CS 5412/LECTURE 1 TOPICS IN CLOUD COMPUTING

Ken Birman

Spring, 2022

THE CLOUD UNDERPINS MODERN COMPUTING

THE CLOUD

Physical: The cloud is a global deployment of massive data centers connected by ultra-fast networking, designed for scalability and robustness.

Logical: A collection of tools and platforms that scale amazingly well. The platforms matter most; as a developer, they allow you to extend/customize them to create your application as a “personality” over their capabilities.

Conceptual: A set of scalable ideas, concepts and technologies.

WHO INVENTED THE CLOUD?

Jeff Dean

Sanjay Ghemawat

Google's Jeff Dean and Sanjay Ghemawat get my "vote".

- Jeff Dean was a University of Washington PhD student. We know him well and he often visits Cornell. Now he is the head of Google Brain.
- Sanjay Ghemawat was a Cornell ugrad, then MIT PhD. He is a Senior Fellow in Google's systems infrastructure area.

Both Jeff and Sanjay are famous for simple and robust ways to scale things up (and writing about them). This was the key to the modern cloud.

I WAS THERE TOO…

- I didn't invent the cloud, but many of my students had huge roles.
- Personally, I created the "self-healing" software that ran the trading floors of the New York Stock Exchange and the Swiss Exchange for 10+ years. The US military uses this technology too.
- Designed the French portion of the European Air Traffic control system, control and created the core software. They've used it since 1996.
- Oracle and Microsoft both use a technology I invented to track the status of their clusters and data centers.
- Recently, I helped create the New England smart grid (for ISONE and NYPA), and helped the Air Force figure out how to leverage the cloud.

SOME OF MY PAST STUDENTS BECAME CLOUD COMPUTING SUPERSTARS

Werner Vogels was in my group until 2005.
He has been CTO of Amazon since 2006.

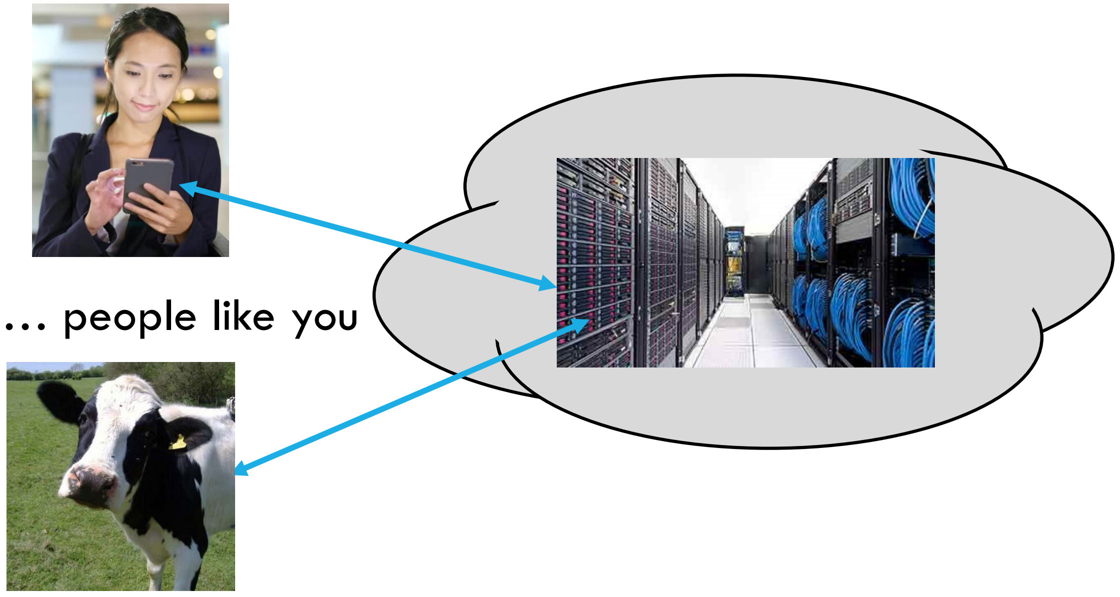
Ranveer Chandra is Chief Scientist for Azure Global, head of Networking Research and responsible for their FarmBeats product.

Yee-Jiun Song: VP Engineering, Facebook

Qi Huang: Facebook video/photo delivery

Dalia Malkhi: CTO for Diem, Facebook's digital currency.

CLOUD COMPUTING

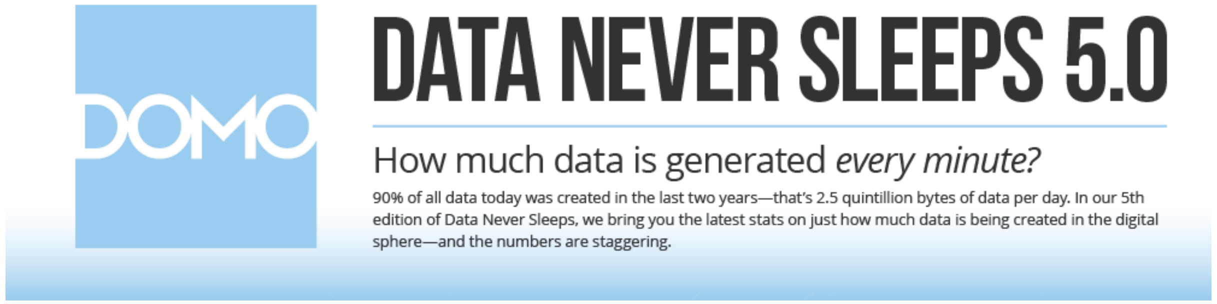


CS5412 is…

- A deep study of a big topic.
- In spring 2021 one focus will be on "smart farming" in Azure IoT Edge.
- The farming focus leverages a Cornell and Microsoft interest (and an Azure product area) and makes it real.

Fog computing!

EVERYTHING IS BIG IN THE CLOUD

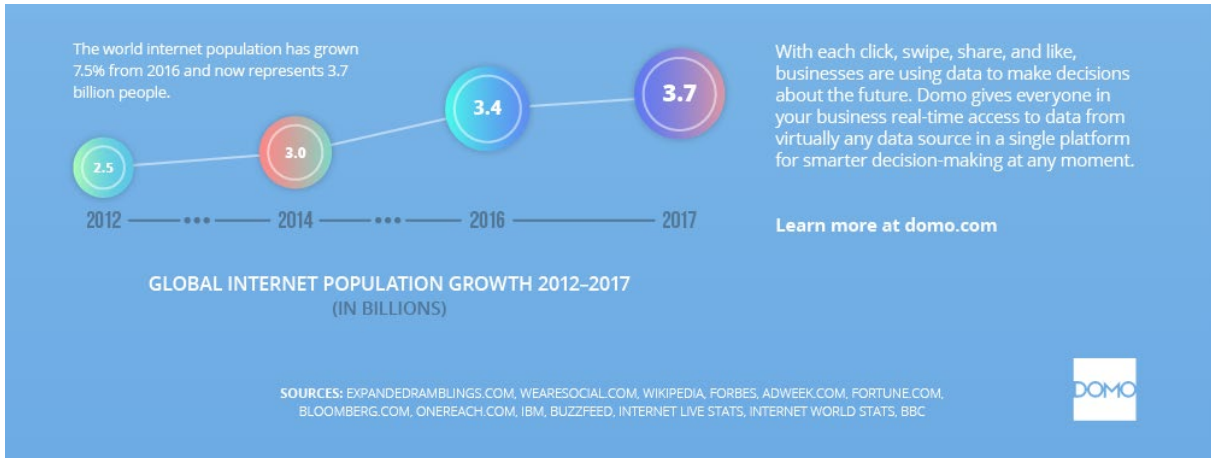


DOMO

DATA NEVER SLEEPS 5.0

How much data is generated *every minute?*

90% of all data today was created in the last two years—that's 2.5 quintillion bytes of data per day. In our 5th edition of Data Never Sleeps, we bring you the latest stats on just how much data is being created in the digital sphere—and the numbers are staggering.



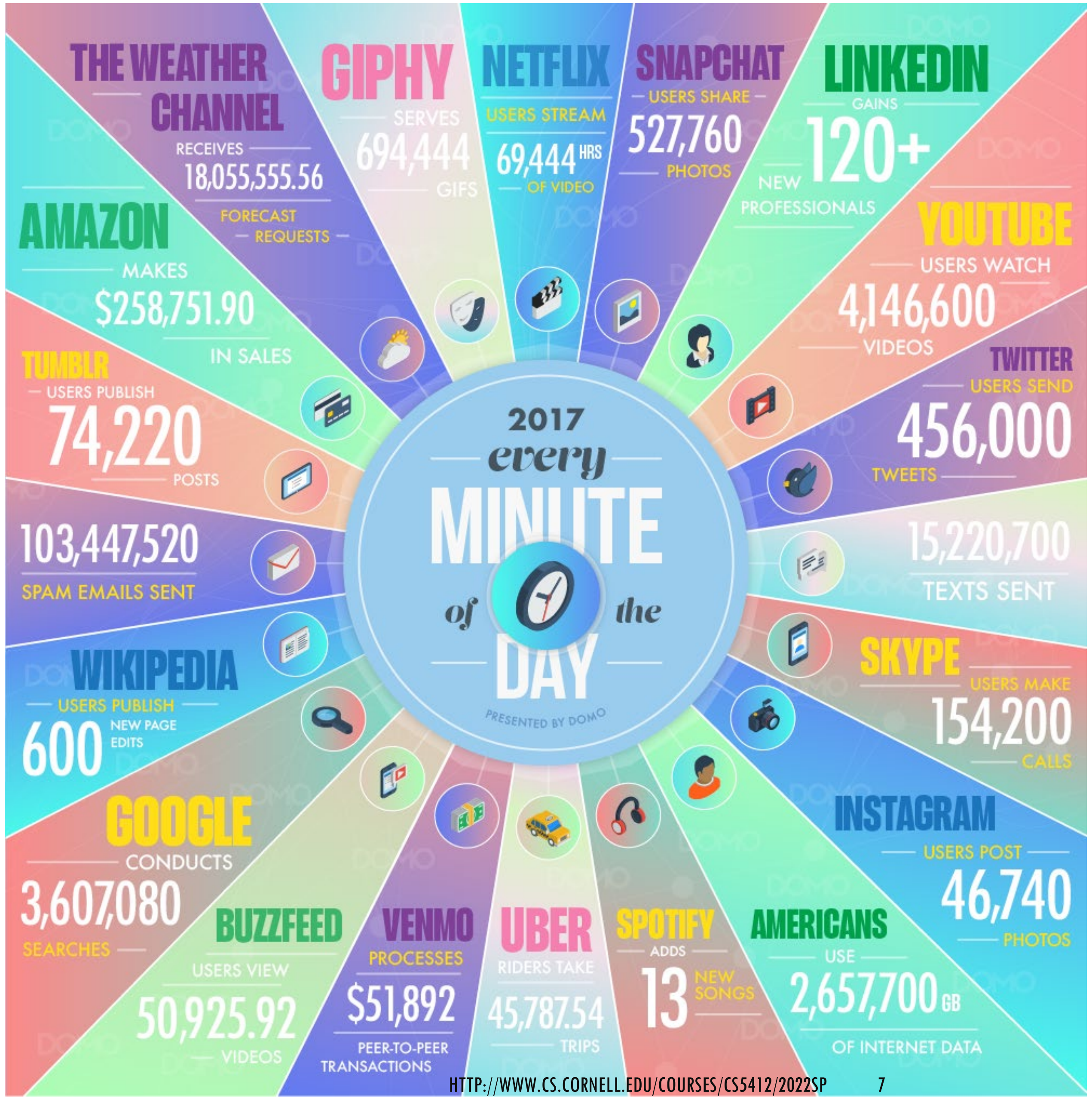
The world internet population has grown 7.5% from 2016 and now represents 3.7 billion people.

GLOBAL INTERNET POPULATION GROWTH 2012–2017
(IN BILLIONS)

With each click, swipe, share, and like, businesses are using data to make decisions about the future. Domo gives everyone in your business real-time access to data from virtually any data source in a single platform for smarter decision-making at any moment.

Learn more at domo.com

SOURCES: EXPANDEDRAMBLINGS.COM, WEARESOCIAL.COM, WIKIPEDIA, FORBES, ADWEEK.COM, FORTUNE.COM, BLOOMBERG.COM, ONEREACH.COM, IBM, BUZZFEED, INTERNET LIVE STATS, INTERNET WORLD STATS, BBC

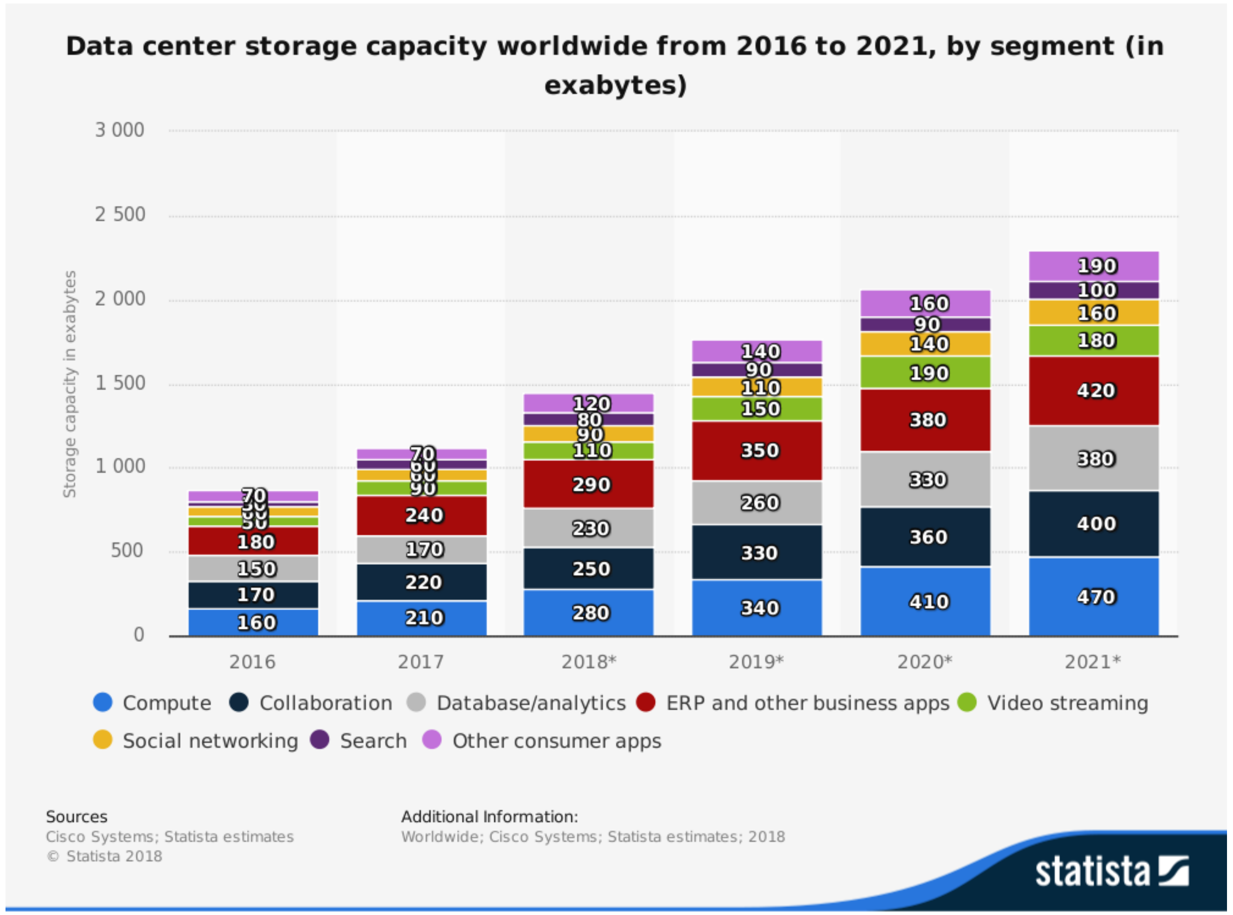


DATA IN THE CLOUD

1 Exabyte of data is 1,073,741,824 GB. (Your hard disk probably holds 64GB, but is way too slow by data-center standards)

The Internet has about 2B websites, and of these, 644M have "active content"

… and all of this is "pre Internet of Things"



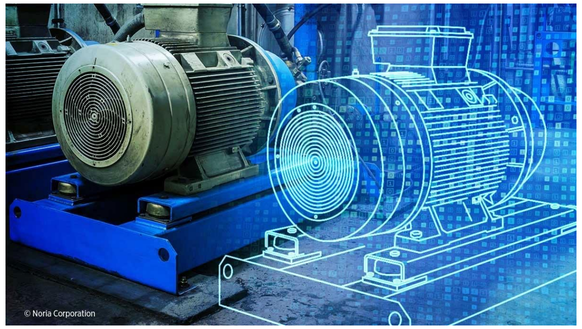
IoT AT THE EDGE: THE BIG NEW THING

Our course looks at many aspects of the cloud but will focus on *live interactions with the outside world.*

This used to be dominated by web interfaces, but increasingly also includes 5G mobility and a wide variety of **Internet of Things sensors and actuators**: the so-called **IoT cloud.**

IoT is a huge opportunity and growth area and we will spend a lot of time looking at how these cloud options work and are used.

CONCEPT: DIGITAL TWIN



We often like to think of the cloud as having a software “image” that matches with IoT in the physical world.

The IoT devices are physical and live “outside” the cloud, but for each device we have a “digital twin” that lives inside the cloud, and is a kind of proxy. Actions on the twin are translated to reading the sensor or telling the actuator to do something.

This model will be very useful to us in CS5412

HOW DID TODAY'S CLOUD EVOLVE?

Prior to ~2005, we had "data centers designed for high availability".

Amazon had especially large ones, to serve its web requests

- This is all before the AWS cloud model
- The real goal was just to support online shopping

Their system wasn't very reliable, and the core problem was scaling

- Like a theoretical complexity growth issue.
- Amazon's computers were overloaded and often crashed

WASN’T GOOGLE FIRST?

Google was still building their first scalable infrastructure in this period.

Because Amazon ran into scaling issues first, Google (a bit later) managed to avoid them.

- In some sense, Amazon dealt with these issues “in real time”.
- Google had a chance to build a second system by learning from Amazon’s mistakes and approaches.

YAHOO EXPERIMENT

A sprint to render your web page!

In the 2005 time period everyone was talking about an experiment done at Yahoo. It was an "alpha/beta" experiment about ad-click-through

- Customers who saw web page rendering faster than 100ms clicked ads.
- For every 100ms delay, *click-through rates noticeably dropped*.

Speed at scale determines revenue, and revenue shapes technology: an arms race to speed up the cloud.

MORE YAHOO FINDINGS

A sprint to render your web page!

Rending the ads first didn't help– in fact it hurt.

Customers wanted to see the "real content" first.

Rendering the ads after the content hurt too.

To get the best click-through rates, render your pages (ads included) fast!

EVERYONE HEARD THIS MESSAGE

At Amazon, Jeff Bezos spread the word internally.

He wanted Amazon to win this sprint.

The whole company was told to focus on ensuring that every Amazon product page would render with minimal delay. Unfortunately… as more and more customers turned up… Amazon's web pages ***slowed down.*** This is a "crisis of the commons" situation.

THE CRISIS OF THE COMMONS

At the center of the village is a lovely grassy commons. Everyone uses it.

One day a farmer has an awesome idea. He lets his goats graze on the commons. This saves the time of herding them to his fields. This gains him hours that he uses to improve his goat cheese factory.

He earns extra money with his award-winning cheeses.

THE CRISIS OF THE COMMONS

… his neighbors love the idea! All of them decide to use the commons.

In no time all the grass is gone and the commons is reduced to dust.

For Amazon, success was like that. The first shoppers loved the site, but then "everyone" wanted to use it, and it overloaded and collapsed.

THE CLOUD AND THE “THUNDERING HERD”

In fact this is a very common pattern.

Something becomes successful and within weeks, everyone wants to try it.

In the cloud you can make this work!
The cloud is about spin-on-a-dime scaleup stories.

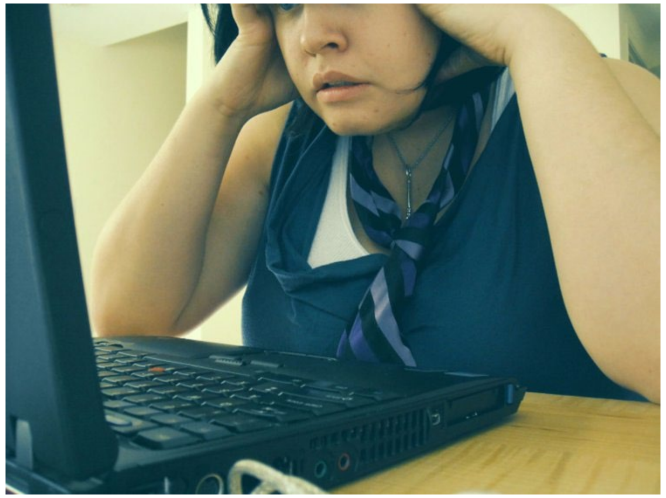
STARTING AROUND 2006, AMAZON LED IN REINVENTING DATA CENTER COMPUTING

Amazon reorganized their whole approach:

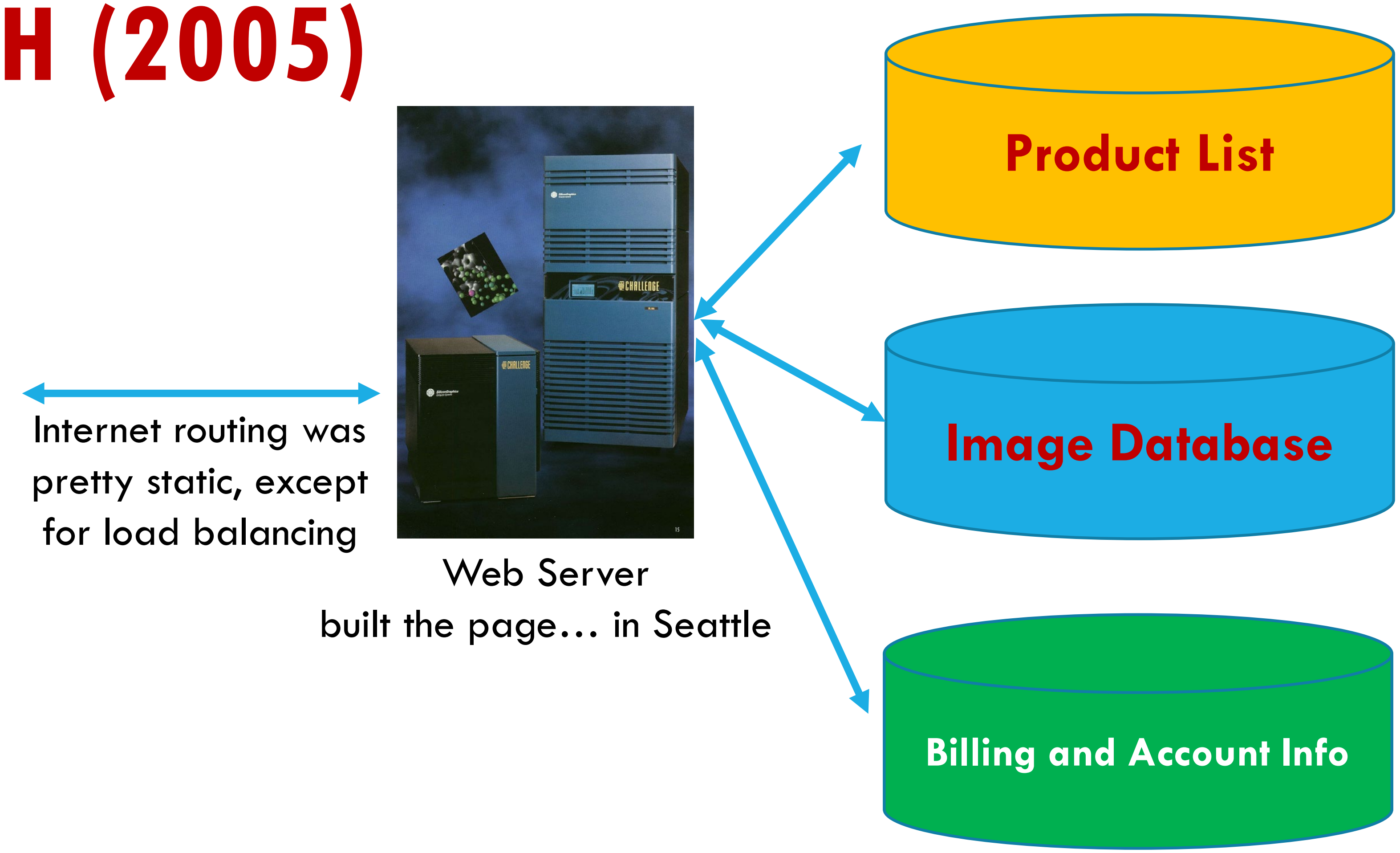
- Requests arrived at a “first tier” of very lightweight servers.
- These dispatched work requests on a message bus or queue.
- The requests were selected by “micro-services” running in elastic pools.
- One web request might involve tens or hundreds of μ-services!

They also began to guess at your next action and precompute what they would probably need to answer your next query or link click.

OLD APPROACH (2005)



Computers were mostly desktops

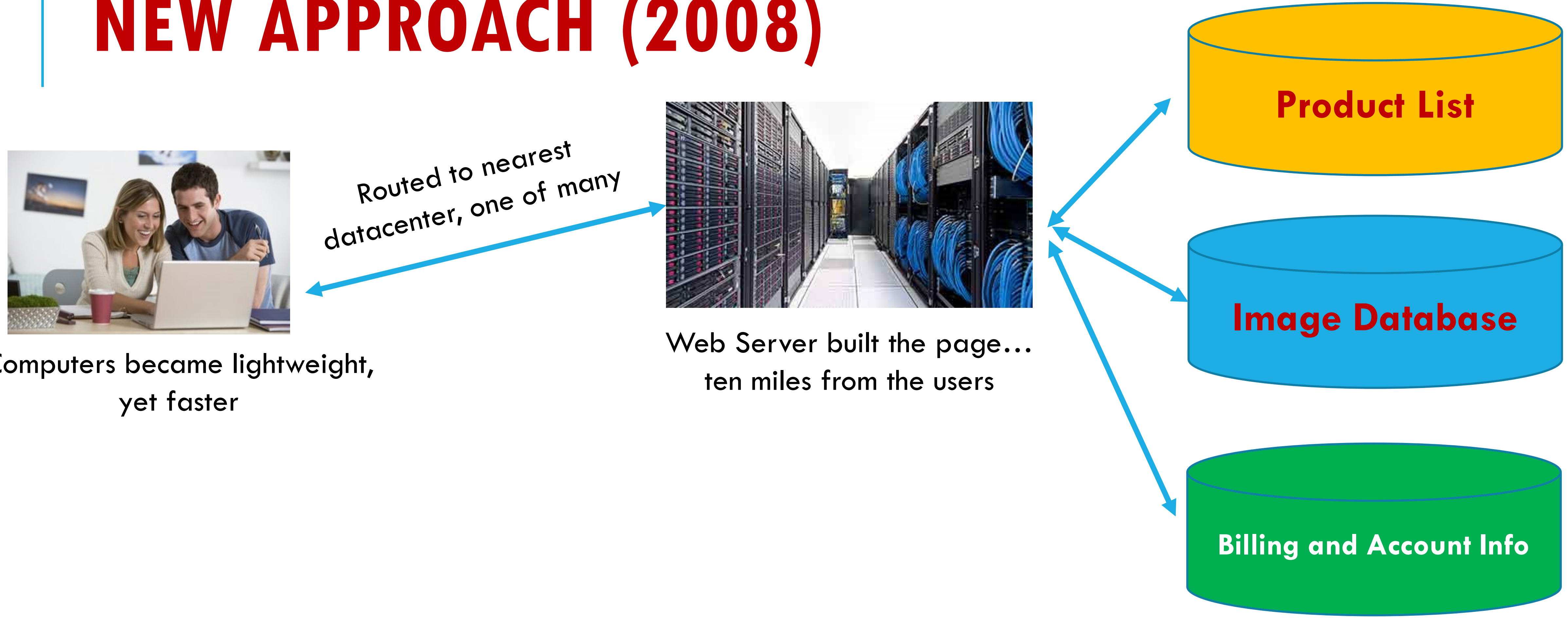


Internet routing was pretty static, except for load balancing

Web Server
built the page… in Seattle

Databases held the real product inventory

NEW APPROACH (2008)

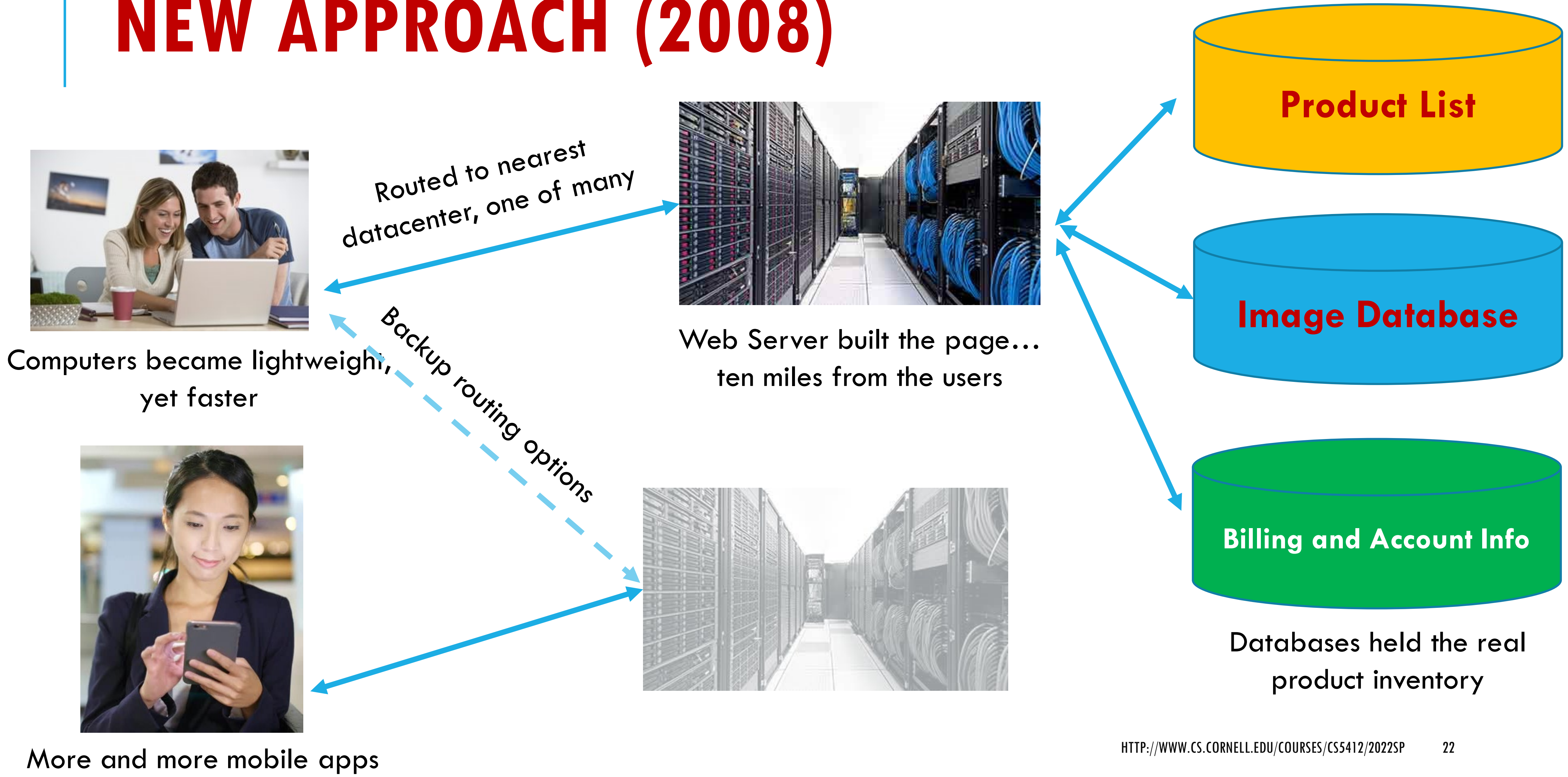


Computers became lightweight, yet faster

Web Server built the page… ten miles from the users

Databases held the real product inventory

NEW APPROACH (2008)

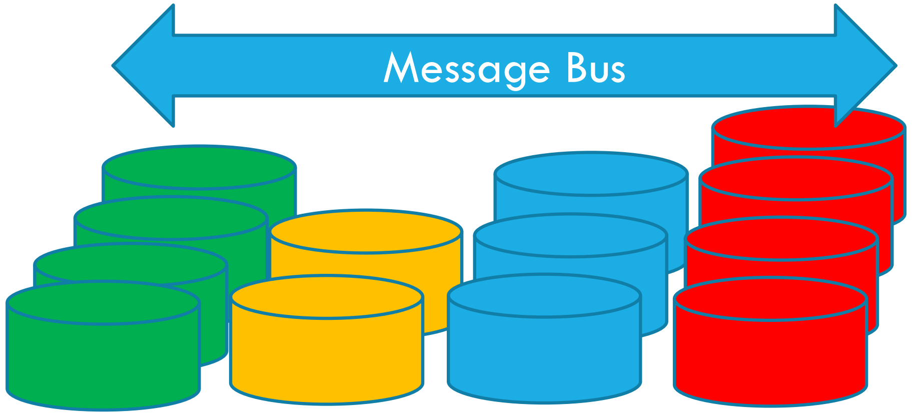

Routed to nearest datacenter, one of many
Backup routing options
Computers became lightweight, yet faster
More and more mobile apps
Web Server built the page… ten miles from the users
Product List
Image Database
Billing and Account Info
Databases held the real product inventory

NEW APPROACH (2008)

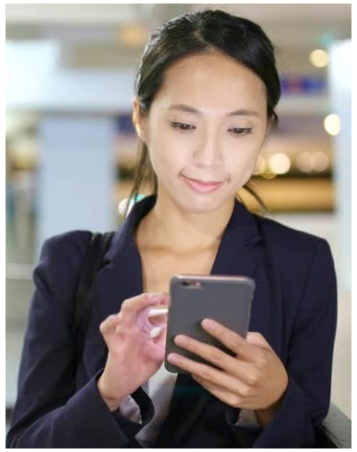
Desktops with snappier response

Routed to nearest datacenter, one of many

Web Server becomes simpler and does less of the real work

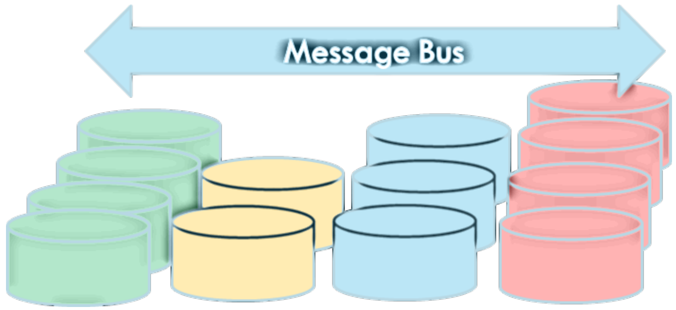


Racks of highly parallel workers do much of the data fetching and processing, ideally ahead of need… The old databases are split into smaller and highly parallel services.



More and more mobile apps

GeoReplication



TIER ONE / TIER TWO

We often talk about the cloud as a “multi-tier” environment.

Tier one: programs that generate the web page you see.

Tier two: services that support tier one. We will see one later (DHT/KVS storage used to create a massive cache)

TODAY’S CLOUD

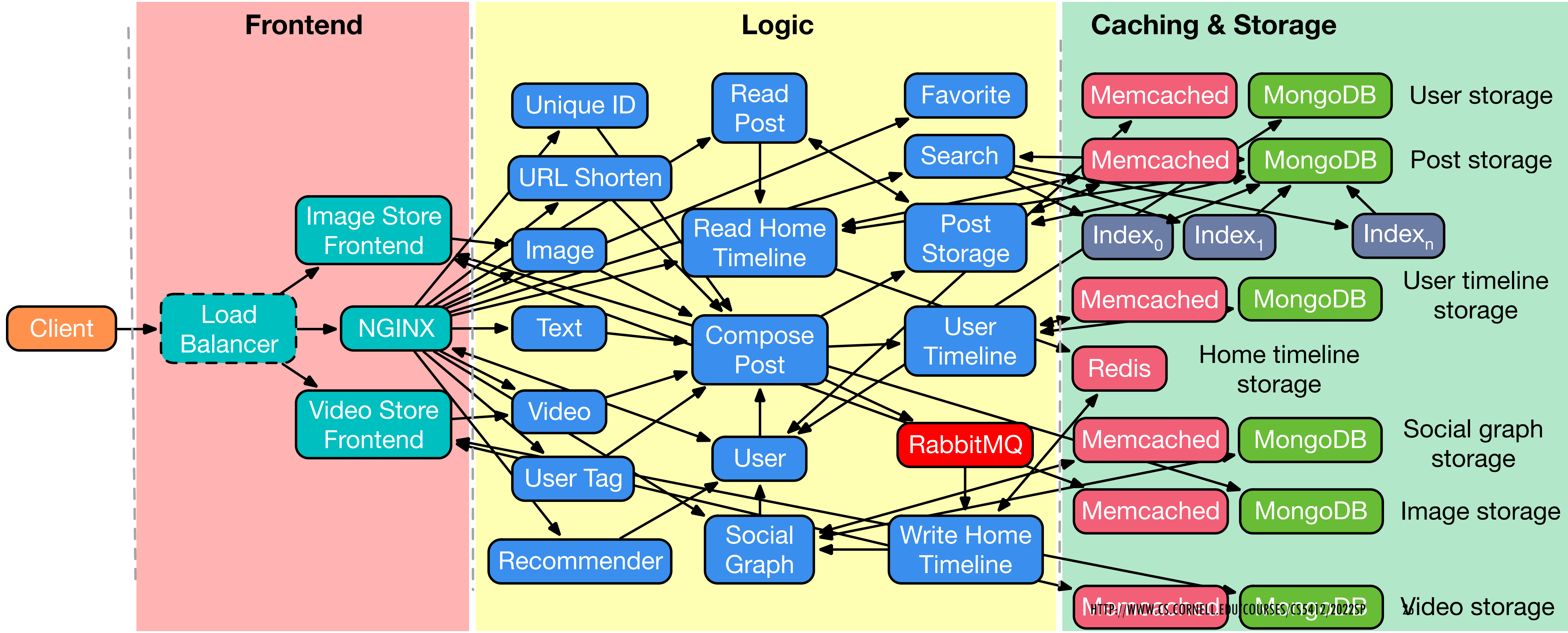
Tier one runs on very lightweight servers:

- They use very small amounts of computer memory
- They don’t need a lot of compute power either
- They have limited needs for storage, or network I/O

Tier two μ-Services specialize in various aspects of the content delivered to the end-user. They may run on somewhat “beefier” computers.

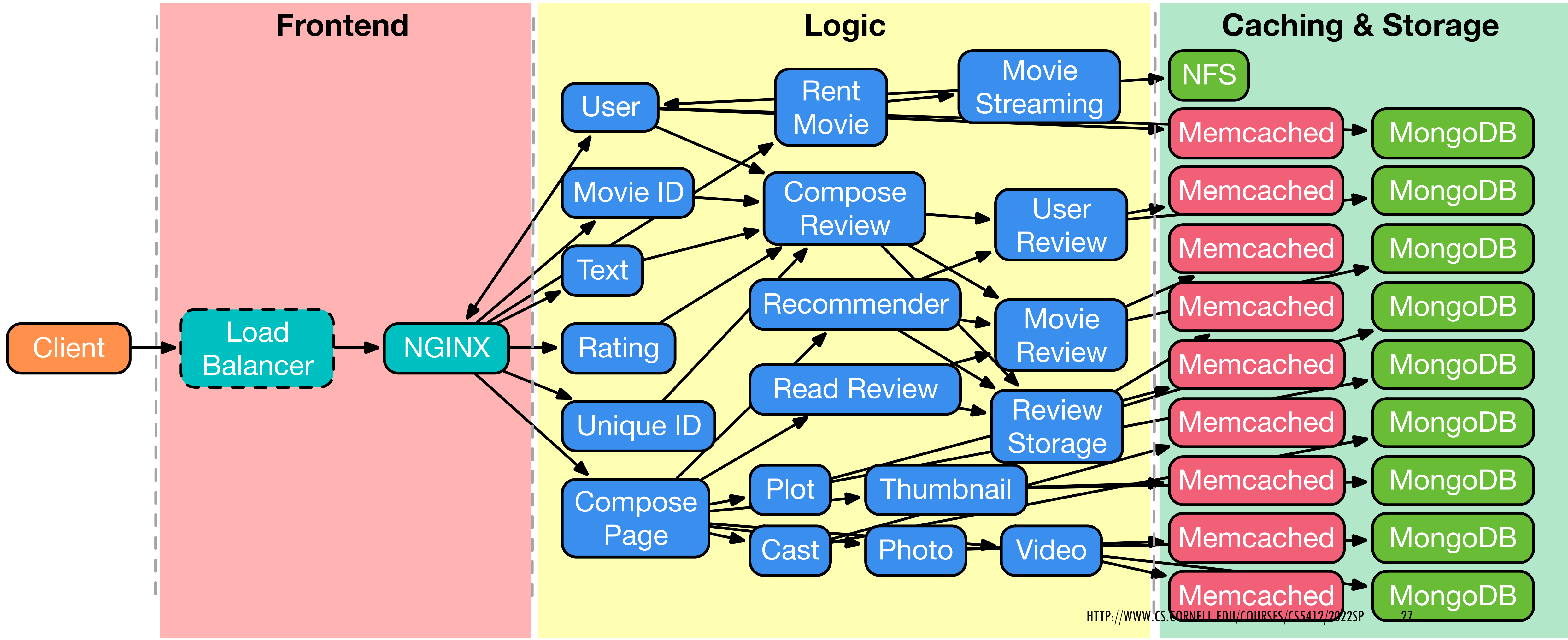
End-to-end Microservices (from Christina Delimitrou)

Social Network



End-to-end Microservices (from Christina Delimitrou)

Media Service



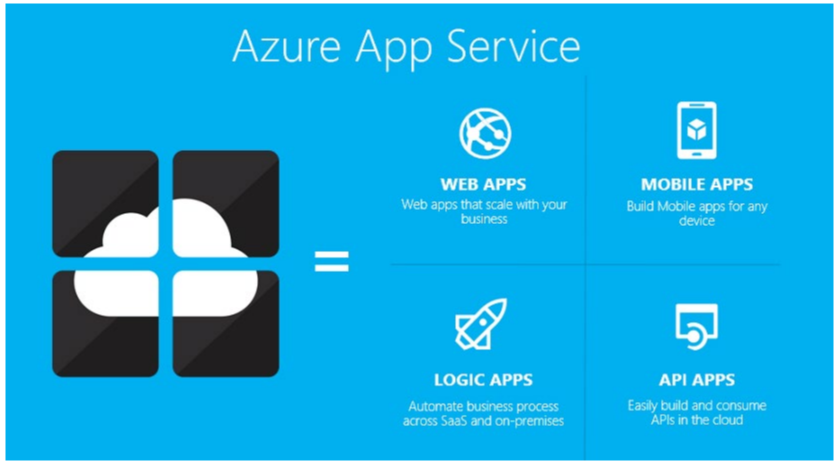
EACH MICROSERVICE IS A PARALLEL “POOL”!

Every one of those little nodes is itself a small elastic pool of processes

A microservice (μ-service) is a kind of program designed so that the data center can run one instance… or many instances, “elastically”, to deal with dynamically varying demand.

The idea here is that any instance can handle any request equally well, so there is no need for very careful “routing” of specific requests to specific instances. This lets the data center adapt to changing loads easily!

THESE POOLS ARE MANAGED AUTOMATICALLY

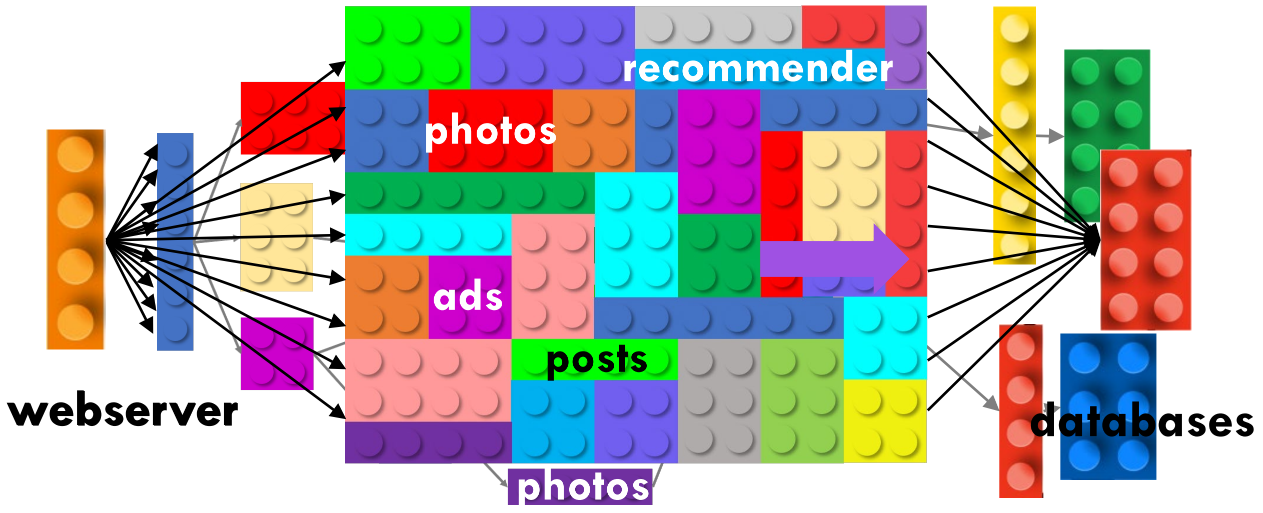

Azure App Service
WEB APPS
Web apps that scale with your business
MOBILE APPS
Build Mobile apps for any device
LOGIC APPS
Automate business process across SaaS and on-premises
API APPS
Easily build and consume APIs in the cloud

In Azure, for example, there is a tool called the “App Service” (we’ll use it!)

The App Service manages a big collection of compute resources in the cloud. Developers can install your own services in it (as “containers”). Configuration files tell it when to launch them for you, automatically.

Among the features is a way for it to watch the queue of requests and automatically add instances or shut instances down to match loads.

Motivation for μ-services (Delimitrou)



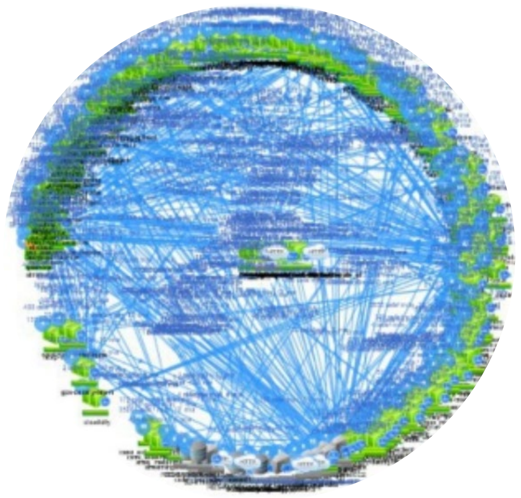
Monolith

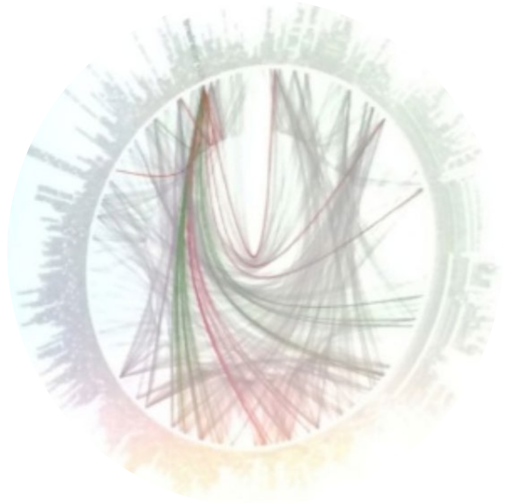
Microservices

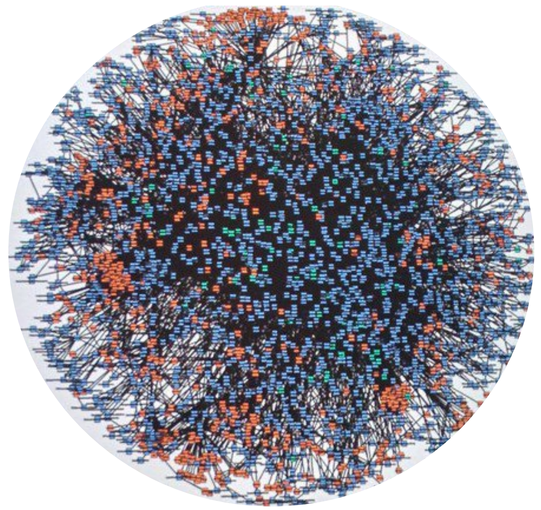
Advantages of μ-services:

- Modular → easier to understand
- Speed of development & deployment
- On-demand provisioning, elasticity
- Language/framework heterogeneity

Performance management (Delimitrou)


Netflix


Twitter

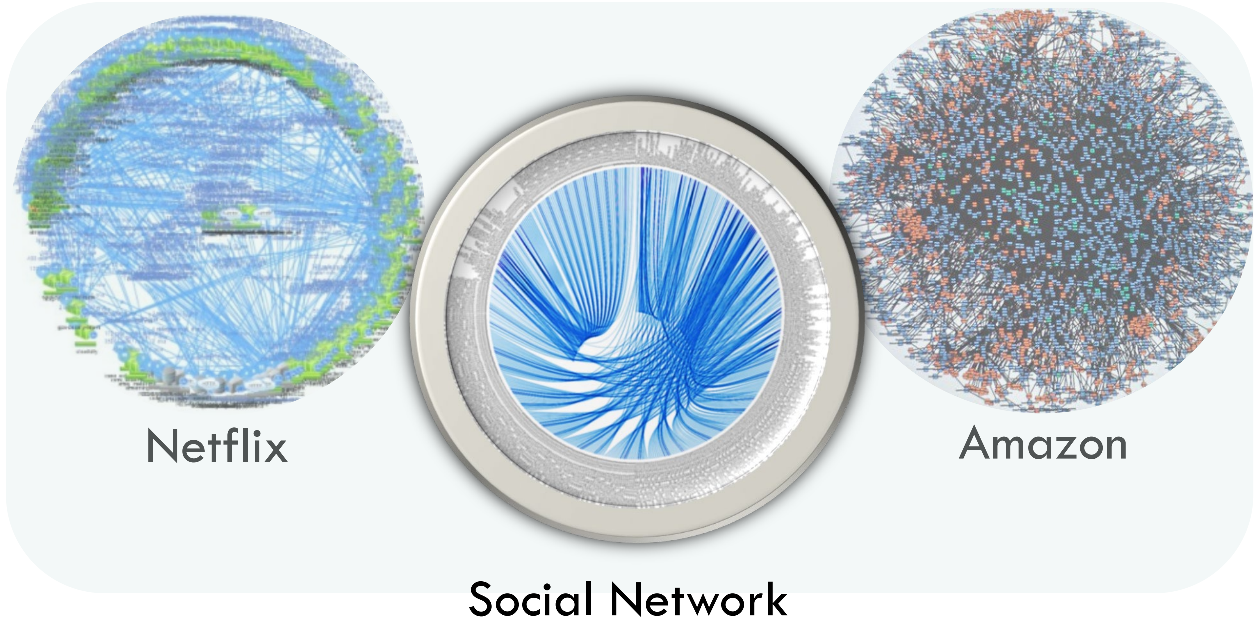

Amazon

Brings many benefits… but complicates cluster management & performance debugging

Dependencies cause cascading QoS violations

Difficult to isolate root cause of performance unpredictability

Performance visualization



Dependencies cause cascading QoS violations

Empirical performance debugging → too slow, bottlenecks propagate

Long recovery times for performance

WHAT DID WE JUST SEE?

The cloud scheduler watched each μ-service pool (each is shown as one dot, with color telling us how long the task queue was, and the purple circle showing how CPU loaded it is).

The picture didn't show how many instances were active – that makes it too hard to render. But each pool had varying numbers of instances. The App Server was automatically creating and removing instances.

Each time the scheduler realized that it should add instances to a slow service, some of the "deadline violations" went away.

WHAT DOES IT MEAN TO “ADD INSTANCES”?

For some applications (ones with NUMA threading for parallelism) we add instances by launching new threads on additional cores.

For others, we literally run two or more identical copies of the same program, on different computers! They use a “load balancer” to send requests to the least loaded instances.

And you can even combine these models…

WHY POOLS OF INSTANCES?

This is really just one of a few ways to get parallelism

Let's look at some of the choices and try to understand why the cloud favors the approach we just saw on the Delimitrou visualization.

SCALABILITY ISSUES ARISE EVEN INSIDE A SINGLE μ-SERVICE INSTANCE

We've been acting as if each μ-service is a set of "processes" but ignoring how those processes were built.

In fact they will use parallel programming of some form because modern computers have NUMA architectures.

How do cloud developers think about this form of parallelism?

DEEP DIVE: BEST WAY TO LEVERAGE PARALLELISM

Not every way of scaling is equally effective. Pick poorly and you might make less money!

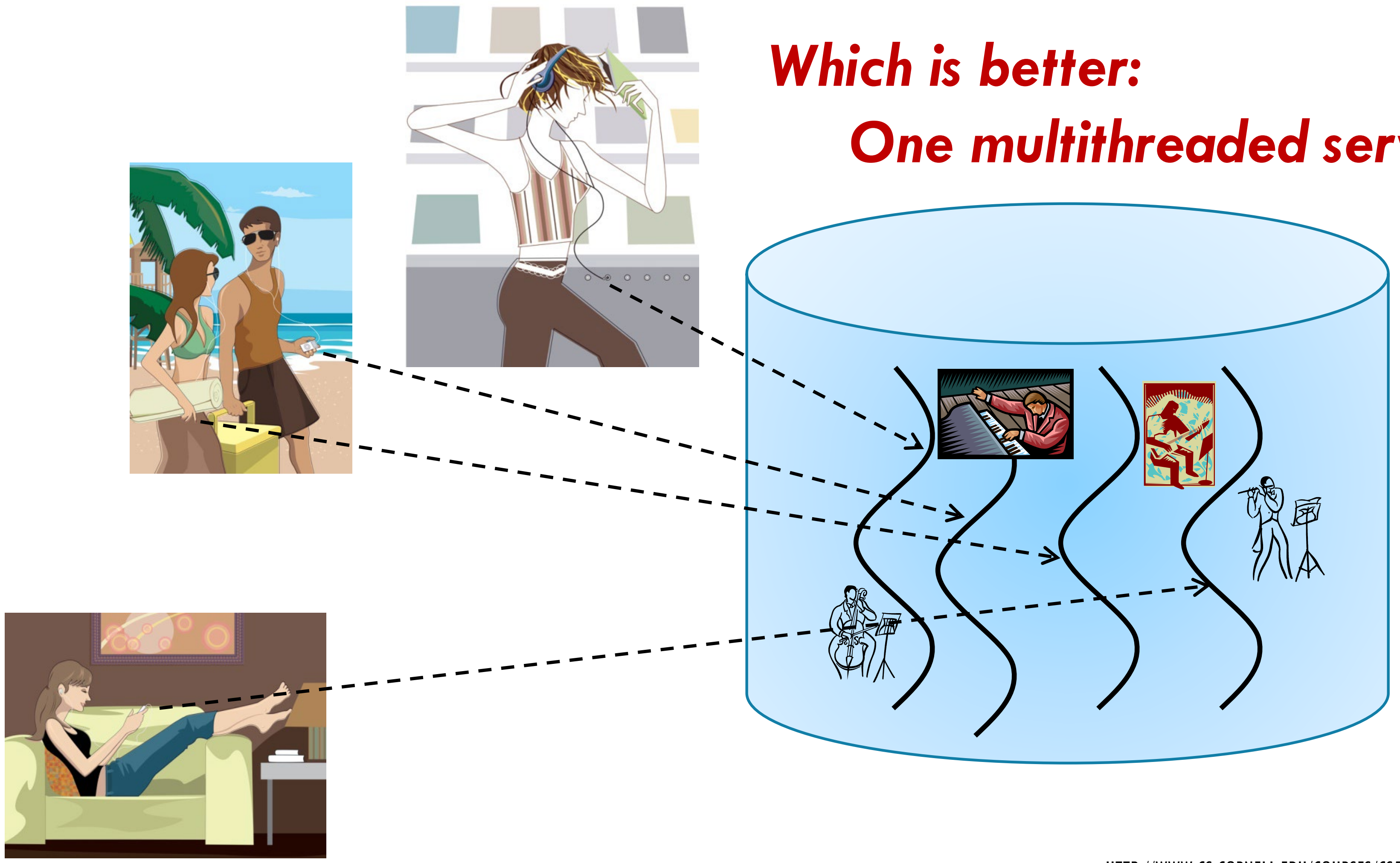
To see this, we'll spend a minute on just one example.

This may feel like a small detour but actually is typical of CS5412

TIER-ONE FOCUSES ON EASY STORIES

Which is better:

One multithreaded server, per node?

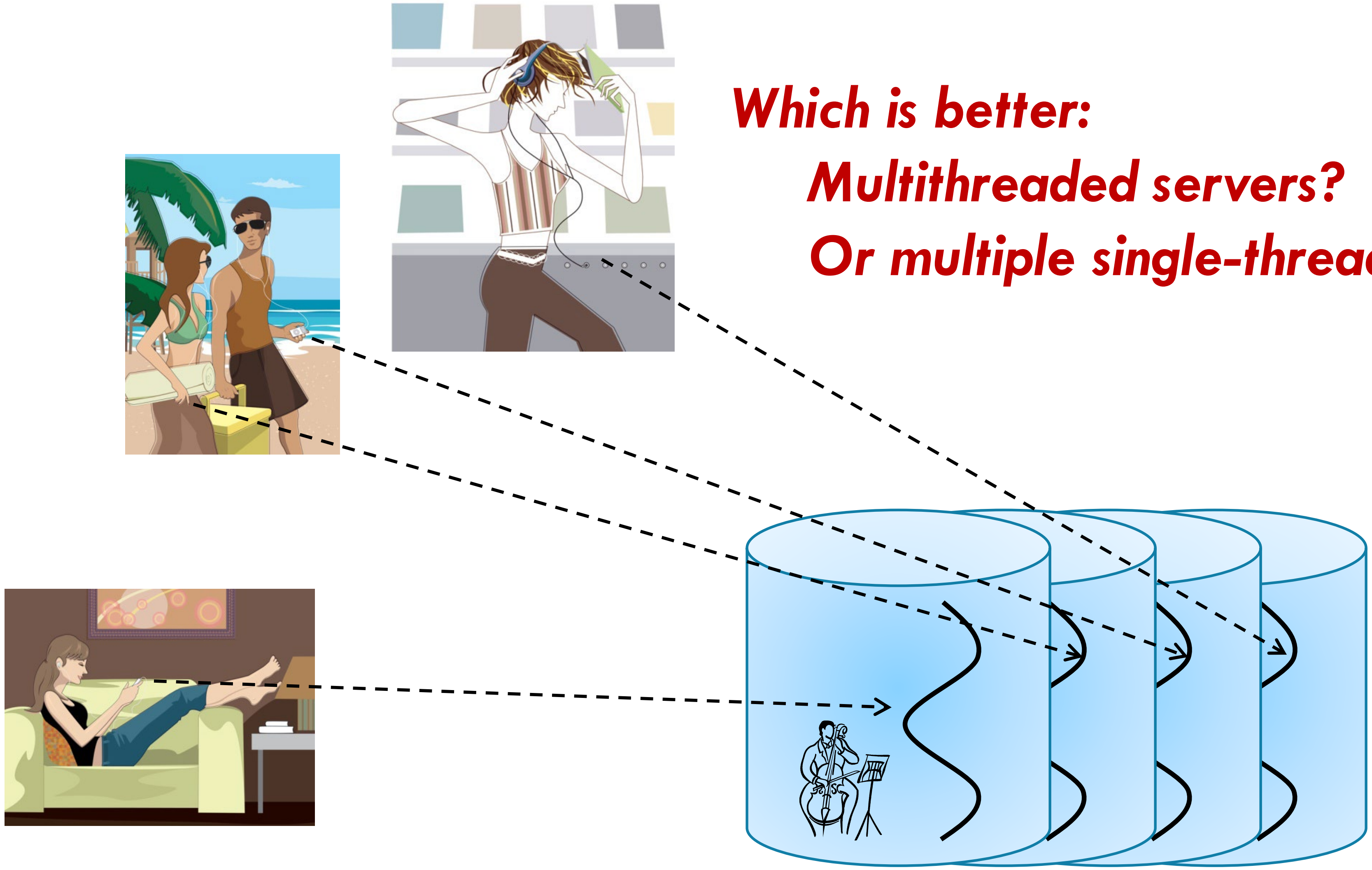


TIER-ONE FOCUSES ON EASY STORIES

Which is better:

Multithreaded servers?

Or multiple single-threaded servers?



Slight digression

WHERE IS THE POOL OF INSTANCES?

The μ-service in this example is the music player web site.

It runs on a set of servers, that each executes the same code. This is the pool of instances.

A single instance is one thread, handling one user.

WHAT YOU LEARNED IN O/S COURSE

Probably, you just took a class where the big focus was concurrency and threaded programs, and probably they taught you to go for multithreading

The story you heard was something like this:

- Because of Moore's law, modern computers are NUMA multiprocessors.
- To leverage that power, create lots of threads, link with a library like "pthreads", and request that your program be allocated multiple cores.
- Use thread synchronization/critical sections to ensure correctness.

BUT IS THIS THE RIGHT CHOICE?

First, we should identify other design options, even ones that look dumb at first glance.

Then we can evaluate based on a variety of considerations:

- Expected speed and scaling (more is good)
- Complexity of the solution (more is bad)
- Cost of the solution (more is bad)

WHAT YOU LEARNED IN O/S COURSE

Another thing you learned about was the *virtual machine* approach.

With true virtualization, programs run on private virtual machines.

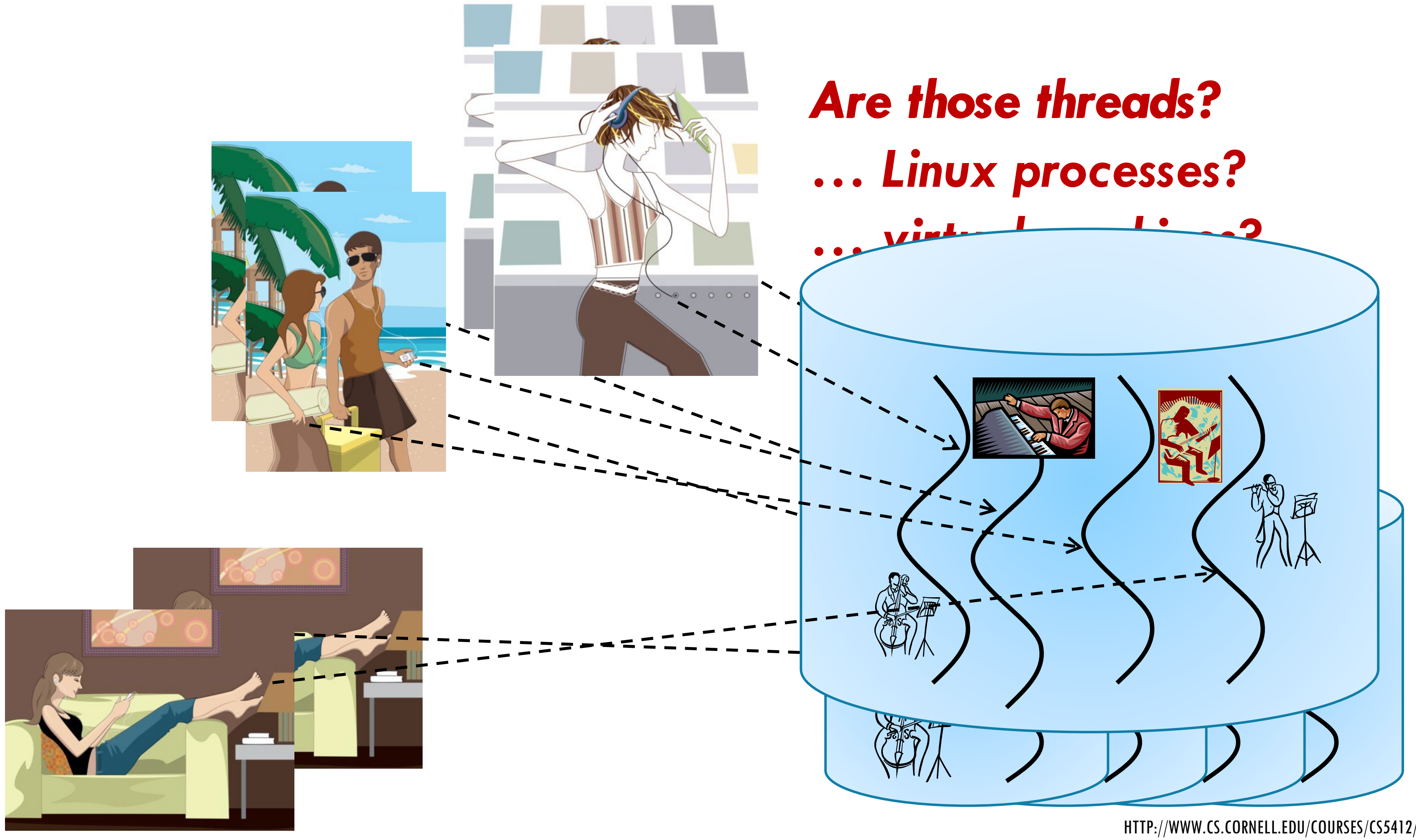
Today, the more modern and preferred alternative is “containers”, which give the illusion of a private virtual machine in a Linux process address space. A container is not a true VM, but “looks” like one.

… EVEN OUR "EASY" CLOUD POSES CHOICES!

Are those threads?

… Linux processes?

… virtual machines?



HOW WOULD YOU DECIDE BETWEEN THEM?

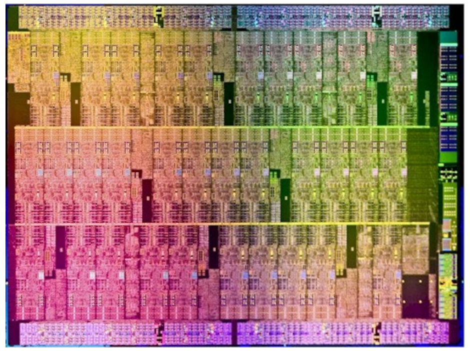
Basically, we have four options:

1. Keep my server busy by running one multithreaded application on it
2. Keep it busy by running N unthreaded versions of my application as virtual machines, sharing the hardware
3. Keep it busy by running N side by side processes, but don't virtualize
4. Keep it busy by running N side by side processes using containers

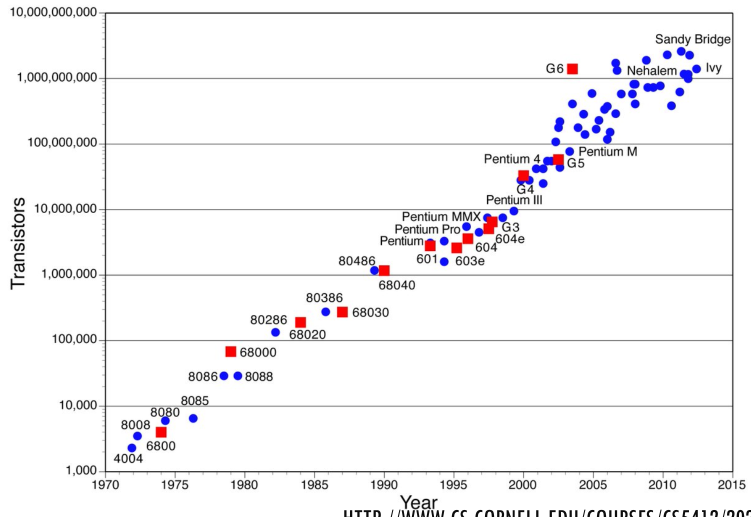
SURPRISE! A MULTI-THREADED SOLUTION IS THE WORST CHOICE!

This is almost always a surprise to CS5412 students. To appreciate the issue, we need to understand more about modern server hardware

Early days of the web were before we fell off Moore's curve. Today's servers are *NUMA machines* with many cores.



32-core Intel Aubrey chip. Some servers have as many as 128 cores today!



NUMA ARCHITECTURE

A NUMA computer is really a small rack of computers on a chip

Each has its own L2 cache, and small groups of them share DRAM.

With, say, 12 cores you might have 4 DRAM modules serving 3 each.

- Accessing your nearby DRAM is rapid
- Accessing the other DRAM modules is much slower, 15x or more

NUMA hardware provides cache consistency and locking, but **costs can be quite high if these features have much work to do.**

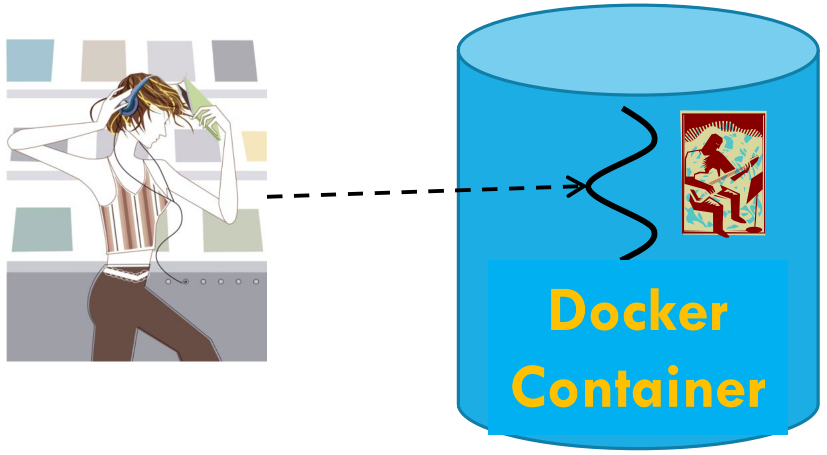
CAN WE WORK AROUND THOSE COSTS?

Yes!

CS4414 students learned about a whole grab-bag of ideas for avoiding NUMA overheads.

So they probably could build fast multithreaded code. But putting our servers in single-threaded applications within container VMs is easier.

WHAT MODEL WINS?



One "μ-player" per user, but many containerized instances per machine

Best is "container virtualization" with one process for each distinct user. This eliminates lock contention and memory sharing, allowing the NUMA code to run faster. It is also much easier to develop the code.

One NUMA server might host hundreds of these containers.

The "best" code can easily be 10x or even 100x faster than similar code that "fights" the hardware. A smart implementation will earn more money!

THIS LEADS TO AN INSIGHT!

In fact the way to approach this is to program in a way that matches best with the hardware.

In 2006 Linux and the main languages weren't very well-matched!

We saw the idea of a μ-service model, but those μ-instances still need to run on actual machines – Linux machines, and efficiently.

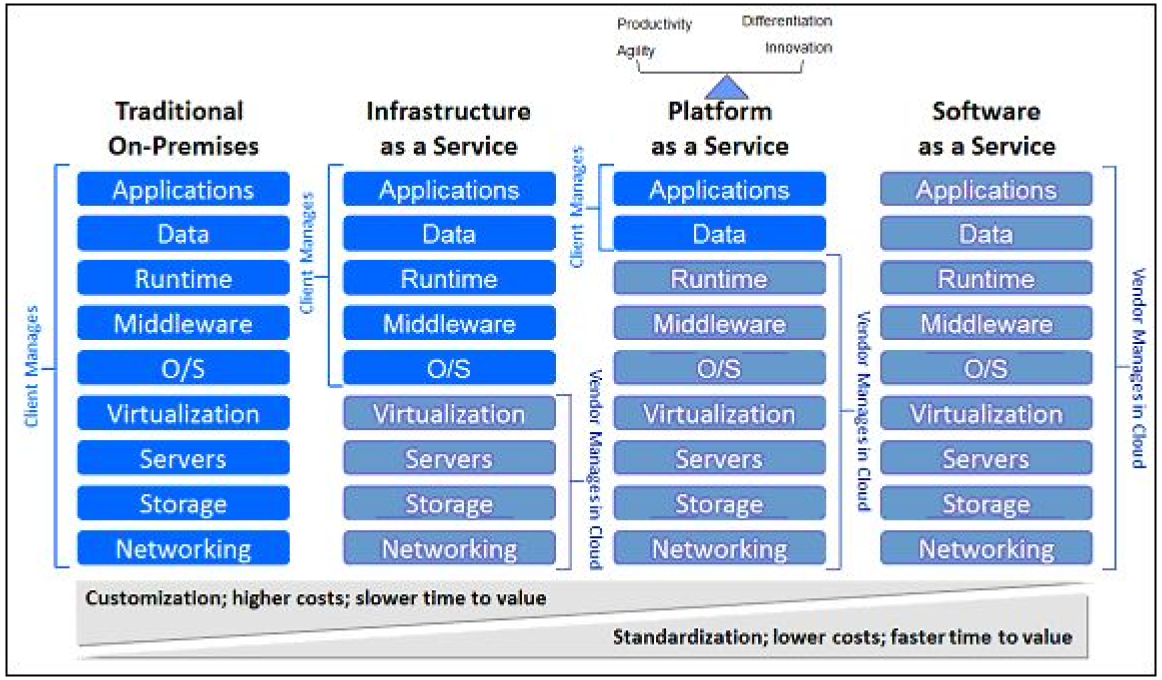
THE CLOUD IS EVOLVING TOWARDS PLATFORMS

You've seen all the XaaS acronyms.

Some, like IaaS, basically are "rent a machine, do what you like".

PaaS, meaning "platform as a service" (but really it means "customizable and extensible platform") is the most successful cloud model today.

PaaS Concept



The cloud favors PaaS.

Basically, the vendor offers all the standard code, pre-built. They create the platform in ways that perform and scale really well.

The developer will just plug in a small function (sometimes called a "lambda") to do the work, like playing back the requested video.

ANALOGY: PIZZA AS A SERVICE

You and your roommates suddenly *need* pizza. But you don't agree on which kind is best.

So you call down to Thompson and Bleecker, but ask if you can customize your order. You request a "quattro stagioni" (four seasons). They agree!

The pizzeria is offering a "platform" (the crust, the options…) and you are plugging in customization (the choice of topping and how to arrange them)

THOUGHT QUESTION

Can you think of a few other "uses" for a platform that would support video on demand, but really uses a lambda for the playback function?

Which parts would the platform be handling? How fancy can a lambda really be?

MORE TOPICS WE WILL TALK ABOUT

Azure's IoT Edge

- Sensors and actuators: what are they?
- How are digital twin solutions used?
- How do you customize an IoT cloud?
- Filtering and transforming streaming data

Fault Tolerance and Consistency

Challenges of dealing with real-time data

- Time synchronization, temporal storage
- Concepts of consistency for the cloud edge

Azure's Microservices Platform

- Details of the μ-services concept
- Customizing the intelligent cloud
- Roles played by edge μ-services
- Hardware accelerators for intelligence

Big data analytics to support IoT use cases

- The Apache ecosystem: Zookeeper, Hadoop, Pig, Hive, HBase, etc.
- Spark and its RDD model.

ORGANIZATIONAL STUFF

Spring 2020

ORGANIZATIONAL TOPICS/FAQ

Projects and extra credit opportunities

In-class quizzes

Final exam

ATTENDANCE

… is required, in some form.

Attending class in real time works best, and in person is even better. People who watch videos find it very hard to maintain focus and often fall behind, then get totally lost.

Cloud computing isn't a topic you can just sort of cram right before a test

YOUR FINAL GRADE

A curve. For most people B+ to A. A few get A+, a few get B or worse.

Formula?

- 50% comes from programming assignments (homeworks in weeks 1-4, then project). Extra credit can help with points lost here.
  - Extra credit comes from working with "smart dairy" students, hackathon, BOOM
  - Homework assignments often have extra credit built in, too.
- 50% from exams (in person if possible). Extra credit doesn't help here.

EVERYONE NEEDS TO DO A PROJECT!

You can work alone, or in teams. We encourage teaming, either with people you already know, or through “looking for teammates” on Piazza.

We can also help you form a team if you don’t know anyone.

Teams can accomplish more, and some team projects even become startups

PROJECT TOPICS

You are welcome to invent one of your own, but it has to be a really good fit for cloud computing and not just some random computing project.

These projects span a range of topics

- Big data + AI tools: *these will not get top grades in CS5412 unless they make use of continuously streamed "live" data in an automated way.*
- IoT Edge applications on dairy farm (limited number of groups)
- New cloud technologies layered on Cornell's Cascade platform
- Exciting web-hosted startup concepts that reach a proof of concept stage.

CS5999 (MENG PROJECT CREDITS)

Some people enlarge their CS5412 projects with our permission, based on a written proposal. In this situation they would add 3 credits of CS5999, which allows them to count the project towards MEng project credits.

But this means *six hours more work per week, starting this week!*

Those projects are always more ambitious, harder to build, and we closely monitor to make sure that they extra hours really were reflected in extra accomplishments. *They are <u>never</u> just extra credit for the same kind of work.*

AZURE ACCOUNTS

Our TAs will be providing Azure accounts you can use for CS5412

Azure is operated by Microsoft. You can select a variety of OS options such as Ubuntu Linux, Windows Server, etc. Linux is the most common.

Then you can log into your instances and set them up any way you like.