DSC-204A: Scalable Data Systems, Winter 2024

Lecture Title: Cloud Computing Introduction

Lecturer: Hao Zhang *Scribe: Anikait Sunil Nair, Nived Neetha Sooraj, Mounika Padala*

1 Introduction

We enter a new era, i.e., cloud computing. The intention is a high-level introduction to cloud computing, like why it happens and why it is better than any other alternative. As important as the components of cloud computing, it is crucial to know why and how cloud computing works.

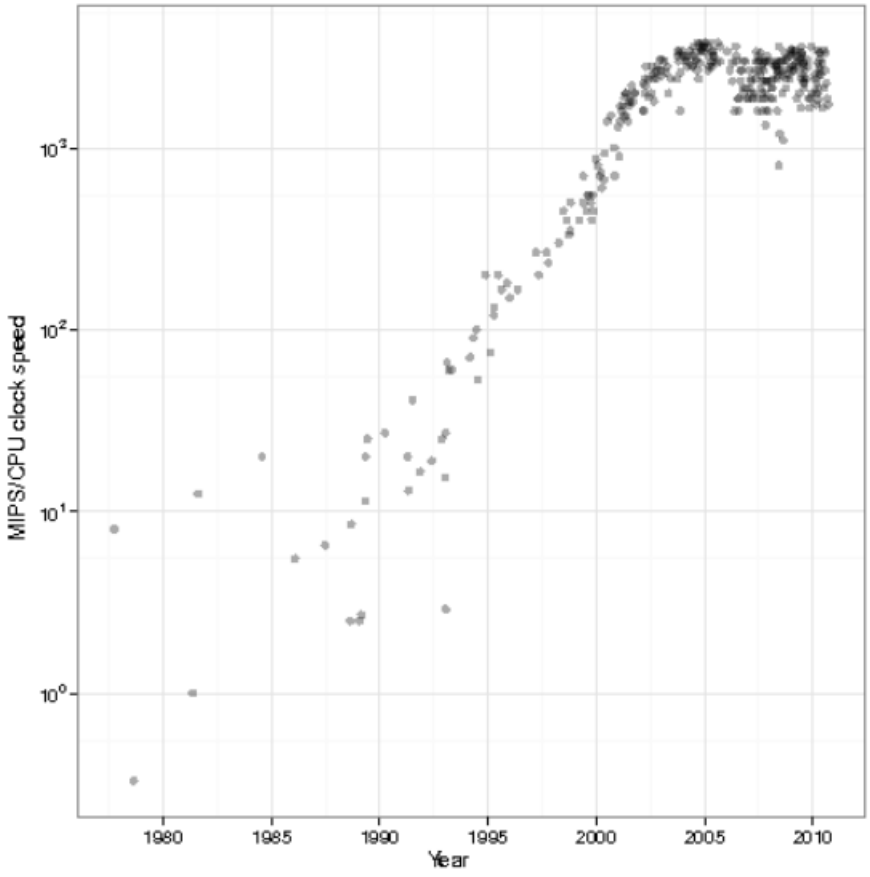
An overview of Cloud Computing and Distributed Systems contains: Introduction to cloud computing which will include system components of cloud computing which include networking, distributed storage, and file systems as well as the computing of data in a distributed way across multiple nodes through which we try to introduce parallelism and consistency. Finally, there will be some advanced topics in the realm of cloud computing and distributed systems.

2 Main Agendas

Why cloud computing and why is it required over some other mechanism: Need-Based Argument: We need more computation and cloud computing can offer and meet our computation requirements. Utility-Based Argument: There are many other methods that we can use for the process of computation, but cloud computing is one of the most effective ways. It strikes a balance between cost and performance.

3 Background of Cloud Computing

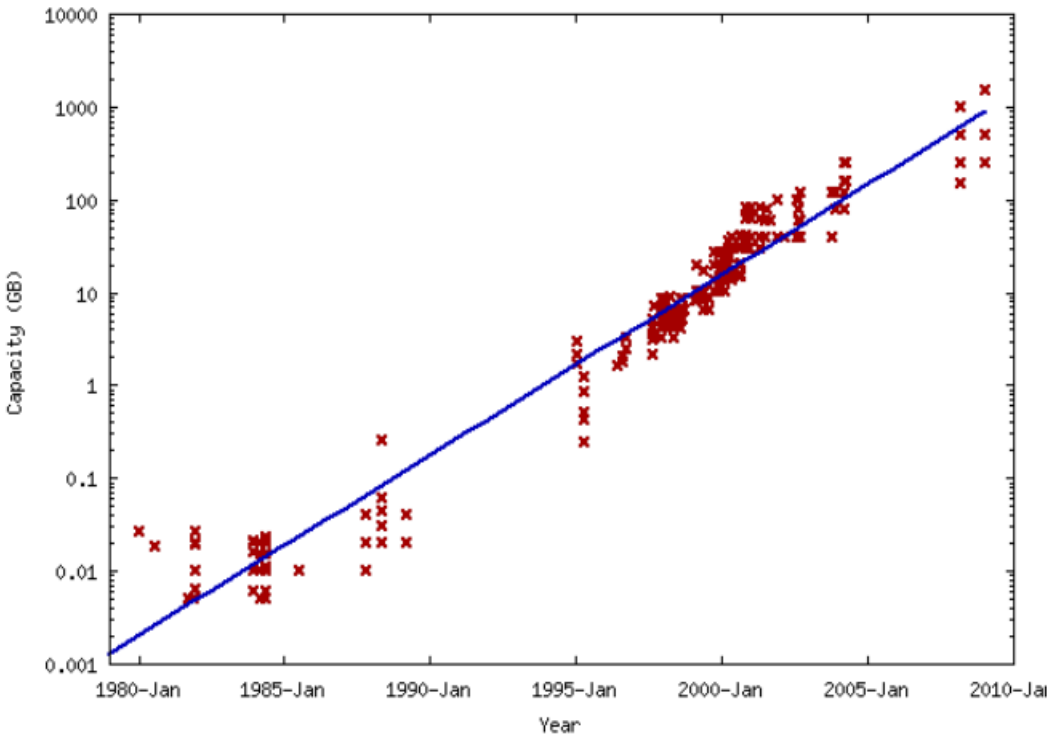
At the start of the 1990's the most popular time for parallel computing and multi-processing. Companies like Intel and AMD made leaps in performance with a growth of about 52% per year. As a software developer, the amount of work put into these processors is minimal, and yet, we see the performance grow year by year. In 2002, we hit a thermal wall or a limit where we were not able to make one core much stronger or faster than before, i.e., we could not fit more things into one transistor. An alternative that was to fix this issue was to be able to use more cores. Adding one more would linearly speed up the program. Intel and AMD implemented these methods and we got into the phase of the multicore revolution, where we had 2-core, 4-core, and much more. Nowadays we are able to see systems that are 16 or 32-cores. A surprising fact is that even though these changes have occurred, the prediction was that this change had to happen about 15-20 years earlier.



4 Data Explosion

On the other hand, during this same period, the amount of stored data is exploding. As an example, we were introduced to the concept of smartphones and social media by various companies. An issue that popped up in the process was that the amount of data that these social media sites generated was huge. and so if we had a certain requirement such as finding a good hotel or restaurant nearby, we had to scan through a humungous amount of data and find insights, which made the task very complex for a single core. Billions of users connected through the internet through social media and the World Wide Web. Another pattern that we saw was that storage was getting cheaper because the chips were getting cheaper and so the amount of data that was being stored grew quickly. But to process this data, we needed more computation power and therefore, more flops.

A problem that we saw was that our single-core and multicore devices reached a limit at one point but the data didn't stop growing. This created a dilemma on how to process the quickly growing data in our hands. The solution to this was to just use more computers and make use of distributed and parallel computing, using a network of devices to solve one task in the hope that it'll take a shorter amount of time.



5 World of Distributed Computing

In the case of distributed computing, instead of using bigger computing, we try to use a loosely coupled set of computers, communicating through message passing, and solving a common goal. An example of this is scouring through our entire social media network and getting information such as a list of our connections and so on. But distributed computing comes with certain challenges too. Just like in the case of humans trying to handle a large group of people, as the number of computers increases in the case of parallel computing, it gets tougher to handle all these devices. Some of the devices may not follow your instructions completely and this may cause a case of partial failure which will affect the process as a whole. Another issue that is seen is asynchrony. When assigned several tasks that have to be done, the process of consistency and synchronization is key. We may see cases where if a task that has to be done initially is not completed, it naturally slows down the tasks that are in queue or have to be done next. This takes a toll on the entire processing procedure. Another aspect is that achieving the state of synchrony is also a very expensive task. There has to be a lot of discussion or norms that have to be set to achieve this synchrony, which is a time-consuming process. Cases of asynchrony will lead to the systems having to go back and solve the task properly again, which leads to extra computation.

Now, we focus on the difference between distributed computing and parallel computing. Parallel computing is what we consider to be the process of having multiple cores or various systems that are compatible and similar. In this case, we hope that the communication between these systems is much better and this is because we expect systems of similar type to be able to communicate better and make progress at the same pace. In the case of distributed computing, it is not necessary to have very similar systems. Here we can see computers that are very low-ended being part of the system too and there is a process of coordination that has to be taken care of so that the task gets completed properly. The reason that we use this method is that it is much cheaper. So in the process of distributed computing, we see a lot of partial failures along

with partial computing but we expect to get the task done in a much cheaper manner.

6 Dealing with Distribution: Programming

A scenario that we also see play out is that in the case of single-core computers, we write code for a task and it gets executed easily. But in the case of distributed systems, there are various computers in play. So when writing code for a task, we have to write it in a format that is acceptable for all the systems, and the form of communication between these systems has to be taken care of too. This communication has to take place at the right time as constant communication may lead to slowing down of processes.

So we see that programming for distributed systems is a tough task. A few ways to program for these distributed systems so that the programming is consistent are Message Passing Interface (MPI), Distributed Shared Memory (DSM), and Remote Procedure Calls (RPC). But even though all these models can be used, distributed programming is still very hard, facing issues such as programming for scale, fault tolerance, consistency, and so on.

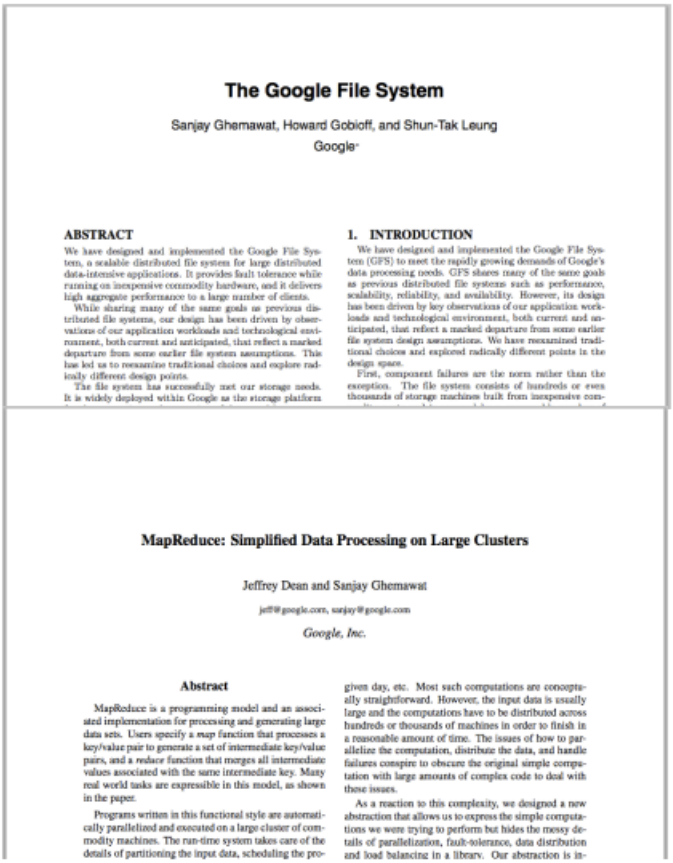
7 Recap: Basics of Computer Organization

Now let's do a basic recap of computer organization. In a single computer, we have storage, disks, and memories for storage and retrieval of data. We have processors, both CPU and GPU for processing data and networks for remote retrieval of data. Now if we make this a distributed system of computers, we use distributed storage and disks where we have many disks that are interconnected and data has to be stored and retrieved through these disks. We have distributed and shared memory where each system has a memory of its own and this has to be managed. Distributed GPU and CPU are also a part of distributed systems as well as networks which are needed to connect these devices.

Just as with the hardware, we need to look into the software changes for a distributed system, i.e., if data centers are the new systems, then what is the situation of the operating system? In the case of classical operating systems, for data storage and sharing, we have files, inter-process communication, and so on. For programming abstraction, we have system calls, APIs, libraries, etc., and in the case of multiplexing go resources, we have scheduling techniques, virtual memory, file systems, and so on. In the case of distributed systems, we have to find a distributed version for all of this. In the case of distributed systems, for data sharing, a technology called key/value storage is used which takes care of data storage and leads to the introduction of data warehouses. For programming abstractions, we have to take care of multiple devices for which there are some famous technologies used such as MapReduce, PIG, Hive, Spark, Ray, etc. Similarly, for multiplexing resources, we have technologies such as YARN, ZooKeeper, BookKeeper, K8S, and so on.

8 Pioneer: Google Infrastructure

A pioneer in the technology of cloud computing and distributed systems is Google. In 2003, Google brought about their Google File System (GFS), which helped them scour the entire internet for information and store it so that it could act as a distributed file system for the entire cluster. in 2004, Google introduced Google MapReduce, which ran queries or jobs on this data. It was also used to manage work distribution and fault tolerance. It was colocated with the file system. Other systems in the market were Apache open-source versions such as Hadoop DFS and Hadoop MR.



The Google File System

Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung

Google*

ABSTRACT

We have designed and implemented the Google File System, a scalable distributed file system for large distributed data-intensive applications. It provides fault tolerance while running on inexpensive commodity hardware, and it delivers high aggregate performance to a large number of clients.

While sharing many of the same goals as previous distributed file systems, our design has been driven by observations of our application workloads and technological environment, both current and anticipated, that reflect a marked departure from some earlier file system assumptions. This has led us to reexamine traditional choices and explore radically different design points.

The file system has successfully met our storage needs. It is widely deployed within Google as the storage platform

1. INTRODUCTION

We have designed and implemented the Google File System (GFS) to meet the rapidly growing demands of Google's data processing needs. GFS shares many of the same goals as previous distributed file systems such as performance, scalability, reliability, and availability. However, its design has been driven by key observations of our application workloads and technological environment, both current and anticipated, that reflect a marked departure from some earlier file system design assumptions. We have reexamined traditional choices and explored radically different points in the design space.

First, component failures are the norm rather than the exception. The file system consists of hundreds or even thousands of storage machines built from inexpensive com-

MapReduce: Simplified Data Processing on Large Clusters

Jeffrey Dean and Sanjay Ghemawat

jeff@google.com, sanjay@google.com

Google, Inc.

Abstract

MapReduce is a programming model and an associated implementation for processing and generating large data sets. Users specify a *map* function that processes a key/value pair to generate a set of intermediate key/value pairs, and a *reduce* function that merges all intermediate values associated with the same intermediate key. Many real world tasks are expressible in this model, as shown in the paper.

Programs written in this functional style are automatically parallelized and executed on a large cluster of commodity machines. The run-time system takes care of the details of partitioning the input data, scheduling the program

given day, etc. Most such computations are conceptually straightforward. However, the input data is usually large and the computations have to be distributed across hundreds or thousands of machines in order to finish in a reasonable amount of time. The issues of how to parallelize the computation, distribute the data, and handle failures conspire to obscure the original simple computation with large amounts of complex code to deal with these issues.

As a reaction to this complexity, we designed a new abstraction that allows us to express the simple computations we were trying to perform but hides the messy details of parallelization, fault-tolerance, data distribution and load balancing in a library. Our abstraction is in-

9 Question to Ponder

An important question among all this that creates a lot of discussion is that even though Google has pioneered and created many distributed systems and technologies that shape today's cloud computing, why is it that Amazon and even Microsoft win over Google Cloud in Cloud computing market shares? One of the arguments that can be made is the ease of use in the case of Amazon cloud products as compared to Google. Google as a company has tended to focus on the technological aspect while Amazon, who had internally built their cloud system has made it very easy to use for their engineers as well as the users and this is something that attracts more users towards Amazon as compared to Google. One example is that Amazon is a great preacher of the system of macroservices. So macroservices is the case of using a lot of abstraction and building many services that talk to each other so that you can distinguish roles for each service, which will eventually help you get your task done. Google on the other hand does not use these different service methods and rather just puts them all into one place to get the task done. What happens here is that not every user wants to know everything and use every service. Some users have specific needs like just wanting security, just wanting memory, and so on. In such a case, the technology of Amazon gains more prominence among the audience as compared to Google. Looking into stats, the usage of Amazon is about 70%, and while Microsoft is growing to about 20%, Google still stays at about 10% usage even though putting forward more technologies.

Do your own study on this, this is important because this course lets you analyze the technology trend and

make sure you are spending your time on the right thing.

Are Google earning from Royalty because they already made these?
Previously they are not because they are making a lot of things open source. Recently Google started making profits from GCP. But it's mostly not because of the old stuff like distributed data processing and its mostly because of Machine Learning and stuff.

Sky computing Goal of sky computing is to make cloud computing more of a commodity. Do you think a second computing will happen?

10 Comparison between traditional model and utility computing

A company needs distributed compute and storage.

First argument
Data Explodes so we need more compute and storage. Due to physical limitations we cannot make a single computer faster so we need distributed computing and this is very reasonable. Because we need distributed computing , we need the cloud because the cloud happened to provide all these resources. Not a solid argument because there are other ways to provide distributed computing, like supercomputers etc. Supercomputers are very powerful and as powerful as cloud or maybe even powerful. It has all those high end memory and internet connects. Then why is supercomputer today not as permanent as cloud computing? There are just a few supercomputer centers but there are so many clouds and everyone is putting their career on cloud.

Second Argument
Utility based argument So the paper for the reading summary 3, "A Berkeley View of Cloud Computing" dealt with utility computing. Consider a company needs more computer storage.

10.1 Traditional model

In traditional model we manage and store computers on premise. There are still a few silicon valley companies that bet on premise and say if you need more computing you should buy more computers. They do help their clients like banks and hospitals as they dont want their things to go to the cloud. They have a very strong sense of security. In a traditional model you are:

- Responsible for security as you are in control and you manage your security.
- Responsible for power.
- Responsible for network and hardware.

A person's demand for computer storage may increase and decrease and they may want to scale up and down. So in a traditional model we order or get more computers and connect them to networks . So if your computers are running on Microsoft or Apple and an update is rolled out you will need to manually perform updates on all the individual systems. You will need to incorporate security patches etc.

10.2 Utility Computing

It stemmed from the concept of public utility such as water or electricity. Consider everyday electricity usage:

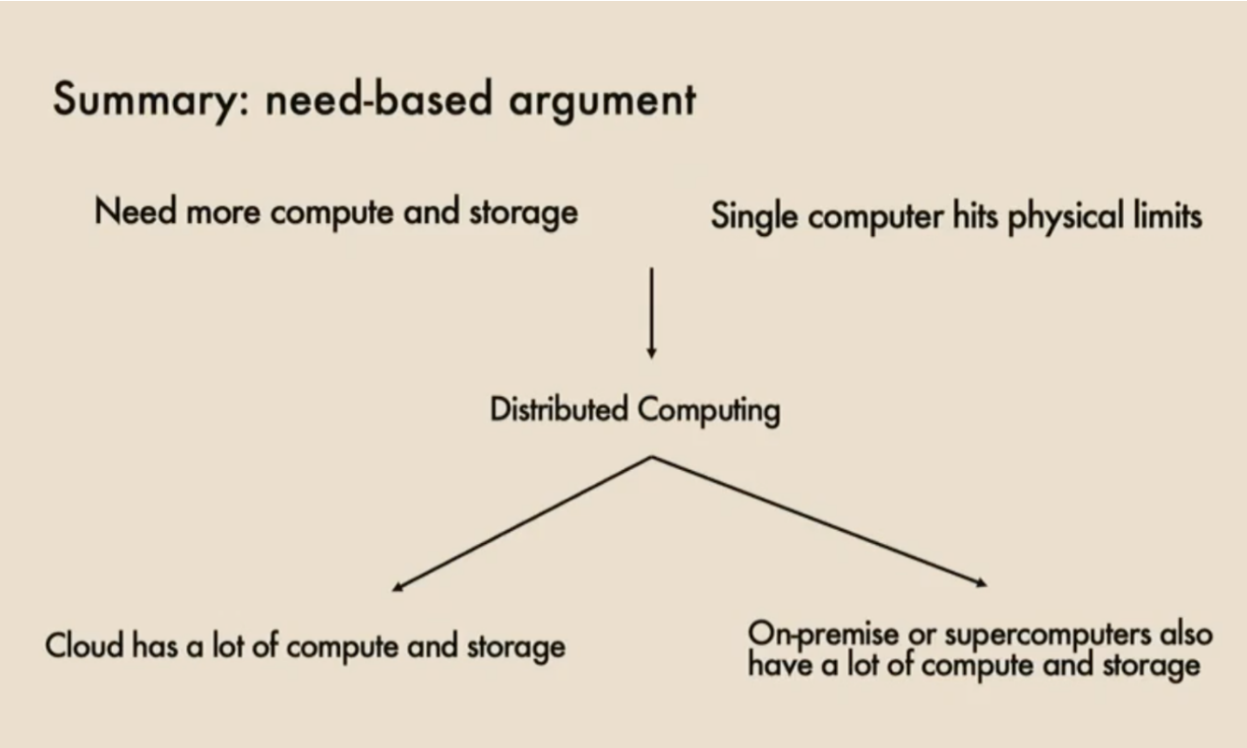


Figure 1: Your Caption Here

- Its summer, we turn on A/C
  - We do not need to notify the electric company that we need more electricity. It is just there.
  - We do not need to go to hardware stores to buy or install more generators.
- In spring,we turn off A/C
  - We do not need to notify electric company that we need less.
- In Winter, we turn on heater
  - The usage goes up and down, but we just use.
  - It is very flexible to scale up and down and we only pay for what we use.

Similarly in **Utility computing**

- the compute power is available on demand.
- We can scale up or down as needed.
- We dont need to determine our needs in advance.

Not the case any more for GPU market.

10.3 Summary

Traditional Model

- Determine needs in advance
- Overestimate -¿ unused compute
- Underestimate -¿ shortage and wanting.
- The company provides on-site security
- Company needs to provide backup power for emergencies.

Utility Computing

- Dont need to worry about accurately estimating needs.
- Pay for what is used
- Scale up and down
- Cloud infra company provides security
- Cloud Infra company provides emergency or fault tolerance.

So the second argument puts forth that utility computing is cheaper and cost effective for most use case and is highly flexible.

11 Cloud Computing

Cloud computing is the intermediate point before we actually and truely achieve utility computing. So basically in cloud computing:

- Compute, storage, memory, networking, etc are virtualized and exist on remote servers,
- Rented by application users and is maintained by the infrastructure companies.
- They built a lot of data centers.

The opposite of cloud computing is on-premise computing

- It refers to IT infrastructure hardware and software applications that are hosted on-site.
- Has great security.

11.1 Evolution of Cloud Infrastructure:

Basically cloud computing will be going on for 3 generations and we are currently at the second generation. Cloud 1.0 is basically the past and cloud 2,0 is present or the current state of cloud computing and cloud 3.0 is the ongoing research that people are doing.

On premise is like you own your own car but there is an intermediate point where you can rent a car and there is a more advance option were you can do car sharing, like you and your friends can share a car to go somewhere, or you and someone else that you dont know. In the second case there is still a chance that after renting a car you might end up not fully utilizing it. Like you rent a car and travel but when you are staying at your hotel during the travel, you are not using the car and you are actually wasting some time that you pay for. For uber you just call the car for as an on demand service for getting from point A to B.

So considering the evolution in cloud computing and comparing it with the car analogy, we are basically between the 2nd and 3rd server. We can do better than renting a car but we cant fully do car sharing.

11.1.1 Cloud 1.0 (Past)

We have network servers that are interconnected. User rents servers (time-sliced access) needed for data/software. People basically talk to the infrastructure company and rent the servers for example 7 days. Once you get the server its your responsibility to install the operating system and all the things that you need, copy your data and start doing computing. Once you finish computing you give it back. Usually they charge the entire per metal server and times the time it is used by you. For eg: Its 10 dollar per hour and you use it for 24 hours you will pay for the entire time you rented it that is 240 dollars. Even if you did not necessarily use it for the duration for the entire 24 hours.

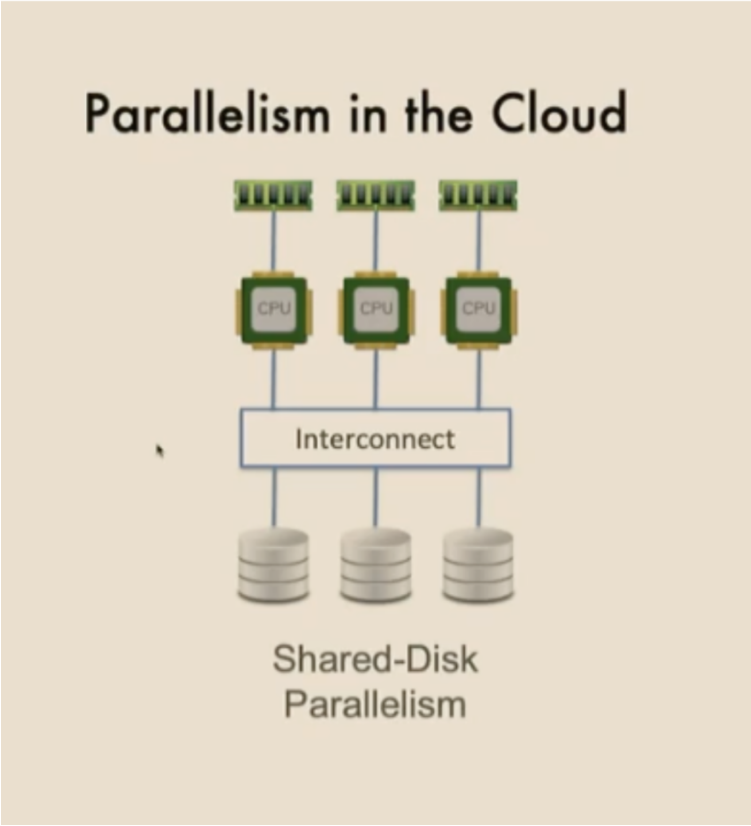
11.1.2 Cloud 2.0 (Current)

Instead of renting the experimental drivers you can basically make an allocation of a set of resources for your use and you run the resources, the abstract things and build a lot of virtual machines on top of your environment. The difference between 1.0 and 2,0 is basically you can install many virtual machines on a single environment and two users can use the same environment and you basically pay less. In summary the resource allocation is much more flexible.

Virtualization of networked servers. User rents amount of resource capacity (eg. memory, disk). Cloud provider has a lot more flexibility on provisioning (multi - tenancy, load balancing, more elasticity, etc.)

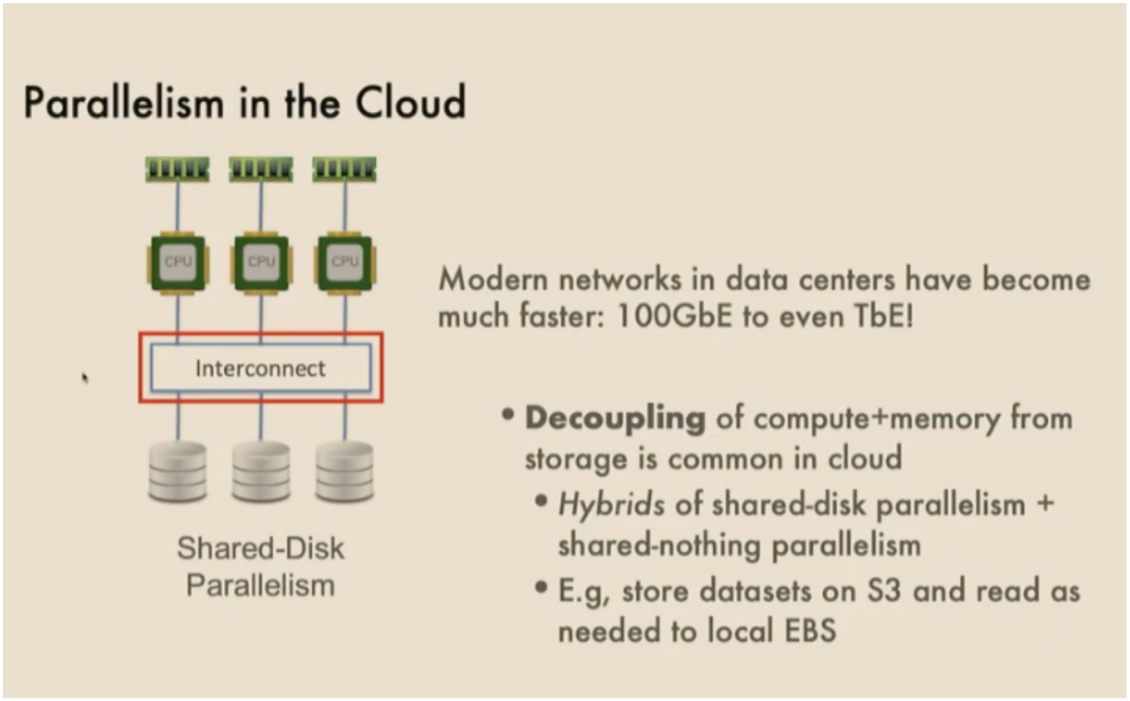
Paraellelism in cloud

Advanced technologies can interconnect all cores, disks and memory in a data center, they are fast like data center networking switches.



Now as you can see this cloud is pretty advanced, like there are a lot of advanced technologies that can basically interconnect disks and memory. They make pretty fast data center networking switches.

- All these cores are connected abstract to build a bigger data center.
- Any piece of software can be installed, with different requirements as per their need for different capability and capacity
- In cloud, we decouple compute and memory from storage and hybrids of shared-disk parallelism and shared-nothing parallelism.



They just install many many virtual machines on top of these data centers. A virtual machine is just a software concept and it can be very flexible. For example I can install a virtual machine which includes 1 core, 1 disk and 1 memory but I can also arbitrarily adapt that number into any number that will provide a different capability and capacity.

11.2 Cloud 3.0 (Ongoing Research)

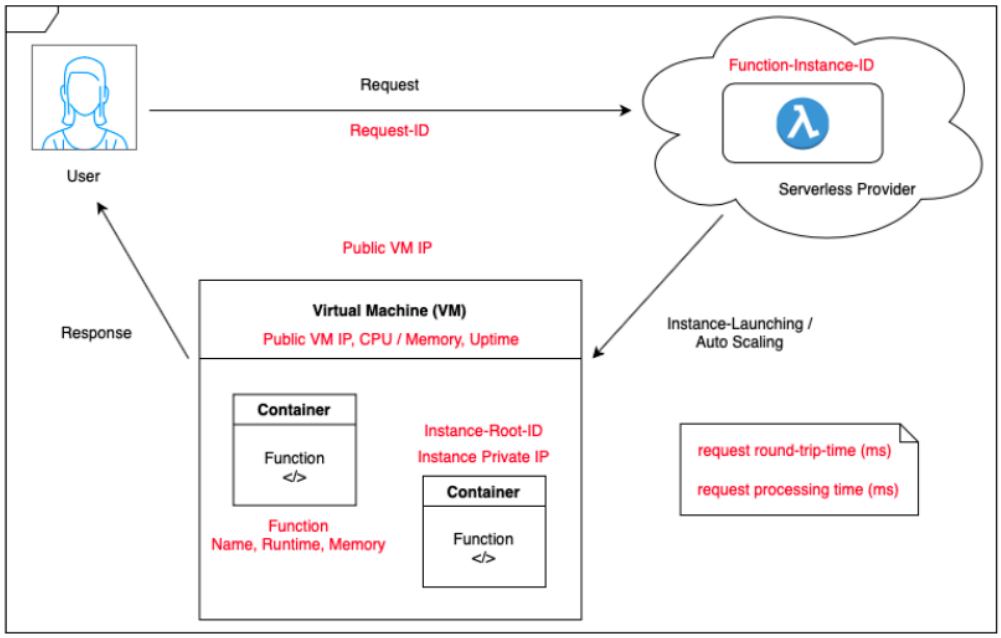
Research Focus - how to have better performance than existing Cloud (2.0)

Solution - Resource Disaggregation

- Usually, when users open an application, their goal is to complete a task. Traditionally, they have the hardware resources with them to complete them.
- Now, when the cloud 2.0 comes into the picture, they have to request specific resources to complete their task. However, giving an exact estimate of the resources needed is challenging. For example, if a person wants to complete a deep learning task, it would be hard for them to provide an exact number of cores or disk space required to request.
- So, Research focuses on providing a function that understands the user's objective, schedules, and runs this function on the data center, charging only for the resources used. This eliminates the need for users to explicitly specify hardware resource requirements, offering a completely abstract experience without concerns about the operating system or specific resources.
- This Function-as-a-service (Faas) is achieved by resource disaggregation. Data center computing is

evolving into a fully disaggregated architecture, with separate data centers for different resources like disks and memory, interconnected by complex network switches that allow for arbitrary communication.

- When users submit a function (as a protest), it is scheduled as a Job. The task is executed across the data center's distributed resources and completed efficiently. Users are charged based on the resources they actually used for the task.
- It is also called sererverless because, now my task is not run by a single computer (server), instead it takes my task as function, as a job, and runs it on different data center resources.



If you are interested in Resource Disaggregation - There are some professors in UCSD who do extensive research on disaggregation results and they publish papers on it, and ship their technology to AWS, so that its adopted.

11.2.1 New Cloud renting Paradigms :

Nowadays, most cloud companies are adopting the Cloud 3.0 approach.
However, from the cloud vendor's perspective, there are other approaches to charging as well. For example, at AWS, there are three main pricing models:

- **On-demand:** This option is for immediate needs, where users pay for their current requirements.
- **Reserved:** Users can forecast their future needs and request resources in advance. Cloud providers reserve these resources for the user, who in turn benefits from lower rates compared to on-demand prices because they are provisioning in advance.
- **Spot:**
  - In the context of operating systems and preemption (where a lower priority task is stopped when a higher priority task comes in), the cloud operates similarly.

- When a user (who has paid more - higher priority) comes in, the task of a lower priority user is stopped.
- The provider tries best to schedule the task but does not guarantee that the task will always run.
- This option is suitable for users who are not in a hurry, can save their work progress every time their task is preempted, and make checkpoints.
- From the cloud provider's perspective, when they have idle resources, they can offer them as spot instances. When there is high demand, they stop these spot instances and make the resources available for on-demand users. This approach helps avoid idle time and earn more.
- There is an entire **Spot market business** behind this. They guarantee that for the same spot rate, users receive an on-demand experience. They try to build another layer on top of all these spot instances. This layer continuously seeks spot instances at different costs, choosing the lowest spot instance cost at any time. If it is taken away, then they submit another request for the user. It's essentially a scheduling problem that they are trying to solve.
- Even within few companies, they build layers that schedule jobs on spot instances to minimize costs.

Users can choose from these different provisions and tiers offered by the cloud vendor.

These are some more differences in the table below, which distinguishes between on spot and on-demand.

	Spot Instances	On-Demand Instances
Launch time	Can only be launched immediately if the Spot Request is active and capacity is available.	Can only be launched immediately if you make a manual launch request and capacity is available.
Available capacity	If capacity is not available, the Spot Request continues to automatically make the launch request until capacity becomes available.	If capacity is not available when you make a launch request, you get an insufficient capacity error (ICE).
Hourly price	The hourly price for Spot Instances varies based on demand.	The hourly price for On-Demand Instances is static.
Rebalance recommendation	The signal that Amazon EC2 emits for a running Spot Instance when the instance is at an elevated risk of interruption.	You determine when an On-Demand Instance is interrupted (stopped, hibernated, or terminated).
Instance interruption	You can stop and start an Amazon EBS-backed Spot Instance. In addition, the Amazon EC2 Spot service can interrupt an individual Spot Instance if capacity is no longer available, the Spot price exceeds your maximum price, or demand for Spot Instances increases.	You determine when an On-Demand Instance is interrupted (stopped, hibernated, or terminated).

11.3 Advantages and Disadvantages:

- **Cloud 1.0:** This model is similar to renting. It offers perfect isolation, as it reduces the need for your own environmental server but comes at a higher cost.
- **Cloud 2.0:** This version is more affordable than Cloud 1.0, yet it still presents some challenges, particularly in estimating the exact resources needed to complete a job. Users must still specify resources, such as the amount of memory, though not as precisely as in the past.
- **Cloud 3.0:** Currently under development, this model aims to be fully **serverless**, charging users only for the amount of computing used for their jobs. Achieving this is challenging due to the need for virtualization and multiple layers of it. Isolating different jobs submitted by users to the same pool

of resources poses a significant challenge, especially when considering the security demands of some users.

11.4 Recap: Cloud over on-premise cluster

Virtualization: Cloud computing utilizes virtualization for resources like compute, storage, memory, and networking, which are hosted on remote servers and rented to application users.

Pros of Cloud Computing:

- Manageability: A major advantage of cloud computing over on-premise clusters is that the management of hardware is not the user's responsibility.
- Pay-as-you-go: Cloud services offer fine-grained pricing economics, allowing users to pay based on actual usage, with granularity ranging from seconds to years.
- Pay-as-you-go: Cloud services offer fine-grained pricing economics, allowing users to pay based on actual usage, with granularity ranging from seconds to years.
- Elasticity: The cloud provides the ability to dynamically add or reduce capacity based on the demands of the workload.
- Service Models: Cloud computing offers various service models, including Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS).

Profit Chain- Now a days, many companies like aws, azure, gcp are exapnding their data centers, they are buying more and more GPUs from NVIDIA; NVIDIA is thereby making huge profits, when the consumers scale up.

However, there is a trend of building on-premise super computer again. Tesla's Dojo (Super Computer). read more at link.