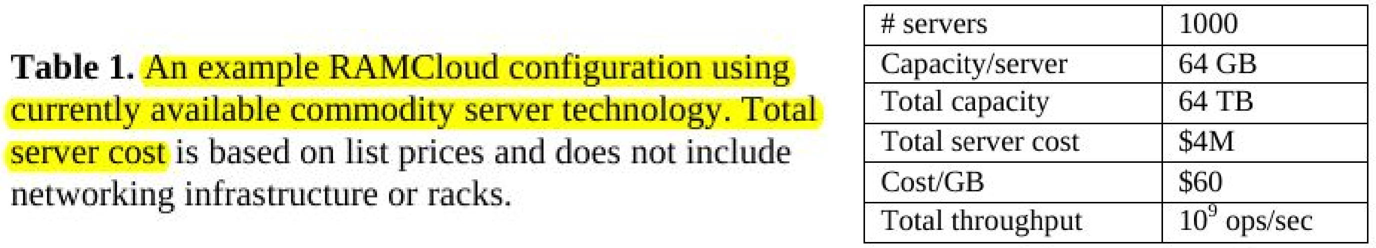
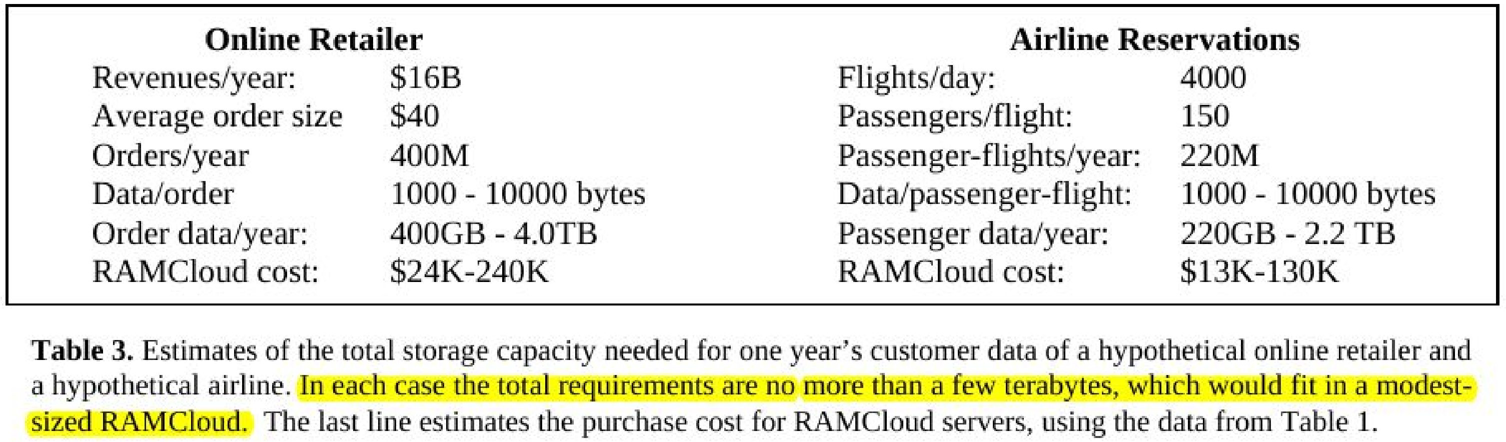


Table 1. An example RAMCloud configuration using currently available commodity server technology. Total server cost is based on list prices and does not include networking infrastructure or racks.

# servers	1000
Capacity/server	64 GB
Total capacity	64 TB
Total server cost	$4M
Cost/GB	$60
Total throughput	10^9 ops/sec

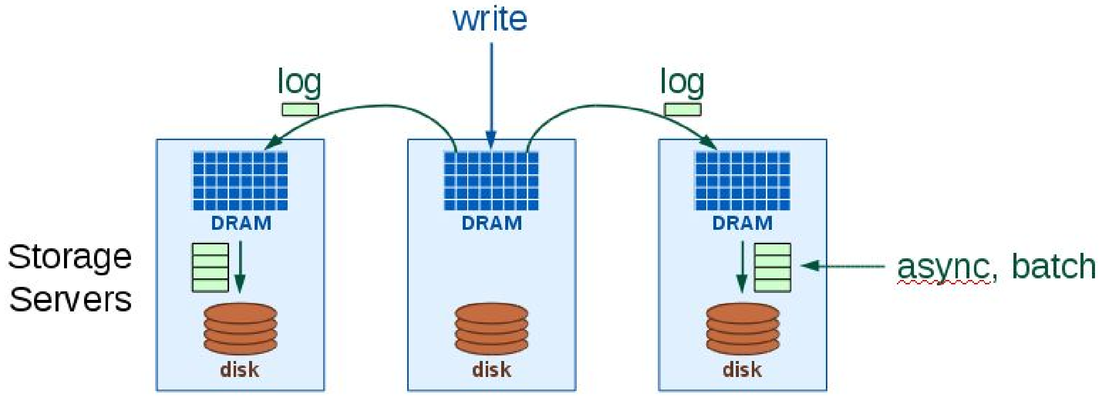


Online Retailer		Airline Reservations	
Revenues/year:	$16B	Flights/day:	4000
Average order size	$40	Passengers/flight:	150
Orders/year	400M	Passenger-flights/year:	220M
Data/order	1000 - 10000 bytes	Data/passenger-flight:	1000 - 10000 bytes
Order data/year:	400GB - 4.0TB	Passenger data/year:	220GB - 2.2 TB
RAMCloud cost:	$24K-240K	RAMCloud cost:	$13K-130K

Table 3. Estimates of the total storage capacity needed for one year's customer data of a hypothetical online retailer and a hypothetical airline. In each case the total requirements are no more than a few terabytes, which would fit in a modest-sized RAMCloud. The last line estimates the purchase cost for RAMCloud servers, using the data from Table 1.

DRAM volatility

- Data durability
  - RAMCloud ensures the durability of DRAM-based data by **keeping backup copies on secondary storage.** It uses a uniform logstructured mechanism to manage both DRAM and secondary storage, which results in high performance and efficient memory usage.



Conclusions

Conclusions

- Technology trends and application requirements dictate that a larger and larger fraction of online data must be kept in DRAM
- **Best long-term solution for many applications** may be a radical approach where **all data is kept in DRAM all the time**
- The two most important aspects of RAMClouds are
  - Extremely low latency (5-10 µs latency)
  - Scale: ability to aggregate the resources of large numbers of commodity servers
- Ongoing project. RAMCloud implementation is available at: https://ramcloud.atlassian.net/wiki/display/RAM/Setting+Up+a+RAMCloud+Cluster

Thank you!

UNIVERSITY OF WATERLOO