Cloud Computing History

CIS437
Erik Fredericks // frederer@gvsu.edu

Adapted from Google Cloud Computing Foundations, Overview of Cloud Computing (Wufka & Canonico)

First, a thought-provoking question

Why bother with the cloud?

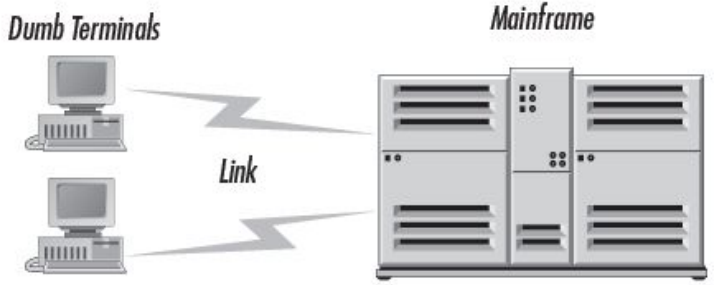
From mainframes to cloud

Initially, computing was handled via dumb terminals and a central server

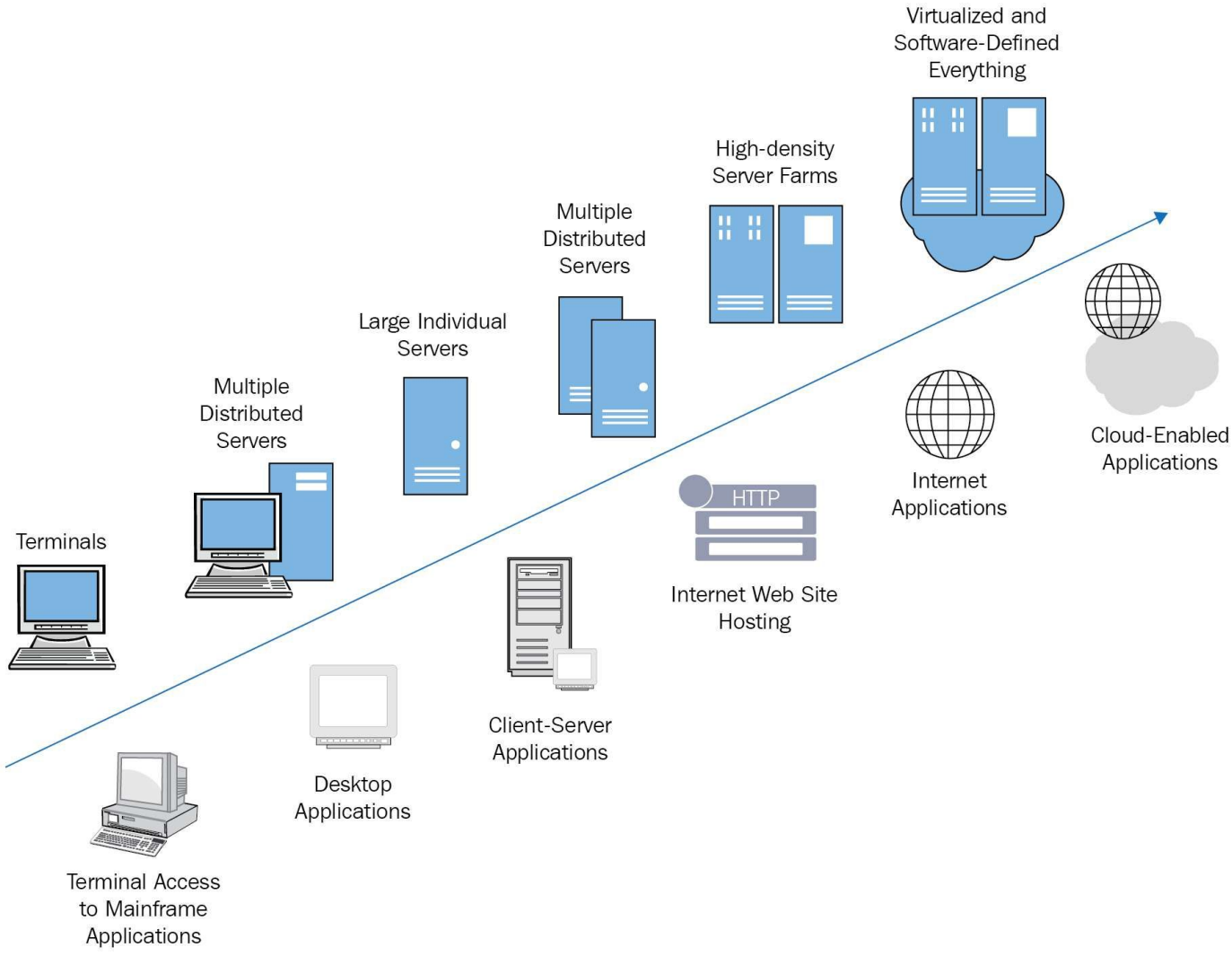
- Meaning, your terminal couldn't do anything

- All the work was handled by some remote system

  - Sound familiar?



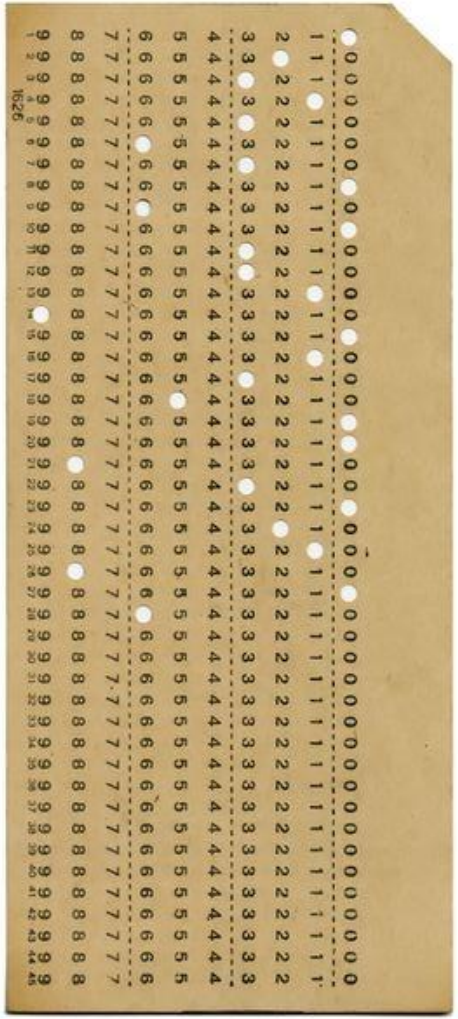
What are some possible disadvantages to mainframes?


Virtualized and Software-Defined Everything
High-density Server Farms
Multiple Distributed Servers
Large Individual Servers
Multiple Distributed Servers
Terminals
HTTP
Internet Web Site Hosting
Internet Applications
Cloud-Enabled Applications
Client-Server Applications
Desktop Applications
Terminal Access to Mainframe Applications

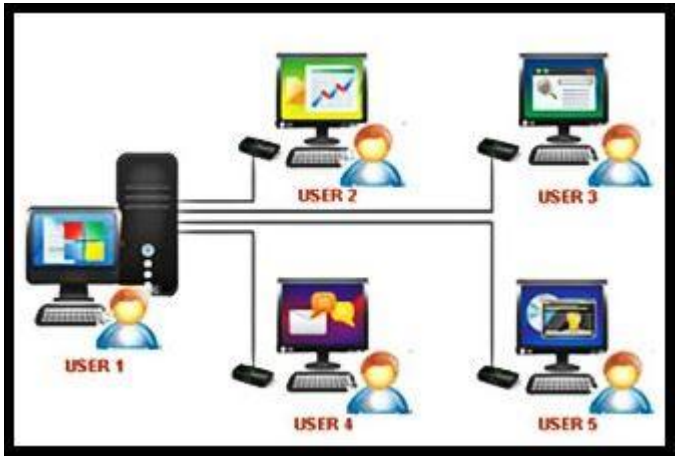
What are some possible disadvantages to mainframes?

- I/O limited
  - Punch card readers
  - One user allowed at any given time
  - CPU can't do anything during I/O

Also, they were *huge*



Multi-user systems



First, multi-user I/O

- Or, more than one user can do things at a time

- Then, time sharing
  - Multiple programs active in memory
  - OS handles active 'time slices'
  - We'll leave that to your OS class

- Then, PCs and C/S

Personal computers

i.e., what you're using now

- GASP

Pre-recent years, could others login and use your computer?

What about now - can somebody remote in and use your machine?

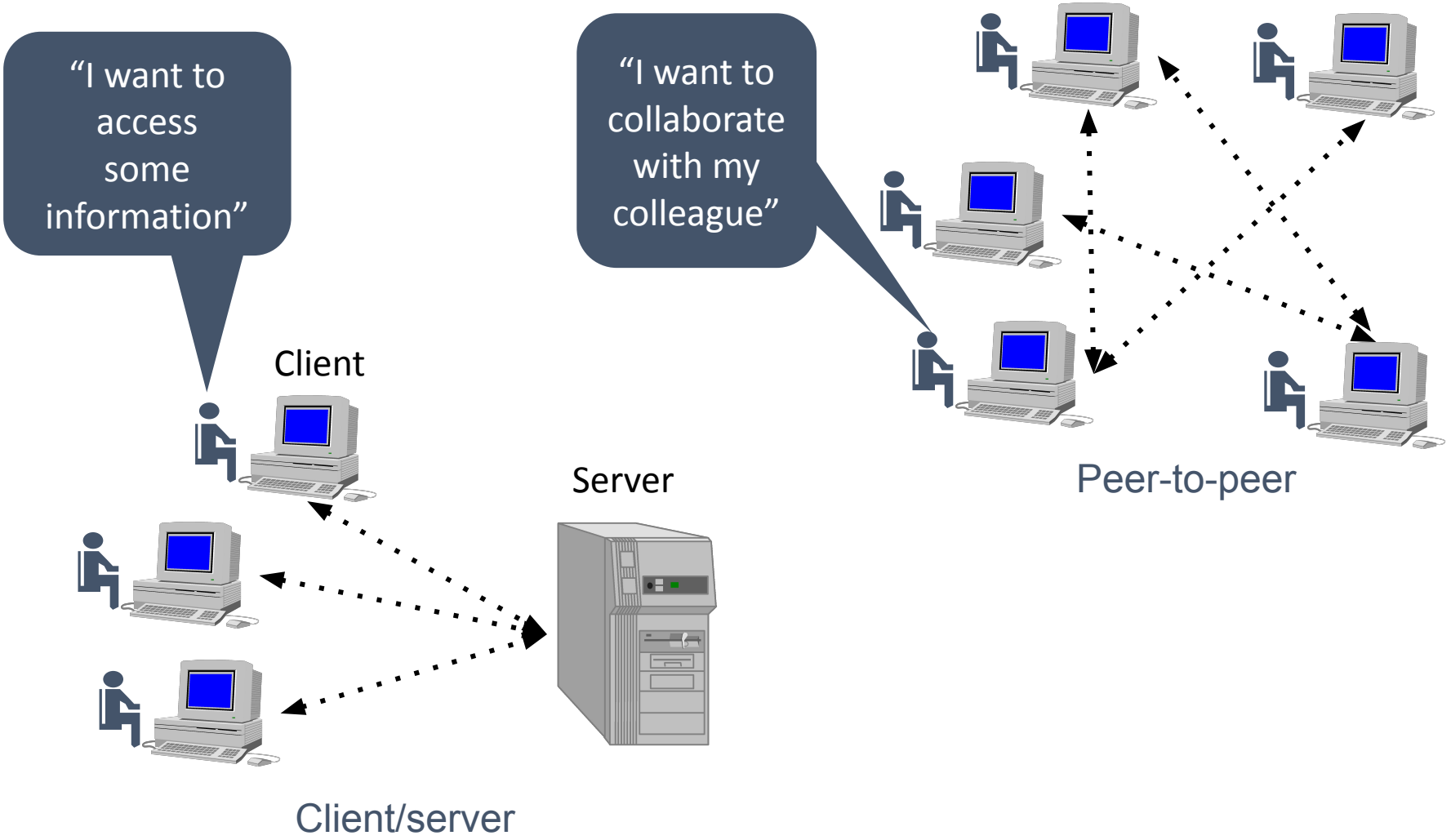
- If so, how?

Regardless, history you probably already knew about

- What about client/server (C/S)?

(a small number of 373 slides if you don't mind)

(also, pretty cool class imo)


"I want to access some information"
Client
Server
Client/server
"I want to collaborate with my colleague"
Peer-to-peer

Architecture types

Client-server

- Asymmetric relationship
- Client makes requests, server makes replies

Peer-to-peer (P2P)

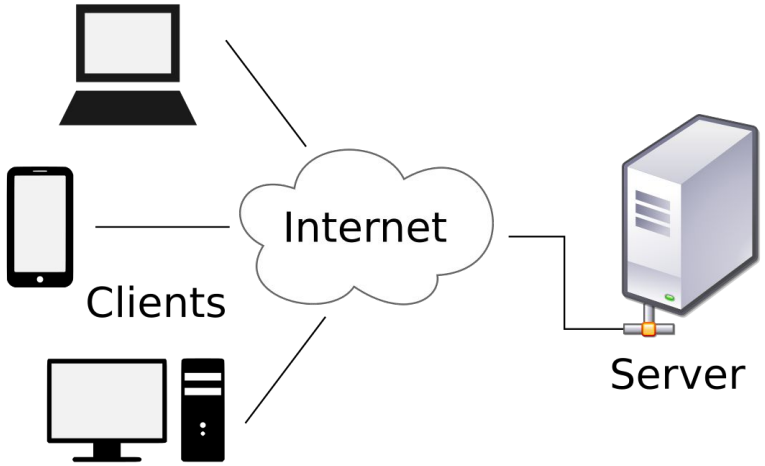
- Symmetric relationship

Client/Server Types???

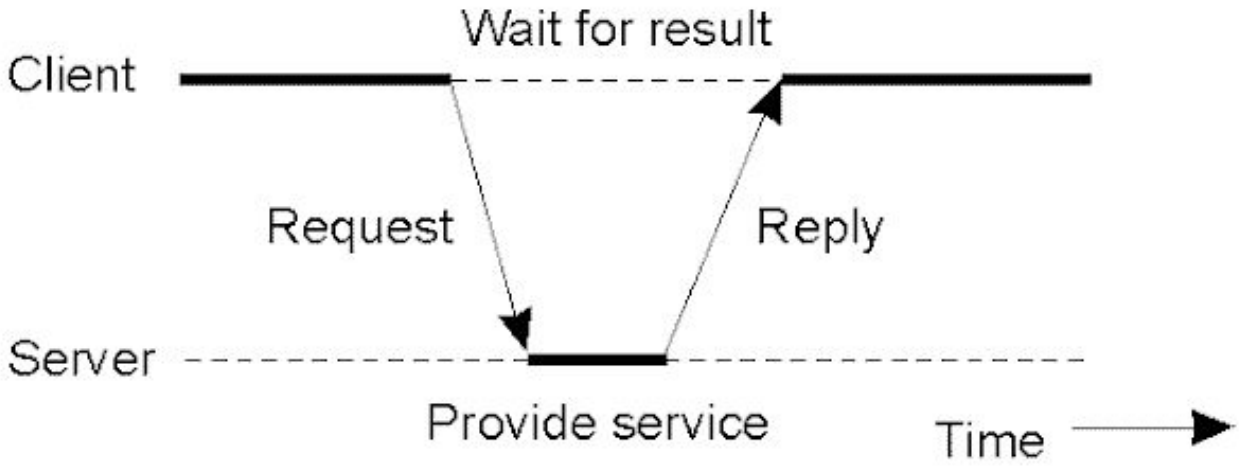
Simple client-server

Server invoked by another server

Multiple servers



Client-Server Interaction



Discussion

In order to make client-server model work,

- What information should the server know from the client?

- What information should the client know from the server?

Discussion

In order to make client-server model work:

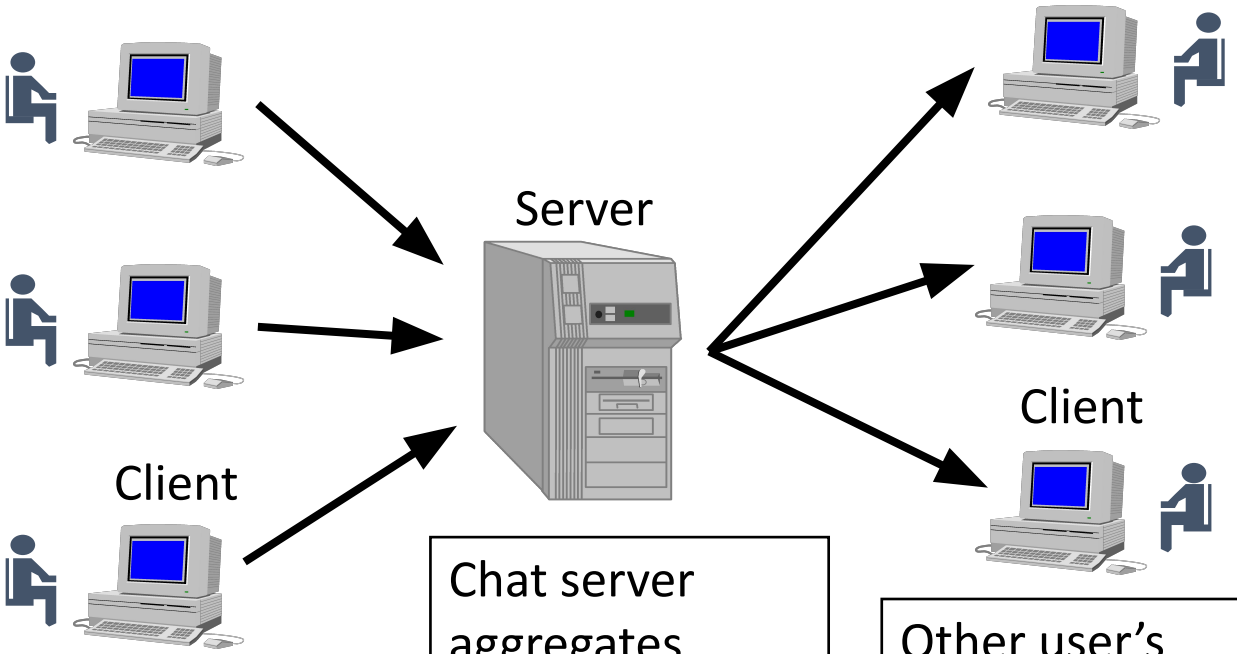
What information should the server know from the client?

- The server does not need to know the existence or address of the client prior to the connection.
- Just keep running for incoming connections from clients

What information should the client know from the server?

- The client needs to know the existence and the address of the server.

Chat application

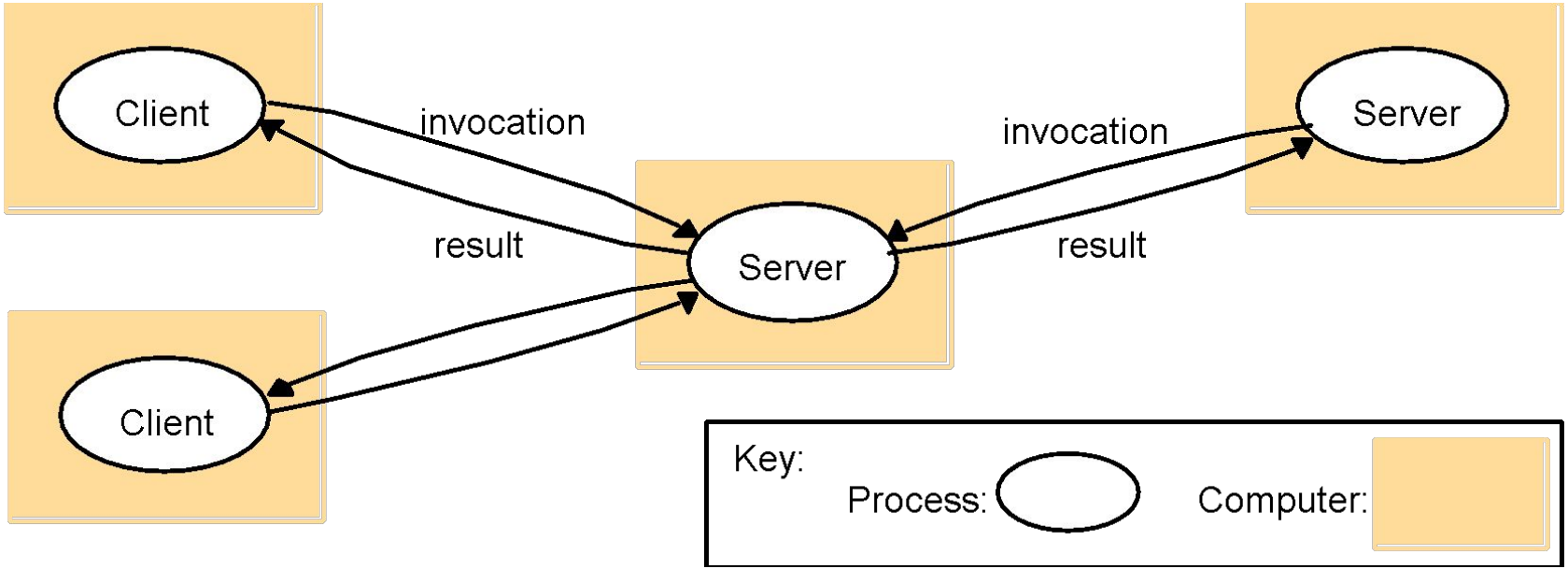


Chat clients send user's typing to server

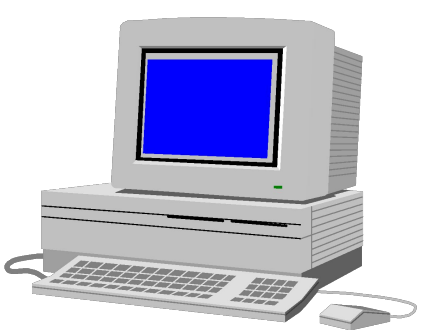
Chat server aggregates typing from all users and sends to all clients

Other user's clients display aggregated typing from chat server

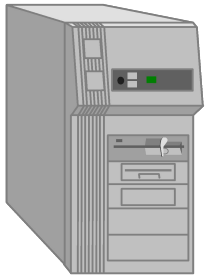
Client/Server (again)



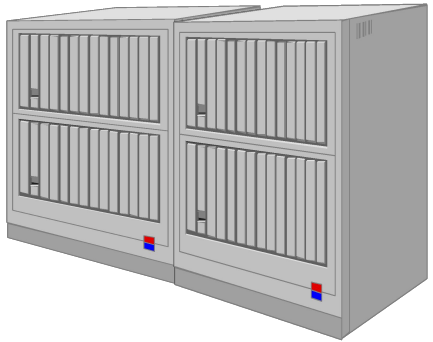
Three-tier client/server



Local-area network



Application server

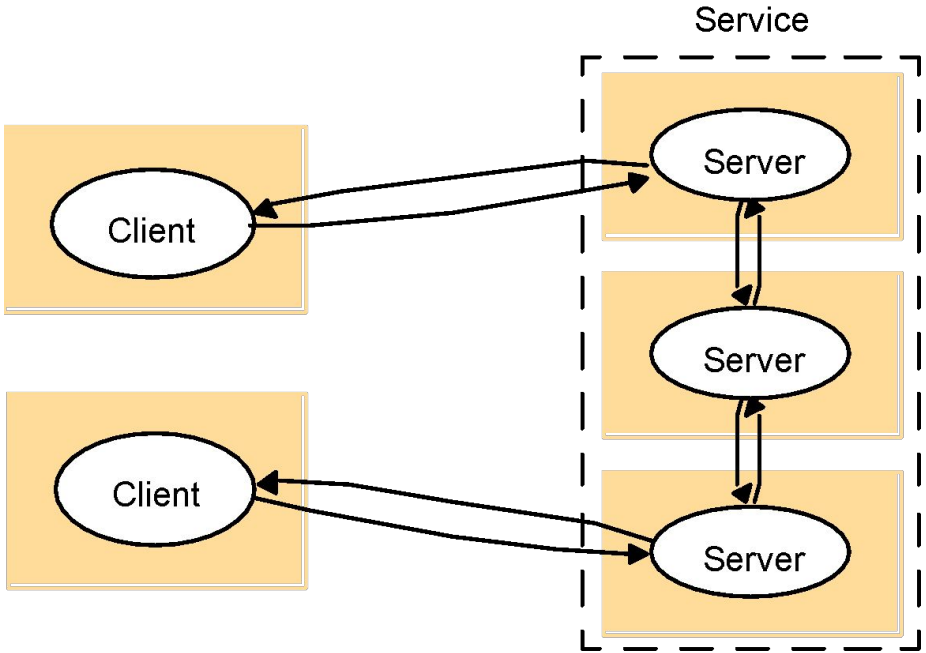


Data server

Client

Note: many clients per application server, several application servers per data server

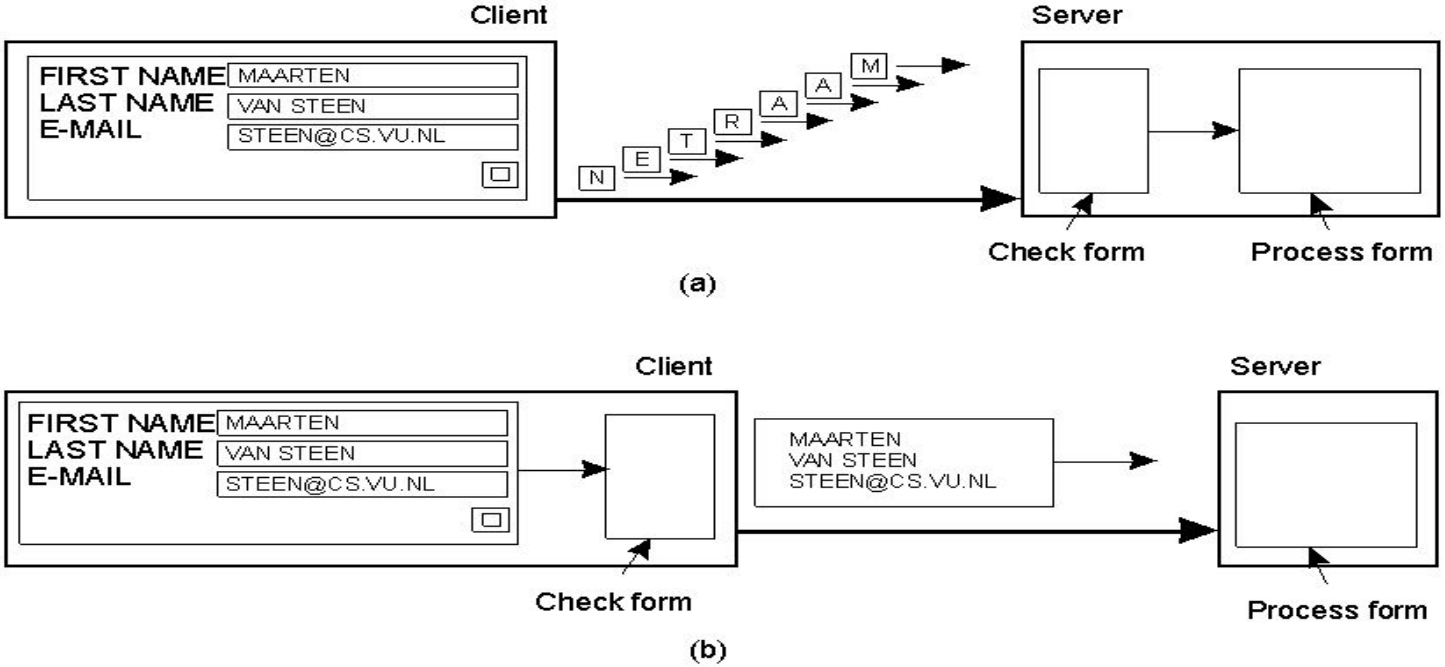
Client/Server (3)



A Service by Multiple Servers

Client/Server (4)

Could move computations to client ➜ but what is **wrong** with this?



Client/Server ➜ Limitations

- Scalability is hard to achieve
- Presents a single point of failure
- Requires administration
- Unused resources at the network edge (client side)
  - CPU cycles, storage, etc.
  - P2P systems try to address these limitations

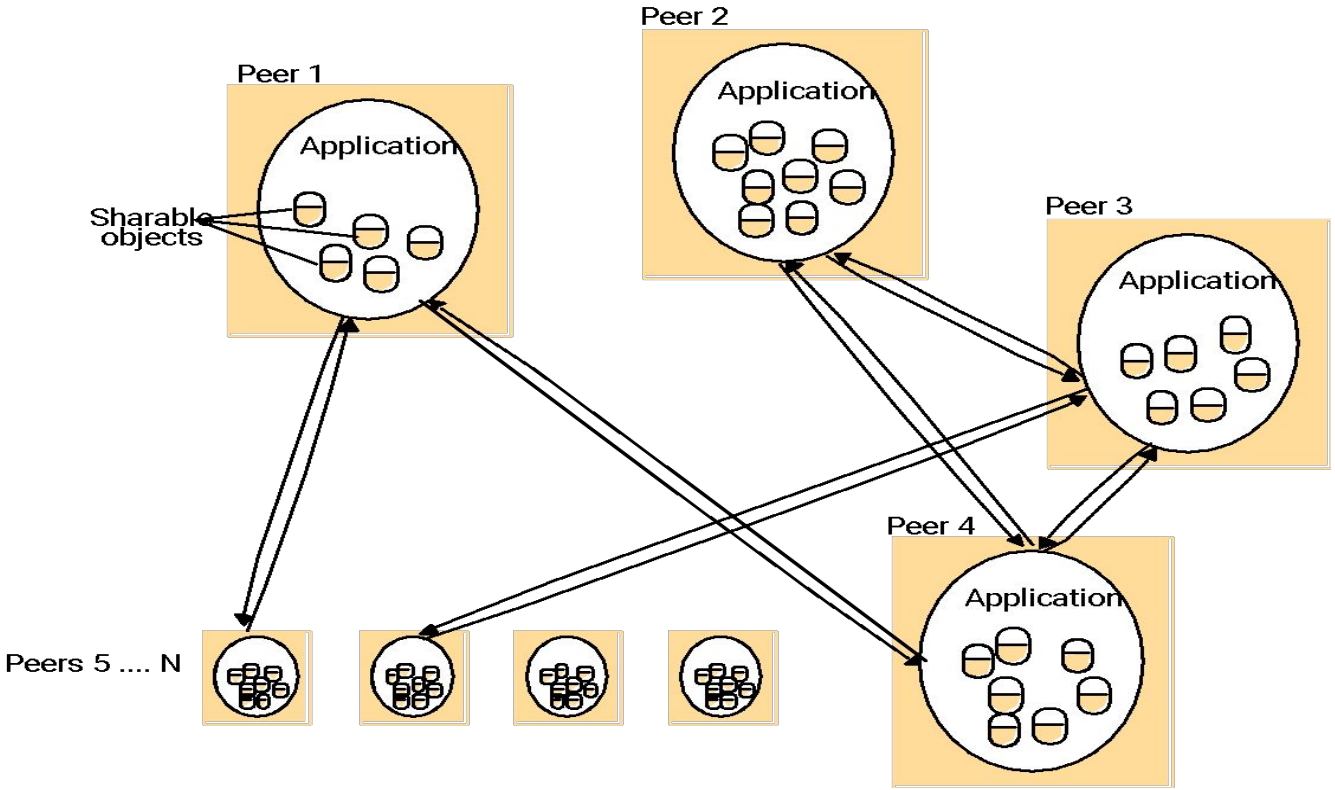
Peer-to-Peer (P2P)

What is it?

Use the **vast resources** of machines at the **edge of the Internet** to build a network that allows resource sharing with **limited central authority**

More than a system for sharing pirated music/movies!

P2P architecture



P2P - Characteristics

Exploit edge resources.

- Storage, content, CPU, human presence

Significant autonomy from any centralized authority

- Each node can act as a Client as well as a Server

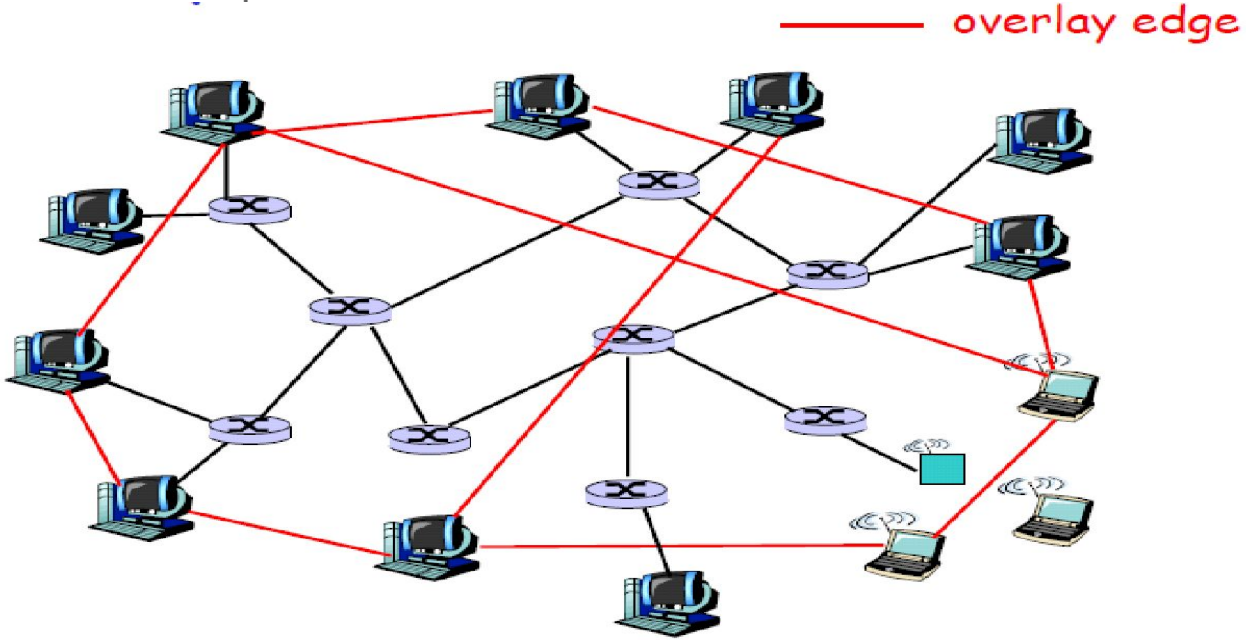
Resources at edge have intermittent connectivity

- Constantly being added & removed
- Infrastructure is untrusted and the components are unreliable

P2P - Overlay network

A P2P network is an **overlay network**

- Each link between peers consists of one or more IP links



P2P - Why talk about it?

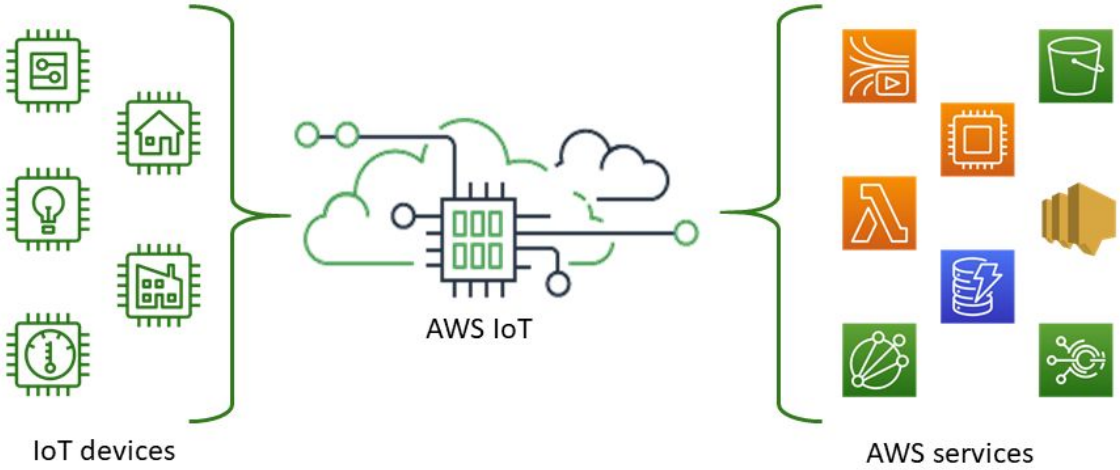
Can do IoT in the cloud!

Because of course there's a solution for it in the cloud providers

https://cloud.google.com/iot-core/

At least it is still active over on AWS...

https://docs.aws.amazon.com/iot/latest/developerguide/what-is-aws-iot.html



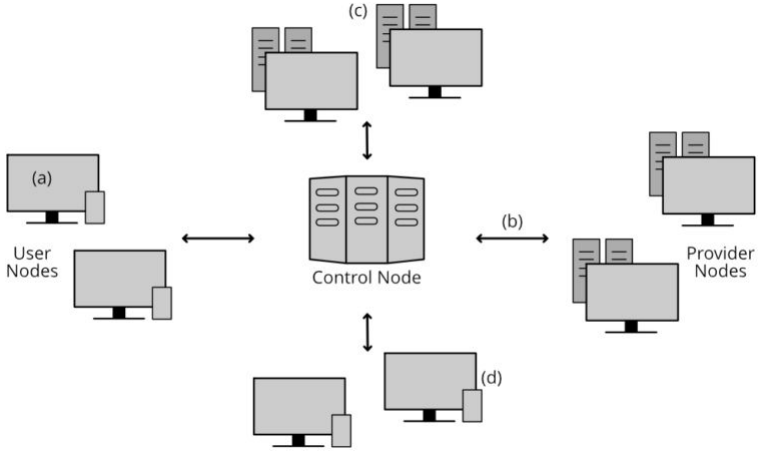
Grid computing

Networked heterogeneous resources

- Offload processing to devices on the grid
- Can be geographically distributed

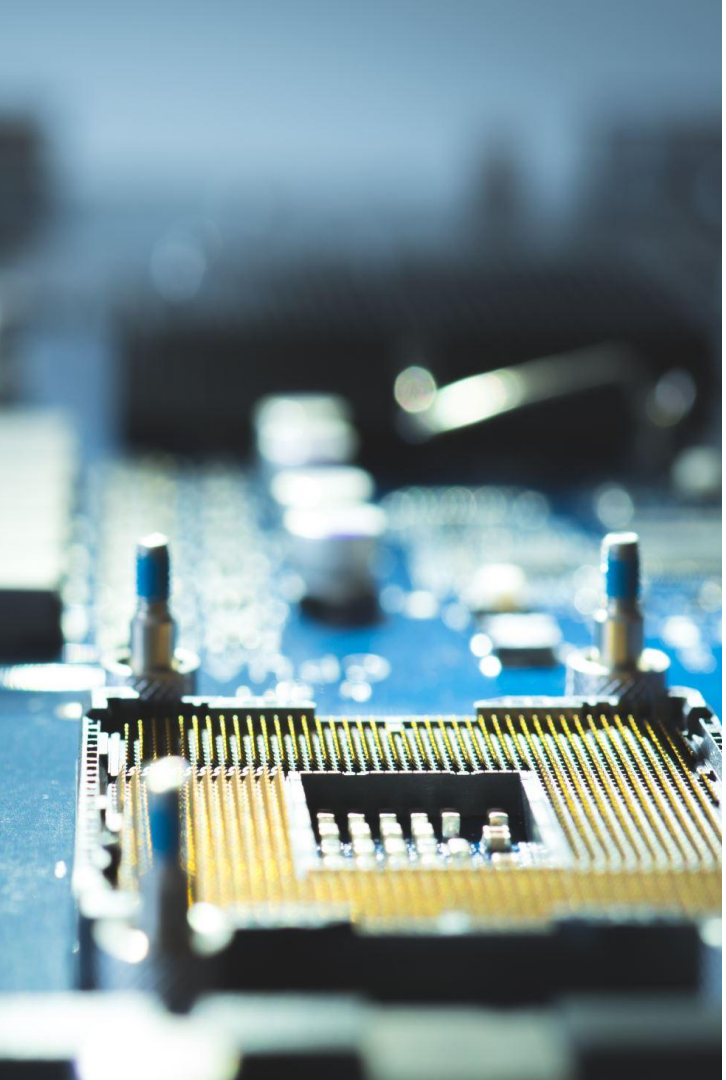
What could be an issue with different types of devices?

- What is a positive?



1. **Control Node**: Computer (Server / Group of Servers) which administers the whole network and keeps account of the resources in the network pool
2. **Provider Node**: Computer which provides its resources to the network resource pool
3. **User Node**: Computer which uses the resources of the network

https://blog.mi.hdm-stuttgart.de/index.php/2023/06/06/are-grids-dead/



Virtualization

Run multiple machines / operating systems on a single system

- Independently and concurrently
- Multiple users each log into their own VM

Why virtualization?

Consider an enterprise data center

- Always need more servers!
- Easier to virtualize workstations/servers
- Cost-effective
- Less hardware to deal with
- Easy to configure and deploy

Basics

Operating systems generally assume that they are in control

- HAHAHHAHAAHA
- Why though? What happens if two OSs try to use the same hardware?

Virtualization used to address this problem

- Virtualize hardware to support multiple OSs
- Abstraction of computing resources

Full virtualization currently accepted as the norm

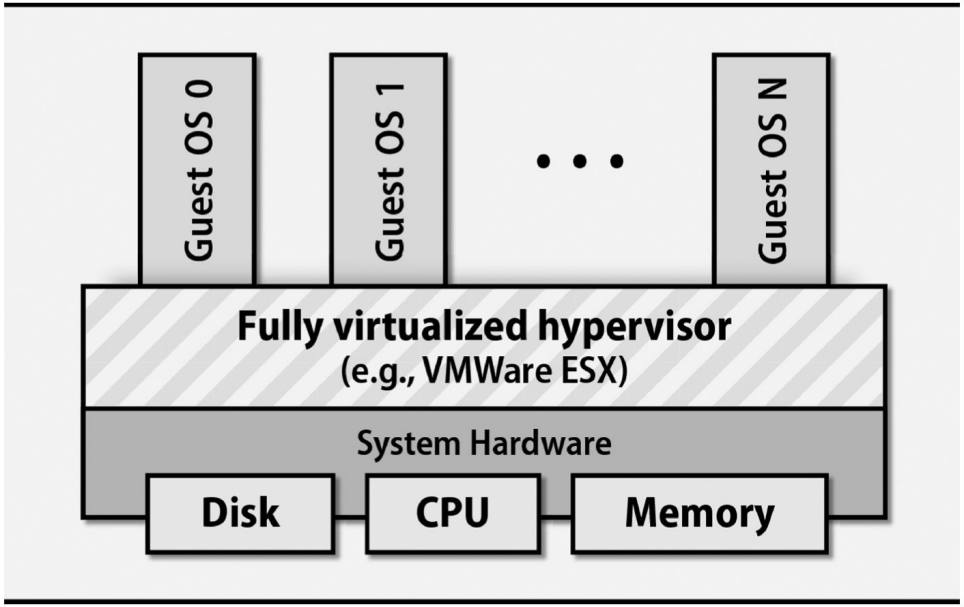
- OS is “unaware” that it is virtualized and thinks its hardware is real
- Hypervisor mediates between VM and real hardware
    - Hypervisor: virtual machine monitor

Full virtualization

Also called “bare-metal” hypervisors

Hypervisor controls physical hardware

- Provides layer of emulation
- OS thinks it’s making calls to real hardware
- Most secure approach
  - Any/all calls are intercepted by hypervisor
  - What happens in the VM stays in the VM

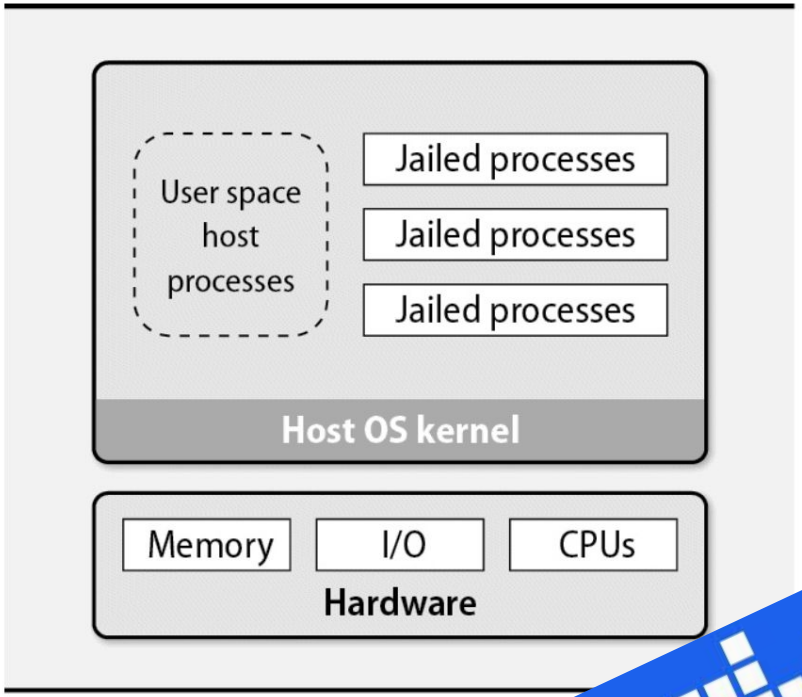


(Will leave other types to your OS/sysadmin classes)

But we'll talk a bit about this one ➜

OS-level virtualization (containerization)

- Rather than virtualizing OS, virtualize application environment instead!
  - Also referred to as containers/containerization
- No translation/virtualization layer
- Kernel isolates processes from rest of system
- Processes share kernel and other services of host OS, but cannot access files/resources outside of container



Key features

Isolation

- Programs run in their separate little space
- In theory, cannot bleed over to other VMs

Encapsulation

- State of machine localized to file on disk

Compatibility

- VM decoupled from bare metal
- Can run whatever OS you want!

And then, the cloud

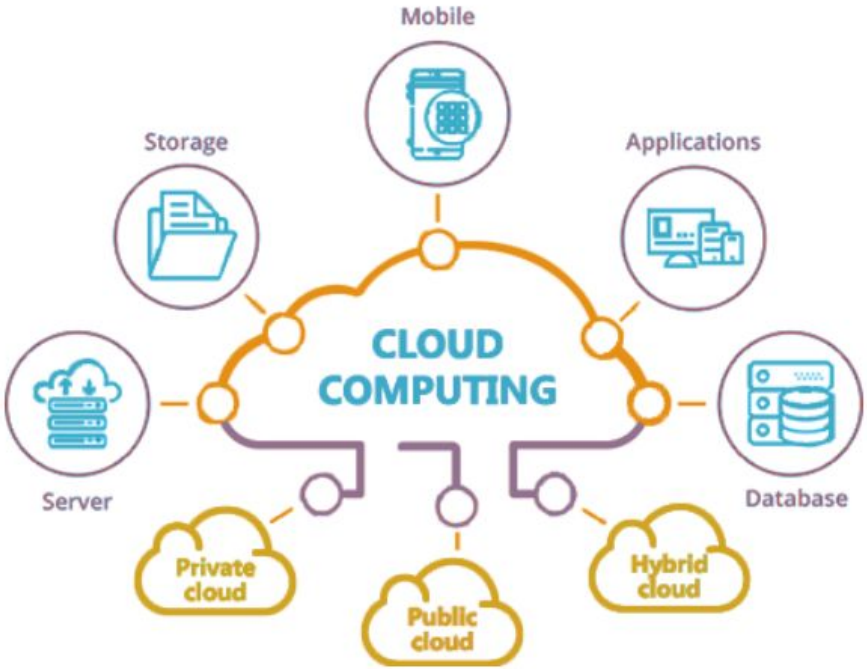
Just somebody else's server?

- Well yes, but also no

Somebody else's server,

- and data lake(s)...
- and geographically distributed...
- and networked

Abstraction on top of virtualization



What about high-performance computing (HPC)

If you are doing research, we have CLIPPER here...

https://services.gvsu.edu/TDClient/60/Portal/KB/ArticleDet?ID=19607

There's also iCER at MSU

Institute for Cyber-Enabled Research

Active Cores

22.0K

Last updated at 8:19

Running Jobs

3,419

Graph Shows Last 24 Hours (UTC)

OnDemand Portal

Currently Active Sessions	6
New Users Last 30 days	39

Last updated at 8:19

External Download

Globus	1 mb/s
rsync Gateway	1 mb/s

Last updated at 8:19

General Queue Times

Short	37m 22s
Long	5h 54m 17s
Long GPU	10h 6m 25s
Long Bigmem	1d 13h 38m

Average of recently completed jobs

Last updated at 8:19

Running Count

General	1425
Buy-in	1995

Last updated at 8:19

Node Utilization

Percent of nodes with active jobs

Last updated at 8:19

Storage Usage

Percent used of total avilable

Last updated at 8:19

Scavenger Jobs

586

Queue Count

General Short	4
General Long	659
General Long GPU	64
Buy-in	930

Last updated at 8:19

https://cloudscaling.com/blog/cloud-computing/grid-cloud-hpc-whats-the-diff/

Relationship to what we're talking about?

Well, we're at the end, yet this is more of a 'fun fact'

- Grid computing and HPC very similar, and I'm sure academics will argue about semantic differences
- Used interchangeably

HOWEVER

- Grid/HPC != cloud

Grid/HPC = performance

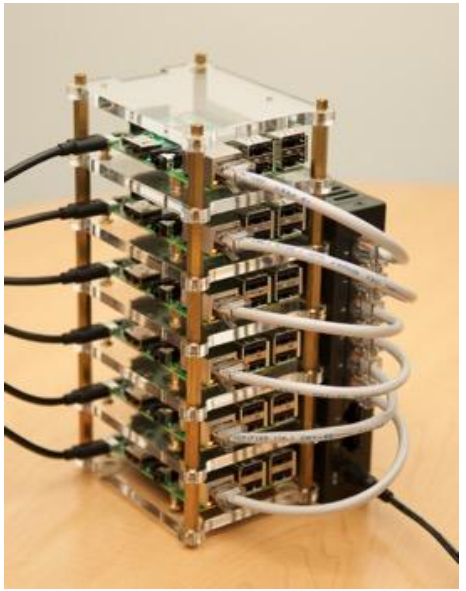
Cloud = scalability

Something HPC/Grid you could do at home?

(i.e., if you wanted to learn about managing a cluster)

(don't use this for real workloads - you'll get better performance off an old laptop)

(you could probably also install cloud software on something like this too)



Raspberry Pi Bramble

(or your favorite microcomputer networked...)

Could you do grid computing / HPC in the cloud?

How?

...is it tenable/worthwhile?

Assuming it still works (cough), we'll do a little demo of this when we hit apps

https://cloud.google.com/cluster-toolkit/docs/overview

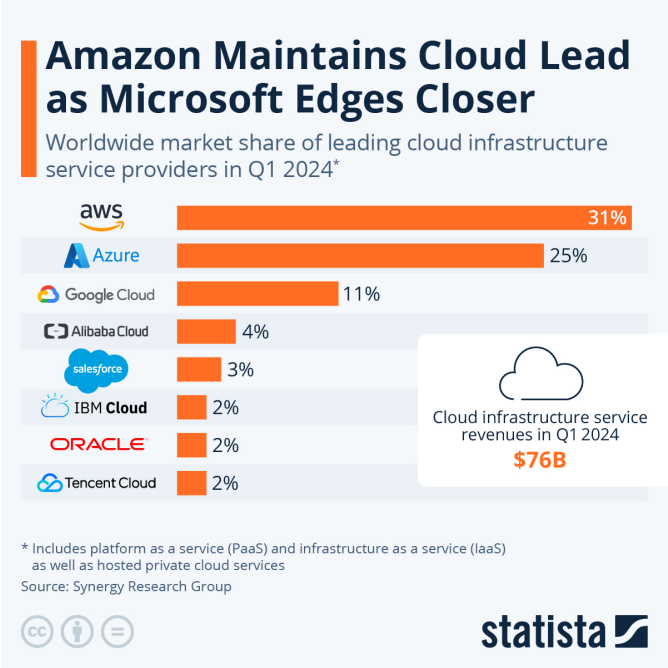
Also, $$$

Cloud history

Essentially, has become a fight between the big three

In terms of history:

- AWS was the first to market (~2006)
  - Internal/preview services available sooner
- Google Cloud and Microsoft not far behind
  - Google (App Engine, ~2008)
  - Microsoft (Windows Azure, ~2008)
- Each offer similar services and there really isn't one killer feature that a single provider offers



Assignment!

Look up 3 different cloud services in class

Your next homework will involve some investigative work:

For this, pick 3, jot down their links, and a quick blurb about what it is