RAM Cloud

Shaima Alghamdi

G37443993

March 2015

Term Paper

Dr. Morris Lancaster

**Table of Contents**

Abstract: 3

1. Introduction: 3

1.1 History of Storage Systems: 4

2. RAMCloud Overview: 7

3. Importance of Low Latency 9

4. RAMCloud Architecture: 12

4.1. Data model: 13

4.2. Server architecture: 14

4.3. Networking substrate: 15

5. Recovery: 16

5.1. Using Scale: 17

5.2. Scattering Log Segments: 18

5.3. Failure Detection: 21

5.4. Recovery Flow: 21

5.5. Consistency: 22

6. RAMCloud Disadvantages: 23

7. Conclusion: 24

References: 26

# Abstract:

RAMCloud Storage System (RCSS) is a new storage technique that allows efficient random read access in cloud environments. The technique provides low-latency and efficiency by storing all data in Dynamic random-access memory (DRAM) at all times to huge and large-scale datasets. The information is kept completely in DRAM and by combining the main memories of thousands of commodity servers creates large-scale systems. Furthermore, it is usually designed to operate on tens-of-thousands of machines (Yifeng Luo et al., 2012). It implements a united log-structured method both for active information in memory and backup data on disk. In addition, a log-structured method allows 80 to 90 percent of memory management utilization while ensuring high performance. RAMCloud also provides non-stop availability by recovering from failures so quickly that the applications never notice failures. Therefore, this combination of low latency and large-scale datasets will allow for a new type of data-intensive applications.

# Introduction:

Scientists have always been trying to find ways to effectively and efficiently store information. The earliest computers used papers to store their information. In fact, the man behind the idea of using paper as a method to store information for computers can be traced back to an English mathematician called Charles Babbage. The history and development of storage systems has come a long way to become what it has today. From magnetic tapes to solid-states drive it have taken a lot of research to constantly bring improvement and innovation to storage systems.

## History of Storage Systems:

In the 1890, an American statistician called Herman Hollerith developed a punched card Tabulating Machine to help process data for the U.S. Census as well as accounting. In fact, many types of punched cards were designed after Hollerith’s original punch card. Punched cards were still be widely used for both data porting and information storage till the 1950s. Shortly, magnetic tape replaced punched cards after the electronic computer was invented. In 1928, German Fritz Pfleumer invented the magnetic tape, which was designed for recording sounds. In the 1950s, magnetic tape was being used in storing information for computers. Magnetic tape was first used by UNIVAC I 1950 computer. As magnetic tape became popular for sound recording, an issue was related to the recorded information that is supposed to be accessed in a sequential manner (Goda, 2012). Therefore, random access required reversing the tape, which resulted in an improbable long wait time.

In contrast, magnetic disks were commercialized in the 1950s, which allowed random accesses with shorter wait time. Moreover, magnetic disks have achieved effective areal density improvements resulting in the reduction of cost per capacity. In 1956, the IBM 350 Disk Storage was introduced as a major component of the IBM 305 Random Access Memory Accounting (RAMAC) system. It is a new storage media that enabled transactions to be recorded as they occurred and simultaneously reflect each entry in affected accounts. It provided random access to any record, maintained records on a real-time basis, removed peak loads, and could concurrently produces the output by either print or punched cards (IBM, 2010). In the 1961, the IBM 1301 Disk Storage Unit has been released. The newly IBM 1301 combined several heads, each for a recording surface, which were attached to one arm and moved together like a comb. This technique contributed to faster seek time. Another product was released in 1973, which is called IBM 3340 Direct Access Storage Facility (Winchester Disk). It was the first technique that sealed metal platters along with a head arm assembly and a controlling circuit. This design reduced the mechanical complexity for disk loading and unloading, thus allowing for size reduction and capacity improvements that came later (Goda, 2012).

Basically, the design of the magnetic disk products came from the basic design of the IBM 3340 magnetic disks. In the 1970 magnetic disks still had large platters between (14- and 8-in), which are requiring large-capacity power supplies to operate as well as large cabinets. However, these disks was only limited to large computer systems. In the 1980, a new product has been announced by Seagate Technology called ST-506, which was the first 5.25-in platter and 5-MB capacity hard disk drive. Nowadays, hard disk products with the size of (3.5- and 2.5-in) platter are commonly used in many devices such as data centers, desktop computers, and laptops. This brought about the idea of having cheap small disk drives that achieve huge capacity, great access throughput, and high accessibility at much lower costs rather than expensive large disk drives. In the 1990s, disk arrays became popular in data centers (Goda, 2012). The magnificent growth of areal density and drive capacity appears to lie mainly behind the fact that magnetic disks had kept the main position in secondary storage media for this half-century.

Though, since 1990 improvements have not kept up with the rapid capacity growth. In other words, because of the drive capacity, drive transfer speed has eventually slowed down. Worse than accessing information randomly in a magnetic disk, the disk head has to suspend for waiting for seeking time and rotation latency. From 1990 to 2000 around 70 percent of seeking time has been reduced merely. The rotation speed of spindle motors have hit the ceiling at 15000 r/min; due to the heat capacity issues further improvements are almost definitely impossible even in high-end products. Improving random access speed is absolutely far behind capacity growth and the gap has grown larger. Unfortunately, there is no considerable work that has been reported to change these trends. Instead, data centers are starting to influence storage class memory for high-end applications that require low-latency random accesses such as flash memory. Magnetic disks are slowly developing to improve reliability, drive capacity, and recoding density (Goda, 2012).

Since we entered the twenty-century, networking became the promising solution for storage. Storage networking can connect storage devices and computers through a network frequently designed for connecting storage devices. In fact, in the sixties J.C.R. Licklider introduced the idea of an "intergalactic computer network". In 1999, Salesforce was one of the first milestones in the history of cloud computing. The vision with Salesforce is to reinvent Customer relationship management (CRM) in the cloud (Mohamed, 2010). The massive amounts of storage resources combined by SSPs (Storage Service Providers) help users to fasten shrink or extend storage capacity and bandwidth in an on-demand method. It became realistic for users to place their business data and business application in remote data centers when both virtualization technologies became available. “Cloud Computing” was the term used to refer to these solutions. From a Cloud Computing perspective, “remote storage services that used to be called SSPs are often provided as a part of full-fledged cloud services” (Goda, 2012). Amazon S3, Windows Azure Storage, and Google Cloud Storage are major cloud-based storage services, which are all designed in close coordination with their other cloud services. These types of storage have also become more prevalent in consumer and end-user products and devices and are not limited to just enterprise solutions. Services like Apple iCloud and Amazon Cloud allow customers to manage and store their purchased contents in remote clouds. Performance and security issues are the most critical things that service providers are trying to resolve. Due to the huge growth of the digital data, managing data is gaining more importance. Evolving storage networking technology is more likely to push this data growth further. Storage system technologies will play deeper and wider role in future of the IT systems.

# 2. RAMCloud Overview:

RAMCloud Storage System (RCSS) is a new storage technique that allows efficient random read access in cloud environments. Datacenters are containing large numbers of servers that are used in RAMClouds, which are divided roughly into two categories: application servers, which perform the logic of the application such as producing Web pages or applying business rules, and storage servers, which supply longer-term shared storage for the application servers. With RAMCloud storage servers have been organized in a new way. There are two key characteristics that distinguish a RAMCloud from other storage systems. First, DRAM is storing all of the information at all times. DRAM is the permanent place for all the data. The use of disk is only for backup. Second, in order for RAMCloud to support thousands of servers it must automatically scale; “applications see a single storage system, independent of the actual number of storage servers” (Ousterhout et al., 2011). The technique provides low-latency and efficiency by storing all data in Dynamic random-access memory (DRAM) at all times to huge and large-scale datasets. The information is kept completely in DRAM and by combining the main memories of thousands of commodity servers creates large-scale systems.

In late 2008, Facebook had 800 memcached servers with 28TB of DRAM. These numbers had more than doubled in the mid of 2009, and the company runs several memcached clusters per datacenter as of 2013, where each cluster consists of thousands of servers. In order to store datasets in much larger than the capacity of a single machine, therefore, RAMCloud is designed to collect the DRAM of more than hundreds of servers within a datacenter. In fact, RAMCloud is written in C++ and runs on commodity 64-bit x86 Linux servers. Although there are special features of Linux that we do not rely on, the advantage of InfiniBand is to low-latency networks that it’s expected to be commodity in the near future. This has achieved 5μs reads and 16μs durable writes for small objects. However, for maximum performance, the advantages of x86-specific hardware features are taken into consideration; “in particular we assume little endianness, lax data alignment requirements, and use the crc32 instructions for fast data check summing and for fine-grained performance measurement. However, we expect that porting RAMCloud to other architectures and operating systems will be straightforward.” (Rumble, 2014)

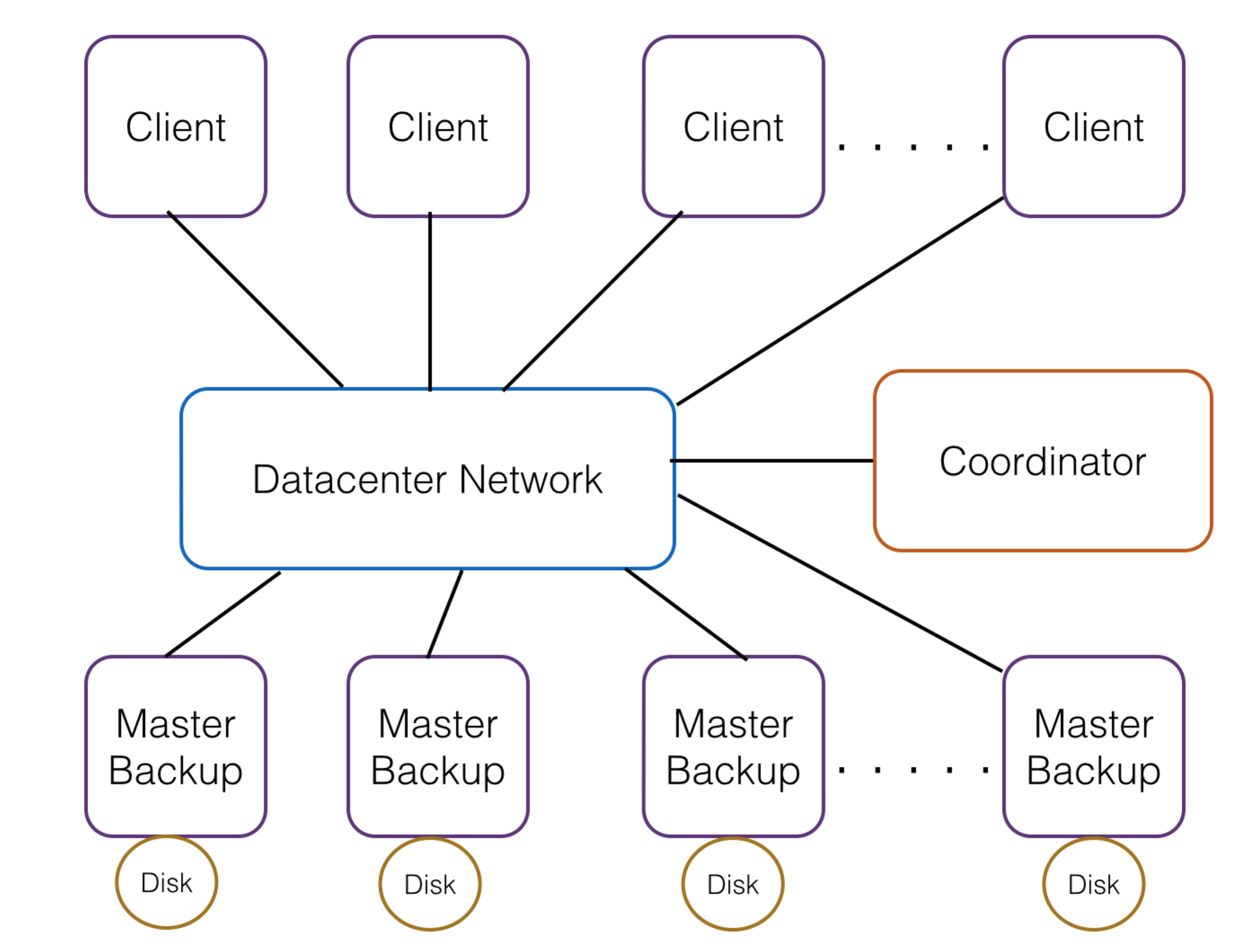
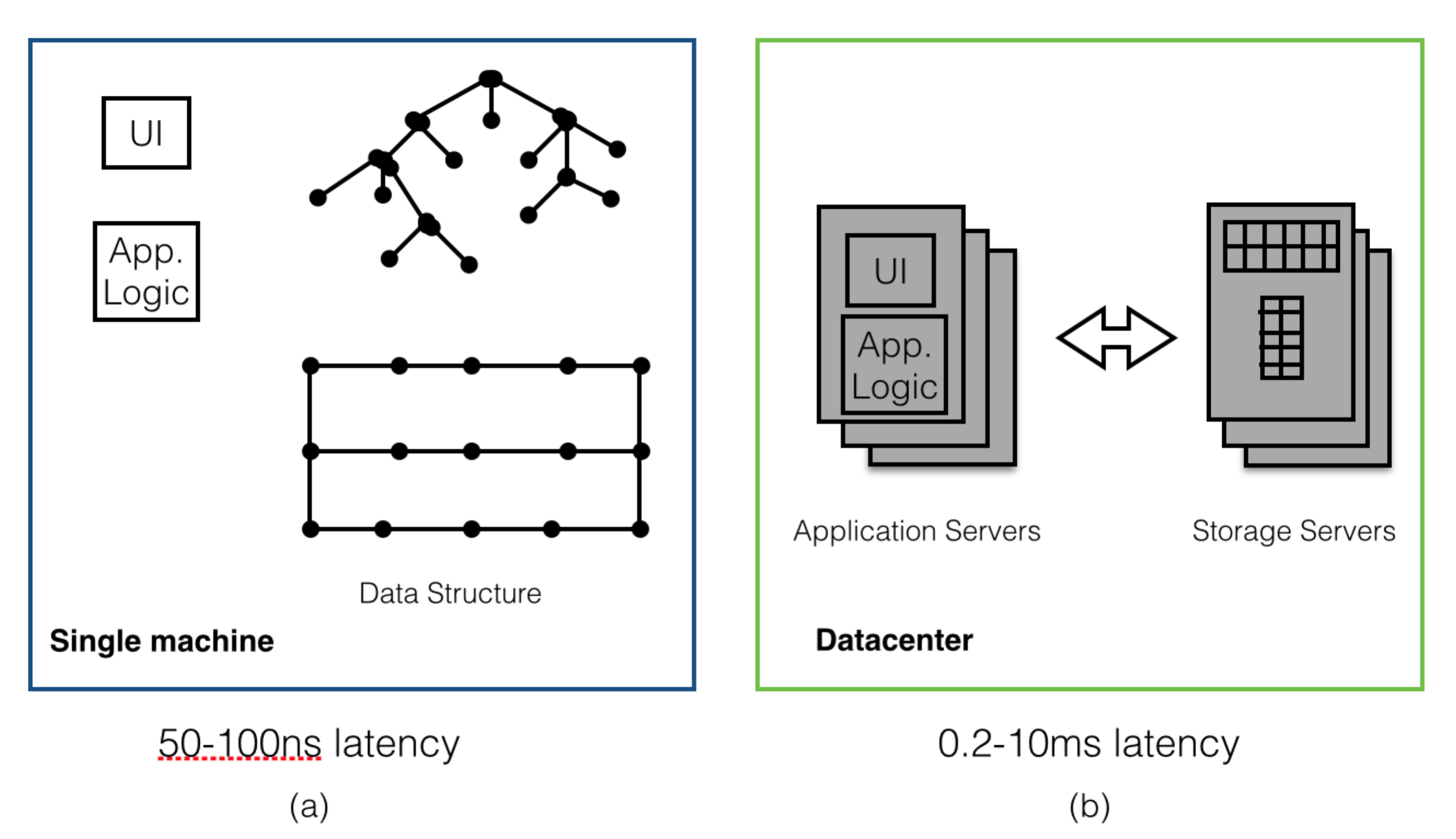


Figure 1: RAMCloud Cluster Architecture. There are two modules for each storage server: a master, which manages objects in its DRAM, and a backup, which stores segment replicas on the disk. Masters and backups are managed by the central coordinator. Client applications access RAMCloud with remote procedure calls.

# 3. Importance of Low Latency

There are many unique aspects of RAMCloud, but the main distinction for RAMCloud is the significantly lower latency when compared to existing systems. Figure 2 shows the reason of the importance of storage latency for large-scale Web applications. The applications are running the application code and all of its data will be loading into the memory of a single machine (see Figure 2(a)). The memory speed that the application data will be accessed will be between (50-100 ns); this results to intensive data manipulation while still providing collaborative reply to users. However, this method restricts application throughput to the capacity of a single machine.

Due to the Web new applications have been created that supported millions of users; it is impossible for these applications to use the single-machine approach of Figure 2(a). Instead, Web applications run on thousands of servers in a datacenter, as shown in Figure 2(b). There are two groups of servers: servers handling incoming HTTP requests, and servers for application data storage. The method that Web applications use is that application servers do not hold data between browser requests. With each request the data will be carried from storage servers then once the response for the request has been delivered to the browser the data will be discarded. With this method, depending on the speed of the network and whether the storage server is memory, flash, or disk, the latency will differ from few hundred microseconds or more, (Ousterhout et al., 2011).

Figure 2: In a traditional application (a) the application’s data is stored in memory on the same machine holding the application logic and user interface code; the last-level of cache miss time (50-100 ns) determined the latency for an application to access its data. In a scalable Web application (b) the data is stored on separate servers from the application logic and user interface code; the latency for an application to access data over the network ranges from 200-300 μs (if data is cached in the storage server’s DRAM) to 10 ms or more (if data is on disk) (Ousterhout et al., 2011).

Unlike the single-machine environment, the Web applications environment has not been scaled uniformly. Comparing to the single-server applications, the total CPU power for the Web application has improved by a factor of 1000x or more as well as the total storage capacity; however, the latency has been reduced up to 5 orders of scale when accessing application's data. Moreover, throughput has not scaled for example, when the application requests small random read, the total throughput of a few thousand storage servers in the configuration of Figure 2(b) is not much more in comparison to the single server in the configuration of Figure 2(a). Therefore, Web applications serve large group of users, and it store large amounts of data, however, while processing a given browser request it cannot use big a mount of data (Ousterhout et al., 2011).

The RAMCloud project began in 2009, Facebook, has used a similar server structure to the Figure 2(b), however, it was experiencing the issues that are related to high latency. As a primary storage Facebook used MySQL database servers for its user data. However, MySQL database servers lack latency and throughput, thus memcached servers have been supplied to these servers, which cache the query’s results in the DRAM. So by 2009, Facebook had almost 4000 MySQL servers and 2000 memcached servers. The rate of the data in memcached was almost about 97% and the latency request was around 300 μs (Ousterhout et al., 2011).

Facebook applications functionality has been limited and created complexity for developers, due to high latency of data access. Thus, to offer suitable reply times for users, a Facebook application server could only make between 100 to 150 sequential requests for data (either by memcached or MySQL) when servicing a given browser requests. Sadly, the functionality that could be provided to users will be limited. So in order to get past the limitation, Facebook had to make adjustments by allowing parallel requests when possible. Moreover, to aggregate large amount of data in each memcached object, Facebook created materialized views, which retrieves more useful data with each request. However, these techniques added huge complexity to developers. Even with these techniques, accessing application's data were still limited (Ousterhout et al., 2011).

MapReduce and Spark consider being one of the existing scalable frameworks that can manipulate large amounts of data. However, in order to hide latency with these frameworks data must be accessed in large sequential blocks. As a result, these frameworks are being used for group of jobs that run for more than an hour; thus, they are not advisable to use for Web applications or for applications that require random access (Ousterhout et al., 2011).

RAMCloud goal is to achieve low latency for a small random access in large-scale applications, which is nowadays, around 10 μs in a large datacenter and 5 μs for small clusters. In fact, this shows an improvement of 50-1000x over standard storage systems used by Web applications now (Ousterhout et al., 2011).

# 4. RAMCloud Architecture:

With the advent of applications that are very data-intensive, the importance of RAM Cloud cannot be over-emphasized. This storage platform allows for a new provision for “uniform low-latency access to very large datasets, and it ensures data durability so that developers do not have to manage a separate backing store” (Ousterhout et al., 2011). It is important to look at the overall comprehensive architecture of the RAMCloud system, the data model, consistency, cluster management and the underlying networking layer that is vital for high performance and low latency.

## 4.1. Data model:

RAMCloud supports a simple key-value store data model. This data model was chosen “because it is general-purpose enough to support a variety of applications, yet simple enough to yield a low latency implementation” (Ousterhout et al., 2011). This was also implemented to circumvent this that will hinder scalability in any form. RAMCloud divides data into tables. Each table has a unique textual name identifier and 64-bit identifier. There are multiple objects in each table which contain the following:

* A variable-length key
* A variable-length value, up to 1 MB.
* A 64-bit version number.
* Operations for creating and deleting tables.
* Operations that read, write, and delete individual objects.
* Operations that manipulate objects in bulk, including multi-object forms of read, write, and delete
* Two atomic operations, *conditional Write* and *increment* and
* Operations to split tablets and move them between masters (Ousterhout et al., 2011).

## 4.2. Server architecture:

As shown in Figure 1, the RAMCloud software package runs on an assembly of storage servers. Along with the assembly of storage servers “coordinated” by a single coordinator, RAMCloud data is accessed “over a datacenter net- work using a thin library layer” (Ousterhout et al., 2011). RAMCloud was designed to support different range of sizes of clusters from very large clusters with up to 20,000 or more servers all the way down to a cluster with 5-10 servers. Each storage server contains two components”

* A master module is the module in charge of the server DRAM management for RAMCloud data storage and the handling of I/O requests from clients.
* A backup module, which is the backup storage module, used to store data copies via local disk or flash memory. It is expected that “storage servers are to be configured with as much DRAM as is cost-effective, which is about 64-256 GB in 2014.” (Ousterhout et al., 2011)

The data stored in each table is divided into portions called tablets. In this case, a small table will consist of a single tablet and would be on just one master. However, a large table will have multiple tablets on various masters. If a table is small, it consists of a single tablet and the entire table will be stored on one master. Large tables are divided into multiple tablets on different masters. A single tablet contains the objects in one table whose key hashes fall in a given range. This method manages to distribute the objects in a given table randomly and uniformly across its tablets.

The cluster configuration is managed by the coordinator, which contains mainly of metadata explaining the recent servers in the cluster, the assignment of tablets to servers, and the recent tables. Also it is responsible for managing recovery of crashed storage servers. There is a single active coordinator, but there might be several standby coordinators, each coordinator will take over when the active coordinator crashes. The information is stored in the cluster configuration by the active coordinator on an external storage system, which is slower than RAMCloud but highly fault-tolerant. The other standby coordinators use the outside storage system to sense failures of the active coordinator, recover the configuration information, and select a new active coordinator. So in order for a single coordinator to manage a large cluster without any obstacle, it should not involve in high-frequency operations such as read and write RAMCloud objects. The configuration information is maintained by each client library, which allows it to recognize the appropriate server for a read or write request without the coordinator involvement. The clients have loaded the cache by only contacting the coordinator. If a client’s cached configuration information converts to an old information because data has moved, the client library sees this when it makes a request to a server that no longer stores the wanted information, at which point it flushes the old data from its cache and fetches up-to-date information from the coordinator (Ousterhout et al., 2011).

## 4.3. Networking substrate:

For the full functionality of RAMCloud, certain thresholds must be set, including high performance and low-latency. The networking layer must also be of a high performance function as well. There should be a high bandwidth network connection of >10Gb/s or higher full bisection bandwidth. This would allow for continuous and simultaneous network connection by all servers/machines without any network bottlenecks at all. Network packets send-and-receive time should also be 10 microseconds or less between servers (for at least 100,00) in the data center.

In recent years, these thresholds have become plausible and can be expected to become conventional. Ousterhout et al., 2011 describes using “Infiniband networking, which offers round-trip latency around 3.5 μs for a small cluster and band- width per machine of 24Gb/s.”

# 5. Recovery:

When a RAMCloud storage server crashes, the objects that had been present in its DRAM must be recovered by replaying its log that the master left behind. It requires the backup storage to read the log segments, then processing the records in those segments in order to recognize the recent version of each live object, and recovering the hash table that is used for storage operations. Till the hash table has been recovered, the crashed master’s data will be unavailable.

Luckily, if the unavailability period can be created shortly, so that the delay is not much more than other delays that are common in regular operation, and if the server crashes occur irregularly, then application’s users will not notice the crash recovery.

For most applications 1-2 second recovery is fast enough to establish “continuous availability”; this speed will only work for servers with at least 64 GB of memory (Stutsman 2013).

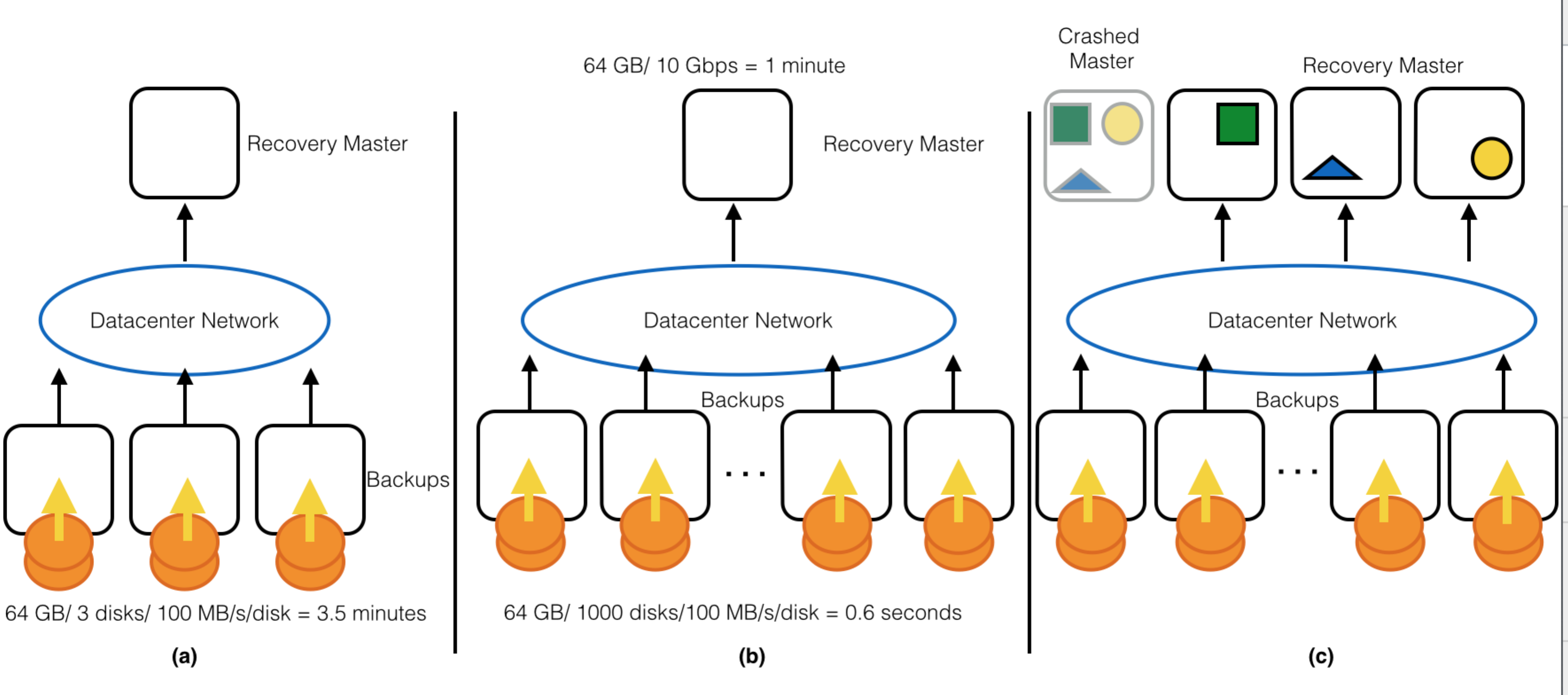


Figure 3: (a) Disk bandwidth is a recovery obstacle if each master’s data has a small number of backup machines. (b) Spreading log segments across many backups eliminates the disk obstacle, but they are recovering all data on one recovery master, which is removed by the network interface and CPU of that machine. (c) Fast recovery is achieved by dividing the data of the crashed master and it is recovering each divider on a separate recovery master.

## 5.1. Using Scale:

Taking advantage of the large resources of the cluster is the key to fast recovery in RAMCloud.

Figure 3a shows a simple mirrored method where each master has two backups and stores copies of all its log segments on each backup. Sadly, this method creates obstacle for recovery because the master’s data have to be read from only a few disks. In fact, with 3 disks, it would take around 3.5 minutes to read 64 GB of data.

Thus, by using more disks during recovery, RAMCloud works around the disk obstacle. As it shown in Figure 3b, each master spreads logs data across all of the backups in the cluster (each segment on a different set of backups). During recovery, these spread log segments can be read concurrently; with 1,000 disks, 64 GB of data can be read into memory in less than one second (Stutsman 2013).

Immediately after the segments have been read from disk into backups’ memories, they must be joined to find the most current version for each object. As shown in Figure 3b, this method is to send all the log segments to a single recovery master and replay the log on that master. Sadly, with this method, the recovery master is an obstacle: with a 10 Gbps network interface, it will take around 1 minute to read 64 GB of data, and the master’s CPU will also be an obstacle (Stutsman 2013).

As shown in Figure 3c, RAMCloud uses multiple recovery masters to remove the recovery master as the obstacle. Through recovery RAMCloud shares the objects of the crashed master into dividers of equal size. Each divider is given to a different recovery master, which gets the log data for the divider's objects from backups and joins those objects into its own log and hash table. In less than 1 second 64 GB of data can be transferred over a 10 Gbps network with 100 recovery masters working in parallel (Stutsman 2013).

Therefore, the overall method to recovery in RAMCloud is to combine the disk bandwidth, network bandwidth, and CPU cycles of hundreds of recovery masters and thousands of backups.

## 5.2. Scattering Log Segments:

In order to get fastest recovery the log segments for each RAMCloud master must be spread equally across all of the backups in the cluster. However, there are many factors that complicate this method:

* Segment location should reflect failure modes. For example, in order to protect against top-of-rack switch failures and other problems that stop an whole rack, a segment’s master and each of its backups should exist in different racks.
* Different backups may have different bandwidth for I/O; segments must be spread so during recovery, each backup uses the same amount of time to read its share of the data.
* All of the masters are writing segments concurrently; they should manage to prevent overloading any individual backup. Backups have limited buffer space.
* Storage servers are constantly entering and leaving the cluster, which might unbalance the distribution of segments and changes the pool of available backups.

Decisions such as duplicate segment placement in a centralized way on the manager would limit RAMCloud’s scalability. For example, per second a cluster with 10,000 servers could back up 100,000 or more segments; this easily could cause the manager to become a performance obstacle (Ongaro et al., 2011).

Instead, each RAMCloud master chooses to place each replica independently, by using a mixture of randomization and refinement. So “when a master needs to select a backup for a segment, it chooses several candidates at random from a list of all backups in the cluster” (Ongaro et al., 2011). Then it chooses the top candidate, using its knowledge of where it has previously assigned segment replicas and information about the speed of each backup’s disk. The top backup considers being the one that can read its share of the master’s segment replicas from disk through recovery. If the backup is in the same framework as the master or any other replica for the current segment then the backup is rejected. When a backup has been chosen, the master contacts that backup to store space for the segment. At this point the backup would reject the request if it were overloaded, in which case the master chooses another candidate (Ongaro et al., 2011).

Using randomization can remove pathological actions such as all masters selecting the same backups in a lock step way. In fact, adding the modification step offers a solution almost as ideal as a centralized manager. Backups will have 8 segments on average when a master scatters 8,000 segments across 1,000 backups using a purely random method. Though, several backups are probable end up with almost around 20 segments, which will result to unequal disk use through recovery. Therefore, adding a small amount of choice makes the segment delivery almost uniform and also lets compensation based on other factors such as disk speed. This technic also deals with the entry of new backups: existing backups do not get selected as much as a new backup, till every master has taken full benefit of it (Ongaro et al., 2011).

RAMCloud masters mark one of the copies for each segment as the main copy. Through recovery only the main copies are read (except if they are unavailable).

Taking into consideration the possibility of storing one of the backup copies on the same machine as the master. This might decrease network bandwidth requirements, however, it has two disadvantages. First, it might decrease system fault tolerance: one copy is already in the master’s memory; a little benefit will be provided in placing a second copy on the master’s disk. So, if the master crashes, the disk copy will be lost beside the memory copy; it would only add a value in a cold start after a power failure. Second, storing one copy on the master might limit the burst write bandwidth of a master to the bandwidth of its local disks. In contrast, with all copies scattered, a single master can possibly use the disk bandwidth of the entire cluster (Ongaro et al., 2011).

## 5.3. Failure Detection:

There are two ways that RAMCloud can detect server failures. First, if a server fails RAMCloud clients will notice by responding to a remote procedure call. Second, even in without the client activity, RAMCloud checks its own servers to sense any failures; this will allow RAMCloud to replace lost copies before several crashes cause permanent data loss. Each RAMCloud server will regularly issue a ping Remote Procedure Call (RPC) to another server randomly selected and reports failures to the coordinator. For clusters with 100 or more nodes the probability of detecting a crashed machine in a single round of pings is almost 63%; within five rounds the odds are greater than 99% that a failed server will be detected (Ongaro et al., 2011).

In either case, server failures are being reported to the manager. The manager confirms the problem by attempting to communicate with the server, and if the server does not respond it initiates recovery. Timeouts must be fairly very short (tens of milliseconds) so that they do not significantly delay recovery (Ongaro et al., 2011).

## 5.4. Recovery Flow:

The recovery process have been supervised by the manager, which proceeds in three phases:

1. Setup. The manager finds all copies of all log segments belonging to the crashed master, chooses recovery masters, and assigns each recovery master a divider to recover.
2. Replay. Recovery masters fetch log segments in parallel and incorporate the crashed master’s partitions into their own logs.
3. Cleanup. Recovery masters begin serving requests, and the crashed master’s log segments are freed from backup storage (Ongaro et al., 2011).

## 5.5. Consistency:

Consistency is one of the unique characteristics that are in RAMCloud, which provides a strong form of (Linearizability, which needs exactly-once semantics), network partitions and even across host failures. Consistency affects crash recovery in two ways. First, a master that is suspected to fail must stop servicing requests before it can be recovered, in order to ensure that applications always read and write the latest version of each object. Second, when recovering, RAMCloud must ensure that only one manager can manipulate and serve the cluster’s configuration at a time (Ongaro et al., 2011).

When it starts recovery RAMCloud will disable a sick master’s backup operations, so the failure master will be required to contact the manager to continue servicing writes. The manager contacts backups at the start of the recovery to locate a copy of every segment in the failure master’s log, even the active segment to which the master may still be writing. When a backup with a copy of the active segment has been contacted, it will reject backup operations from the failure master with a warning that the master must stop servicing requests until it has contacted the manager. Masters will regularly check in with their backups, so disabling a master’s backup operations will also stop it from servicing read requests by the time recovery completes (Ongaro et al., 2011).

By using the ZooKeeper service manager failures will be handled safely. The manager uses ZooKeeper to store its configuration information, which contains of a list of active storage servers along with the tablets they manage. ZooKeeper has its own replication mechanisms to deliver a high level of availability and durability for this information. In order to handle manager failures, the active manager and additional standby managers will contest for a single manager lease in ZooKeeper, which it will ensure that one manager runs at a time. If the active manager fails, its lease will end and it will stop servicing requests. A random standby manager will obtain the lease, read the configuration information from ZooKeeper, and continue the service. The configuration information is small, so the manager failures recover as quickly as other server failures (Ongaro et al., 2011).

# 6. RAMCloud Disadvantages:

The most obvious disadvantages of RAMClouds are high cost per bit and high-energy usage per bit. For both of these facts RAMCloud will be 5-10x worse than a storage system based on flash memory and 50-100x worse than a clean disk-based system. Also, RAMCloud needs more space in a datacenter than disk or flash memory. Thus, RAMCloud is not a good solution, for large amount of data stored inexpensively and the access rate is low.

However, applications with high throughput requirements, RAMClouds will be the best solution. Measuring RAMClouds based on the cost and the energy per operation, they are 100-1000x more efficient than disk-based systems and 5-10x more efficient than systems based on flash memory. However, RAMCloud will provide high performance with energy efficiency for systems with high throughput requirements. DRAM chips offer low-power mode, which it makes it possible to reduce RAMCloud energy usage during low activity periods.

In fact, applications with data duplication across datacenters will not benefit from RAMCloud's advantages. In this kind of environments “the latency of updates will be dominated by speed-of-light delays between datacenters, so RAMClouds will have little or no latency advantage.” (Ousterhout et al., 2011) Moreover, RAMCloud stronger consistency will be hard to achieve for cross-datacenter duplication. However, RAMClouds will still offer extraordinarily low latency for reads even with cross-datacenter duplication (Ousterhout et al., 2011).

# 7. Conclusion:

RAMCloud is a research in achieving low latency at large scale: the goal is to build a storage system that provides the fastest access to the largest datasets. As a result, DRAM is the main location for the data, and it combines the main memories of thousands of servers to support large-scale datasets. RAMCloud uses many techniques, to managing all storage, RAMCloud uses a uniform log-structured tool, using a polling approach to communicate directly with the Network Interface Controller (NIC), and a method to availability that substitutes fast crash recovery for online replication. This system more than 1000x faster than the disk based storage systems that have been the acceptable status for most of the last four decades. The two most important features of RAMClouds are: first, the extremely low latency, second, the ability to aggregate the resources of large numbers of commodity servers. Together, these able RAMClouds to scale to meet the needs of the largest Web applications.

The key in fast recover crashes in RAMCloud is the use of the large-scale storage system. RAMCloud allocates backup data across a large number of secondary storage devices and uses both data pipelining and parallelism to achieve end-to-end recovery times of less than 2 seconds. Fast crash recovery is a key enabler for RAMCloud: it allows a high-performance DRAM-based storage system to provide durability and availability at one-third the cost of a traditional method using online copies.

RAMCloud method is taken to the extreme, such as using DRAM instead of flash memory for storage and designing the system to support at least 10,000 servers.

# References:

Goda, K., Kitsuregawa, M. *The History of Storage Systems.* 2012

IBM (2010, July 15) *IBM 350 disk storage unit***.** Retrieved from<http://www-03.ibm.com/ibm/history/exhibits/storage/storage_350.html>

Mohamed, A. (2010, March 12). *A history of cloud computing.* Retrieved from <http://www.computerweekly.com/feature/A-history-of-cloud-computing>

Ongaro, D., Rumble, M. S., Stutsman, R., et al. *Fast Crash Recovery in RAMCloud.* Dissertation, Stanford University Library. 2011

Ousterhout, J., Agrawal, P., Erickson, D., et al. *The Case for RAMClouds: Scalable High-Performance Storage Entirely in DRAM.* 2011

Stutsman, R. S. *Durability and Crash Recovery in Distributed In-Memory Storage Systems.* Dissertation, Stanford University Library. 2013

Yifeng Luo et al. *RAMCloud Storage System based on HDFS: Architecture, implementation and evaluation*. 2012