



# **CS 5412/LECTURE 1**

# **TOPICS IN CLOUD COMPUTING**

**Ken Birman**  
**Spring, 2020**

# MY QUALIFICATIONS TO TEACH THIS STUFF TO YOU...



- 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 SUPERSTAR PAST STUDENTS

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



Ranveer Chandra was the Farmbeats product lead at  
Microsoft. Now he is also chief scientist for Azure.



Yee -Jiun Song: VP Engineering, Facebook



Qi Huang: Owns Facebook content delivery



Dalia Malkhi: In charge of Libra, Facebook's new currency.





## ... **BLAH, BLAH, BLAH**

But in fact, I didn't actually invent the cloud. Jeff Dean (Google) did that, with Sanjay Ghemawat (Sanjay went to Cornell, but wasn't my student)

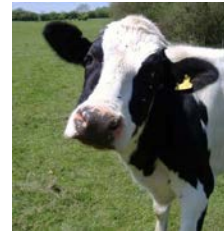
My real focus has always been on consistent data replication and fault-tolerance. The cloud uses this stuff, but it isn't the center of everything.

Still, I'm the guy Amazon turned to when they needed a lead expert witness after they were sued for billions over a cloud issue. (We won).

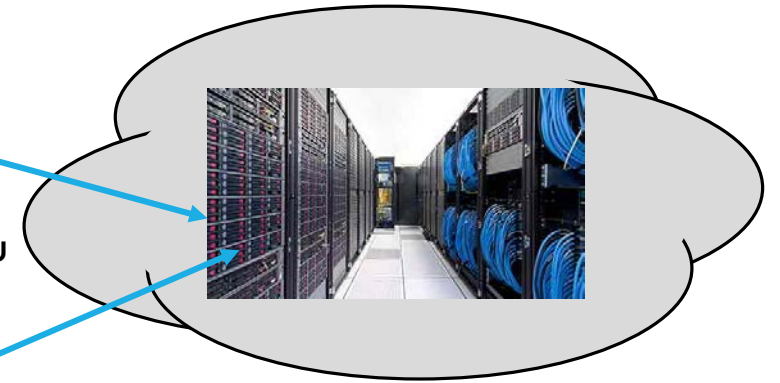
# CLOUD COMPUTING



Today... people like you



Tomorrow... Bessie!



CS5412 is...

- A deep study of a big topic.
- In spring 2020 our 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!**

# 250 PB/DAY!

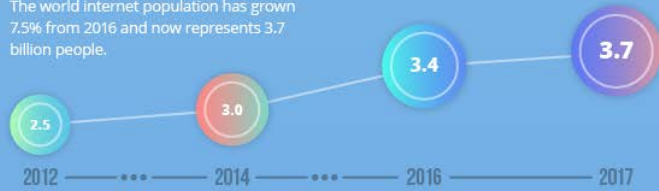


## DATA NEVER SLEEPS 5.0

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.

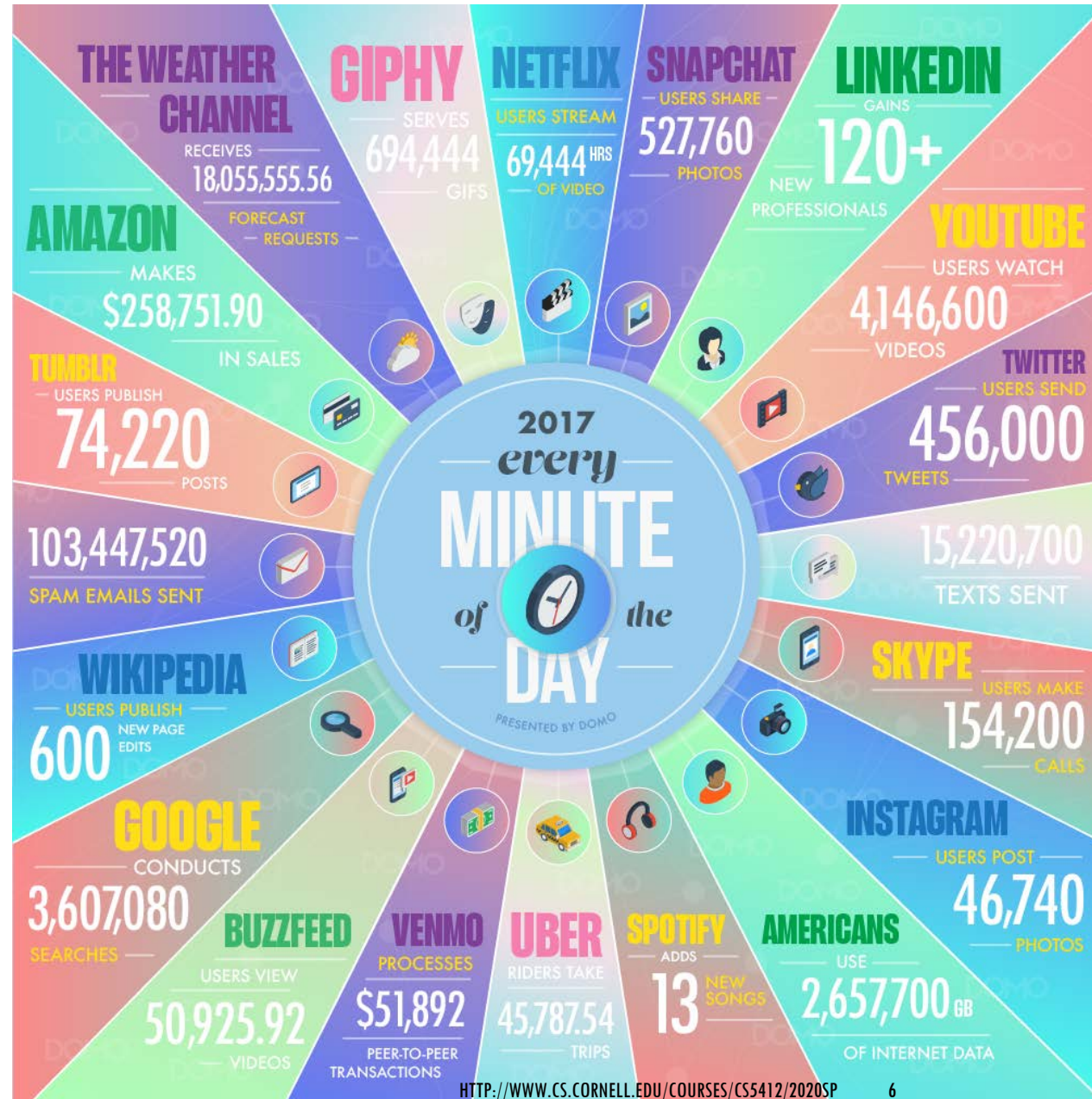


GLOBAL INTERNET POPULATION GROWTH 2012-2017 (IN BILLIONS)

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

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](http://domo.com)



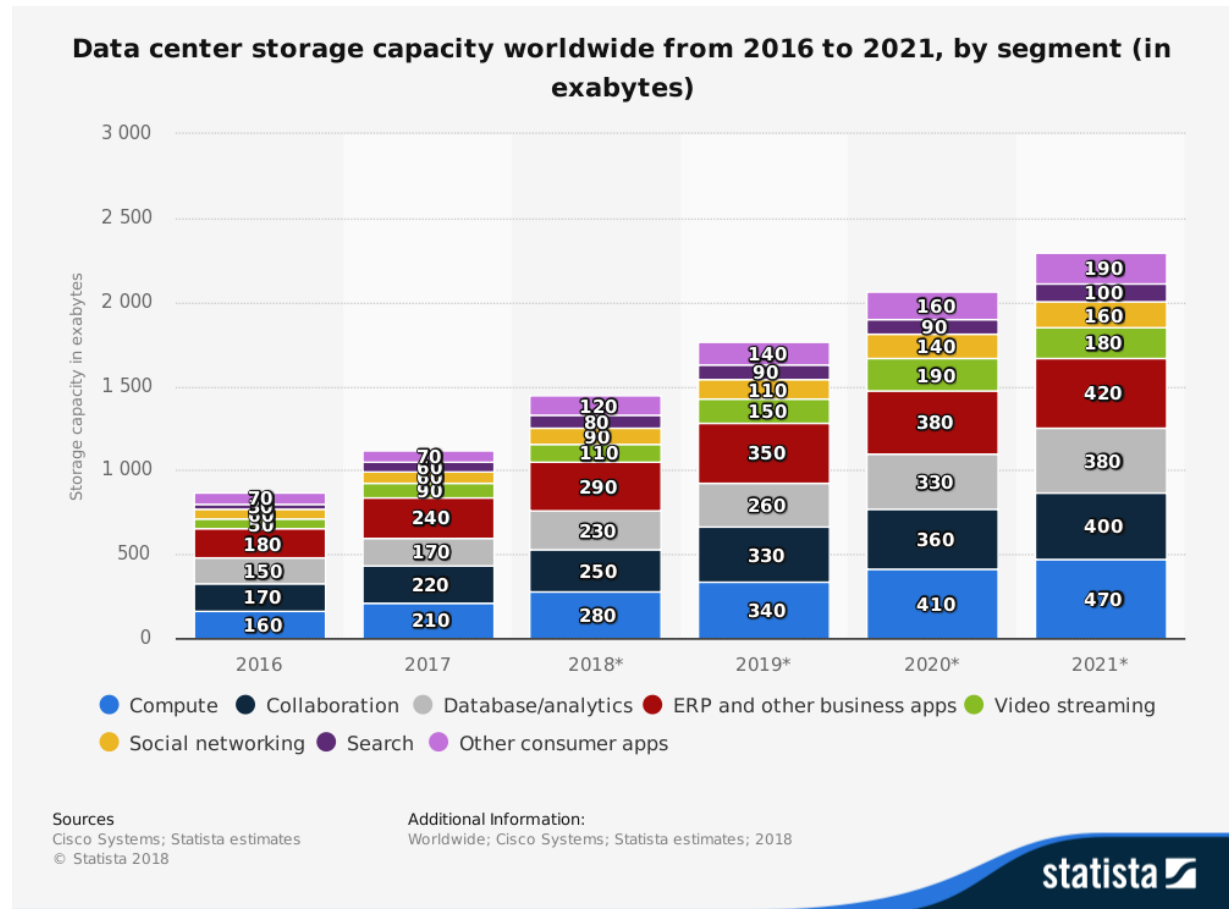


# DATA IN THE CLOUD

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”



# YOU TELL ME...

Total size of all the digital information acquired about you per day?

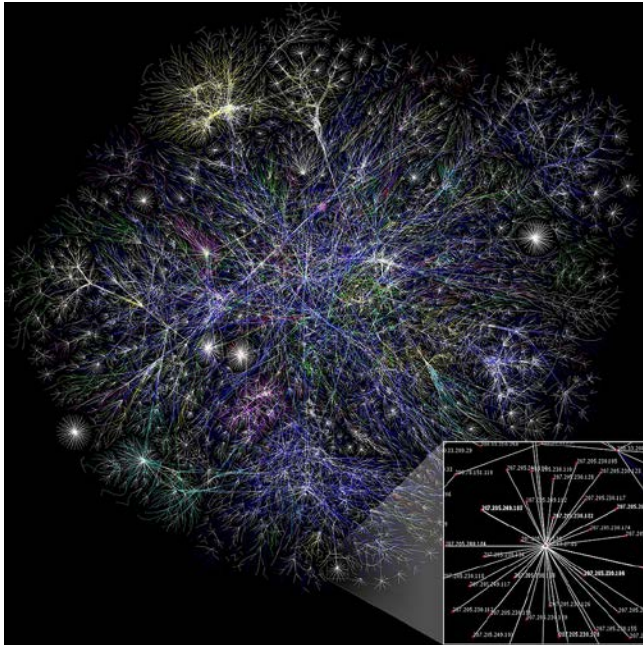
- 2 GB/day

Total amount of stored data in the Internet?

- 400 Exobytes

How many web sites in the Internet today?

- 1B



Speed of an Internet backbone link?

- 100 Gbps per fiber

Speed of an Internet backbone router?

- 322 Tbps

# CLOUD PROVIDERS NEED TO THINK “BIG”!

Google: 40,000 queries per second (1.2 Trillion per year)

YouTube: 1.9B active users per month, viewing 5B videos per day

Facebook: 2.23B active users, 8B video views, 1.5M photos uploaded per day

Cloud: Nearly 4B of the world's 7B accessed cloud resources in 2018

... the scale of computing to support these stories is just surreal!

# ... AND THEY NEED TO THINK “PARALLEL”

At these scales, no computer can keep up.

By the nature of the cloud, it *has* to be massively parallel!



# YOU TELL ME...

How much DRAM in a datacenter server?

- 512GB-12TB

How fast is a single CPU in a NUMA machine?

- 1.8Ghz

How many cores does a NUMA server hold?

- 72 cores

How many threads per core when hyperthreading is enabled?

- 2

How many servers per rack?

- 48

How many servers total in a datacenter?

- ~500,000

How deep is a typical datacenter COS/SPINE routing tree?

- 6 layers

How fast is a datacenter network today?

- 56Gbps

How big is 1-way node to node latency (due purely to the network)?

- 1.25us

What is a typical round-trip latency for a datacenter RPC?

- 100us-25ms

# YOU TELL ME...

How much storage capacity in a server's RAID SSD drive?

- 800 GB

How much in a RAID configuration?

- 100 TB

How much storage in a cutting edge rotating disk (HDD)?

- 15 TB

How much capacity in a memory-mapped Optane drive?

- 16 GB

Peak PCIe bus data transfer speeds, per bus?

- 8.5GB/s

... peak transfer rate for an single SSD unit?

- 200 MB/s

... access delay for a block of SSD storage?

- 100us

Seek time for an HDD?

- 2.5-10ms

# YOU TELL ME...

Size of one email in HTML encoding?

- 10KB

... a typical raw photo?

- 4MB

... that same typical photo, in a compressed format?

- 250KB

... a typical encoded 3-5 minute music video?

- 10MB/min

... a full length movie?

- 4GB

Maximum standard IP packet size?

- 1KB

Jumbo frame size?

- 8KB

# 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

# YAHOO EXPERIMENT



*A sprint to render your web page!*

At Yahoo, they tried an “alpha/beta” experiment

Customers who saw fast web page rendering (below 100ms) were happy.

For every 100ms delay, *purchase rates noticeably dropped.*

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

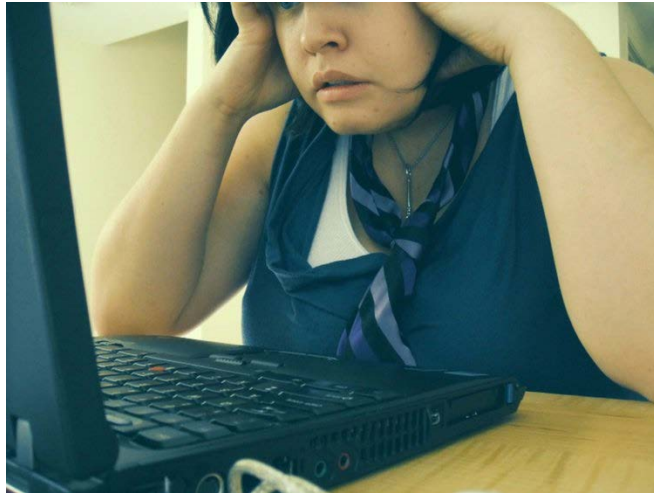
# STARTING AROUND 2006, AMAZON LED IN REINVENTING DATA CENTER COMPUTING

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  $\mu$ -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)

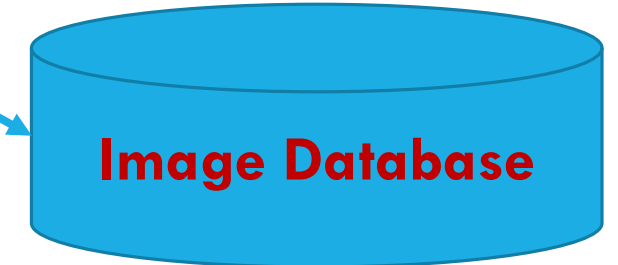


Computers were mostly desktops

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)

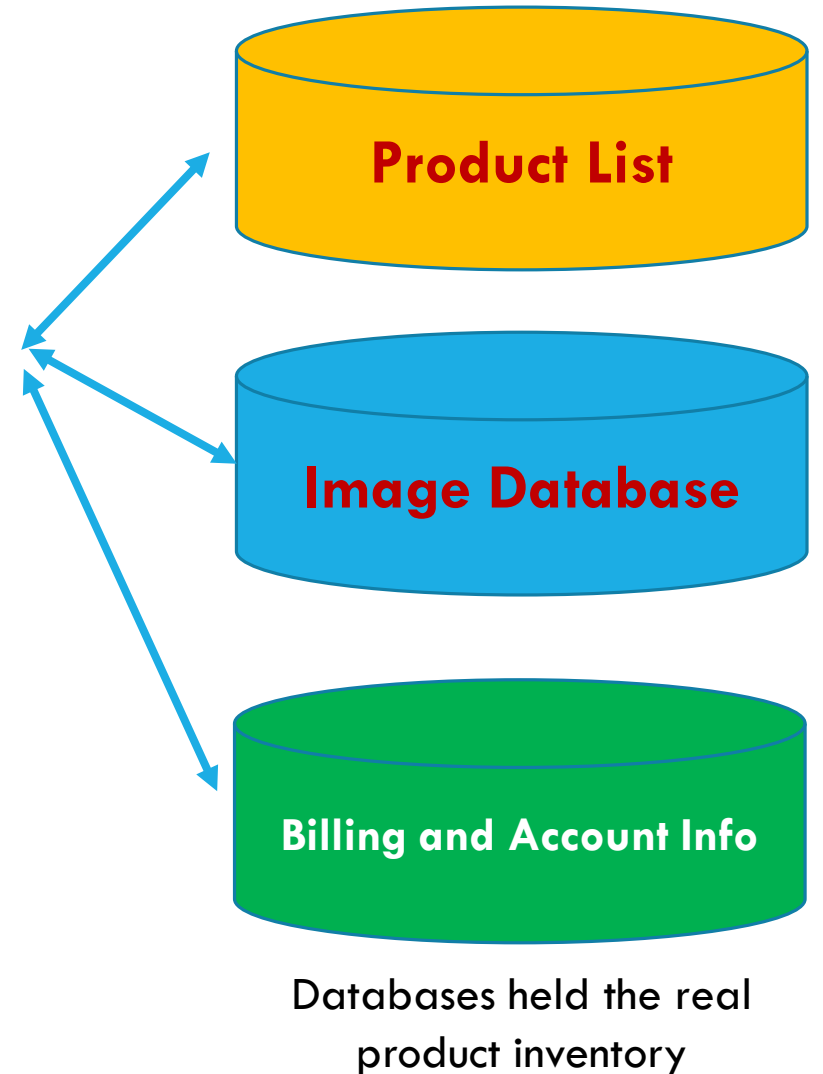


Computers became lightweight,  
yet faster

*Routed to nearest  
datacenter, one of many*



Web Server built the page...  
ten miles from the users





# NEW APPROACH (2008)



Computers became lightweight,  
yet faster



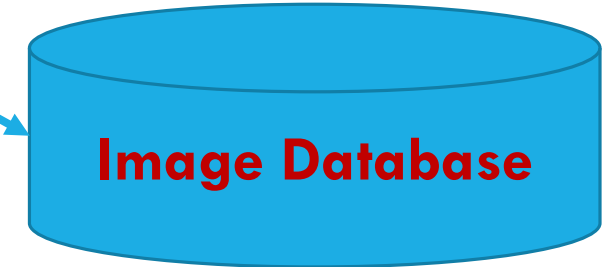
More and more mobile apps

Routed to nearest  
datacenter, one of many



Web Server built the page...  
ten miles from the users

Backup routing options



Databases held the real  
product inventory

# NEW APPROACH (2008)



Desktops with snappier response



More and more mobile apps

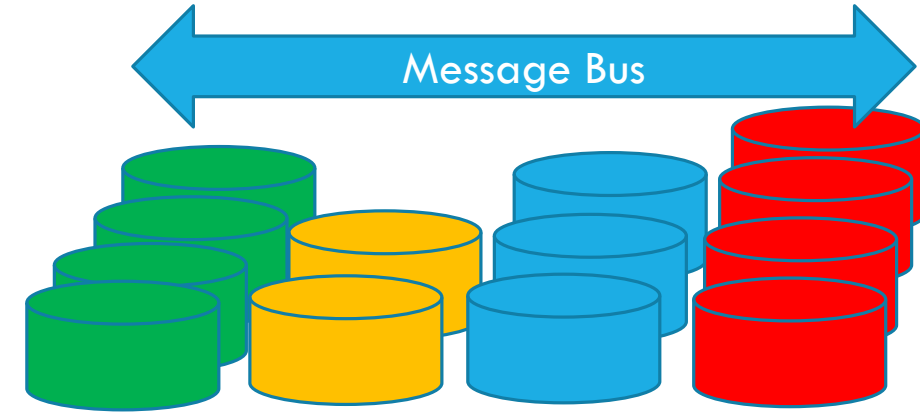
Routed to nearest datacenter, one of many



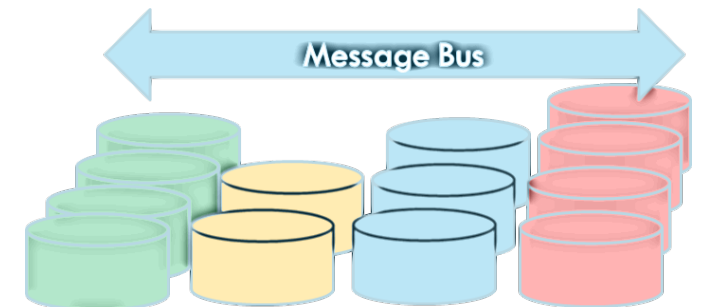
Web Server becomes simpler and does less of the real work



GeoReplication



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.



# 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

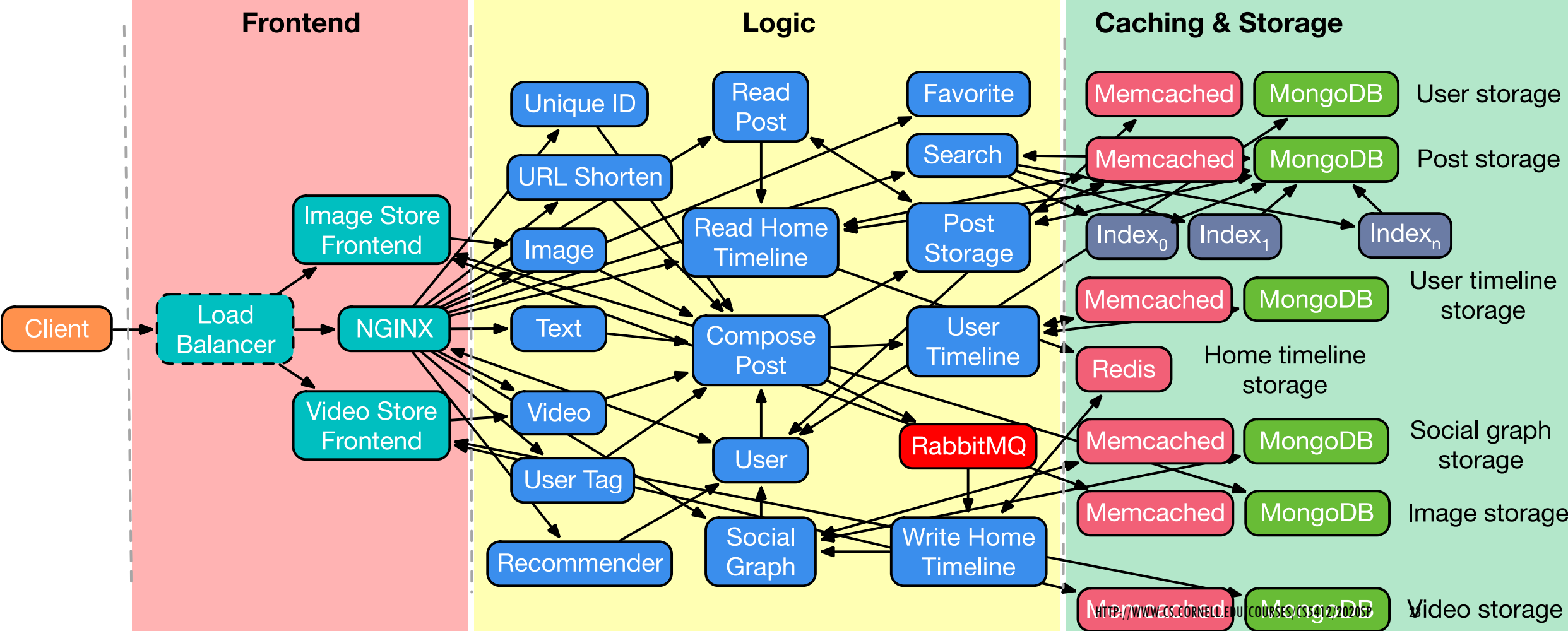
Tier one runs on very lightweight servers:

- 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  $\mu$ -Services specialize in various aspects of the content delivered to the end-user. They may run on somewhat “beefier” computers.

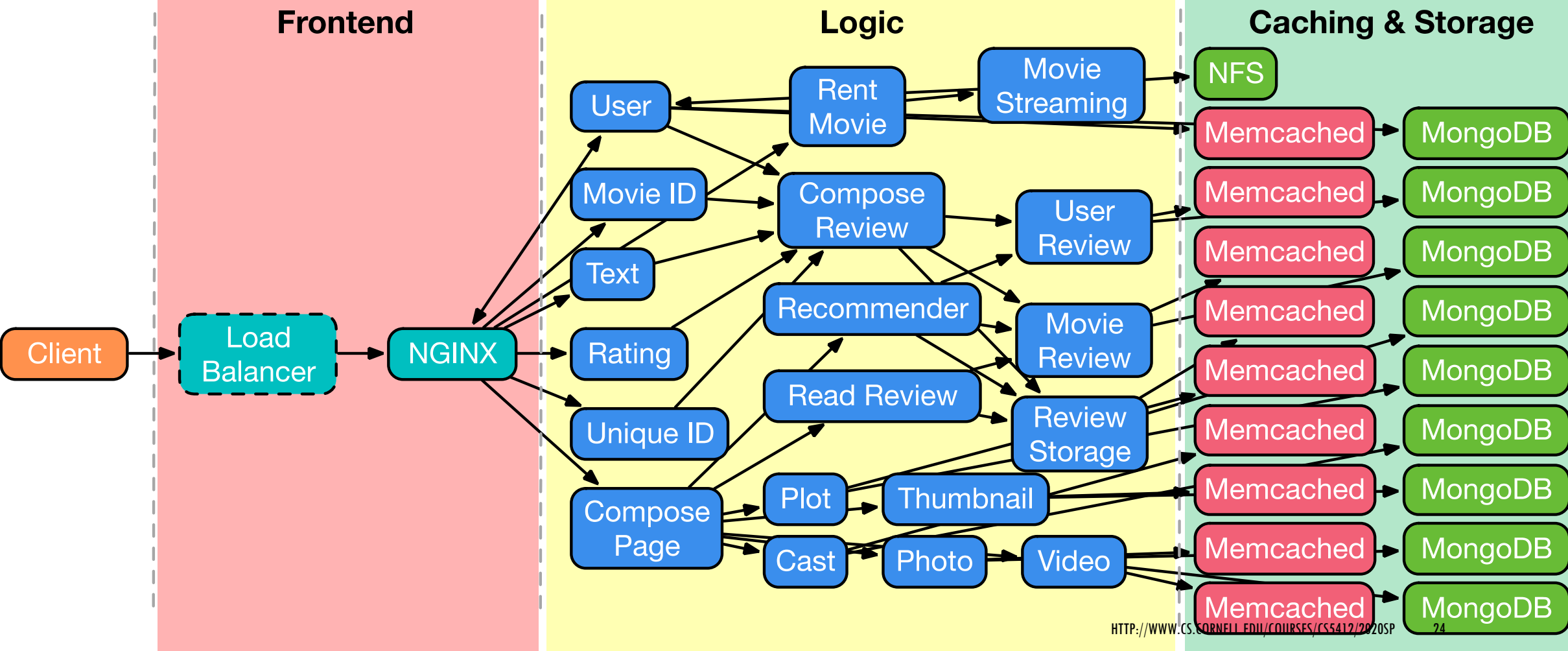
# End-to-end Microservices (from Christina Delimitrou)

## Social Network



# End-to-end Microservices (from Christina Delimitrou)

## Media Service





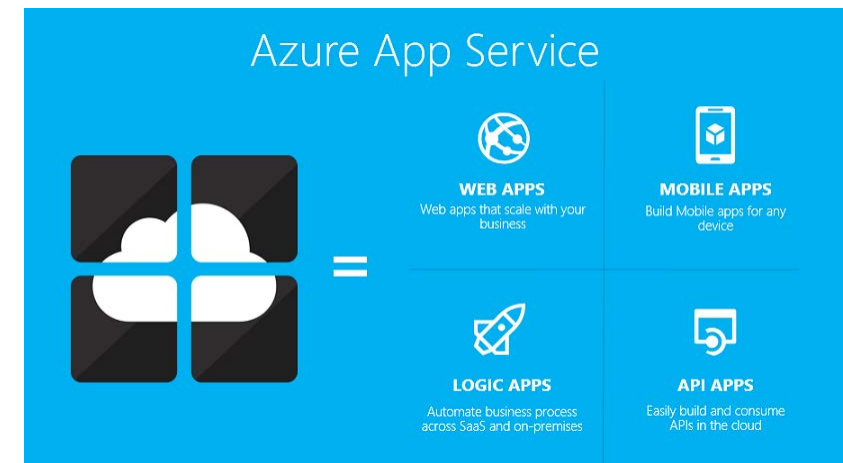
# EACH MICROSERVICE IS A PARALLEL “POOL”!

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

A microservice ( $\mu$ -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

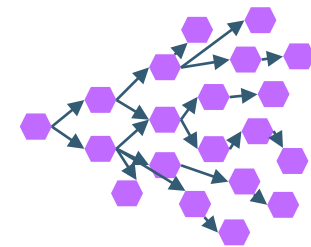


In Azure, for example, there is a tool called the “App Service” (we’ll use it!)

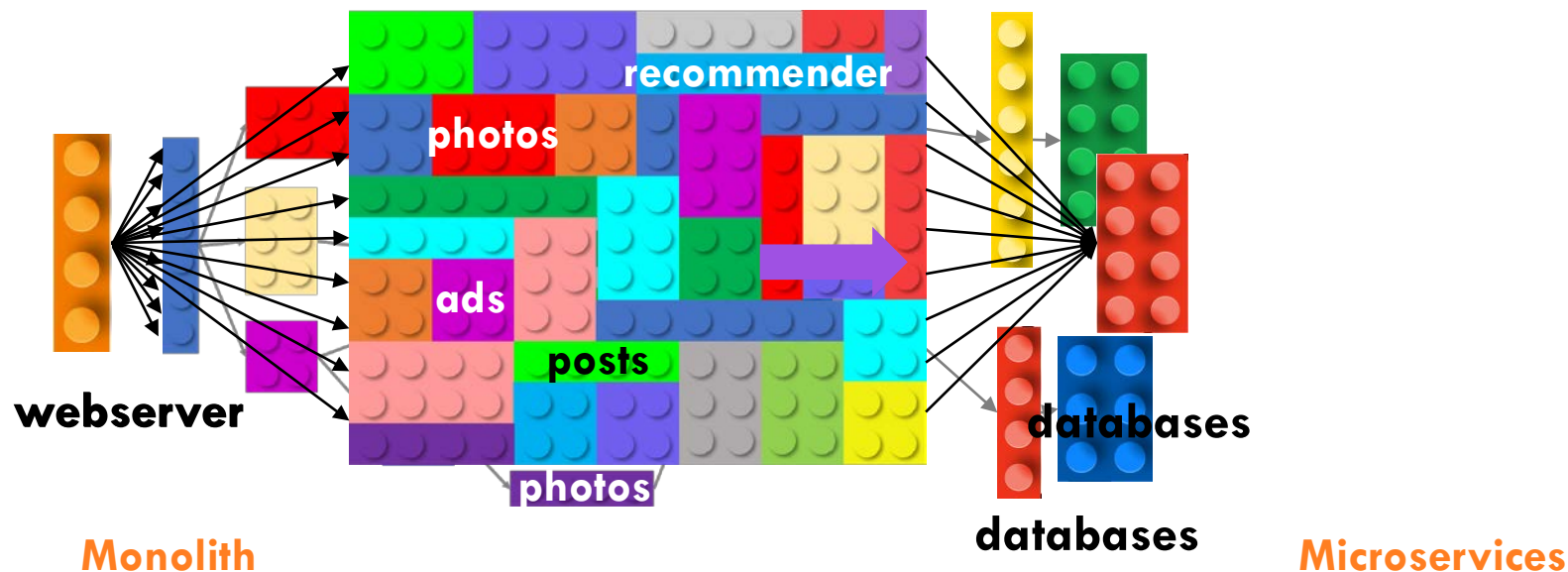
The App Service manages a big collection of compute resources in the cloud. Developers can install your own services in it (as “containers”). Configuration files tell it when to launch them for you, automatically.

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.





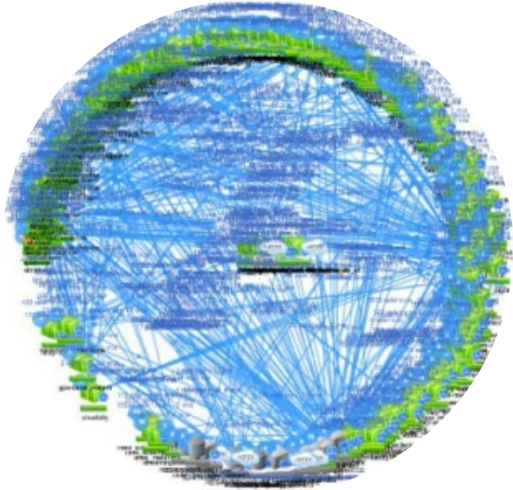
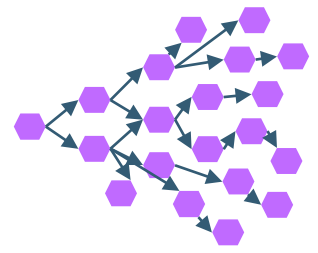
# Motivation for $\mu$ -services (Delimitrou)



## Advantages of $\mu$ -services:

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

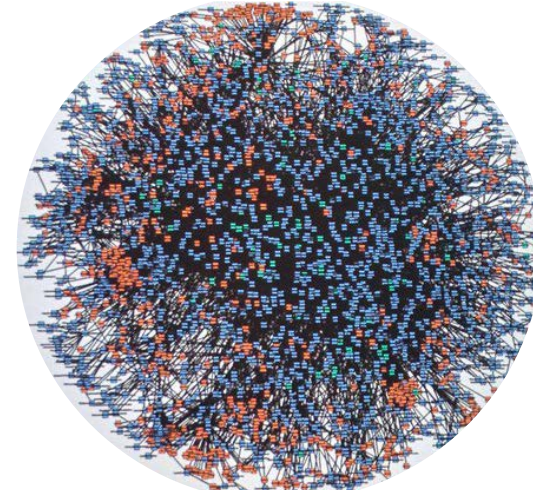
# Performance management (Delimitrou)



Netflix



Twitter



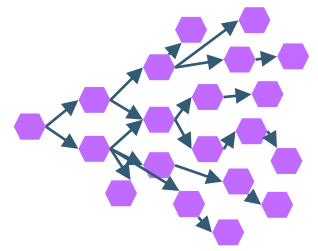
Amazon

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  $\mu$ -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 $\mu$ -SERVICE INSTANCE

We've been acting as if each  $\mu$ -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!

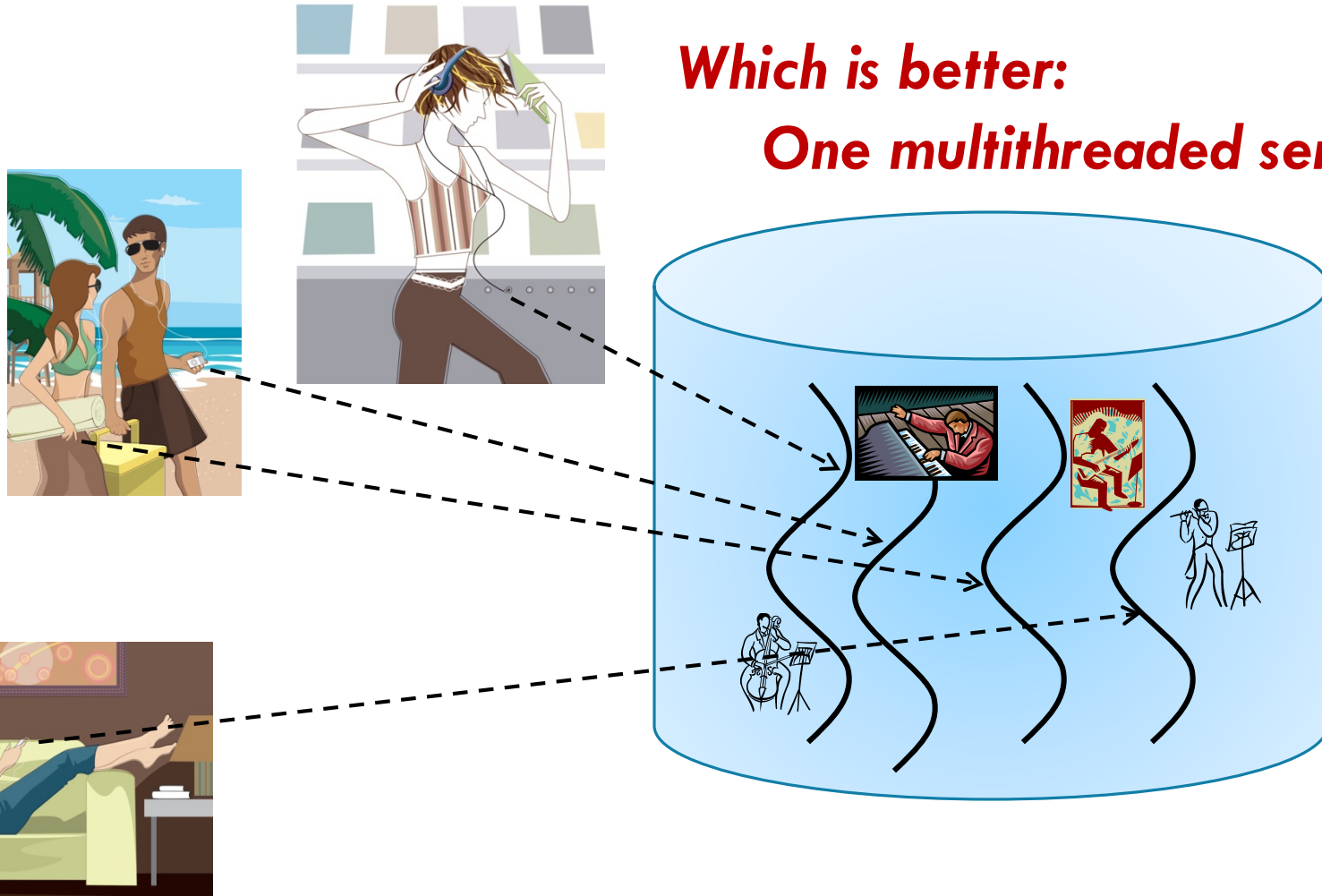
To see this, we'll spend a minute on just one example.

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?*

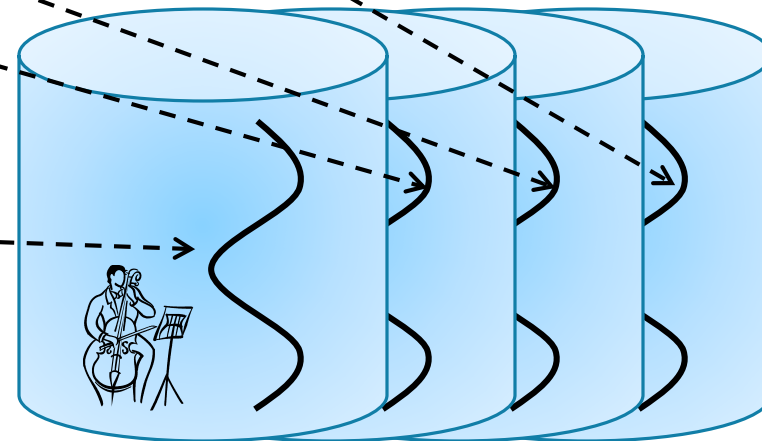


**Slight digression**

# TIER-ONE FOCUSES ON EASY STORIES



*Which is better:  
Multithreaded servers?  
Or multiple single-threaded servers?*



**Slight digression**

# 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)

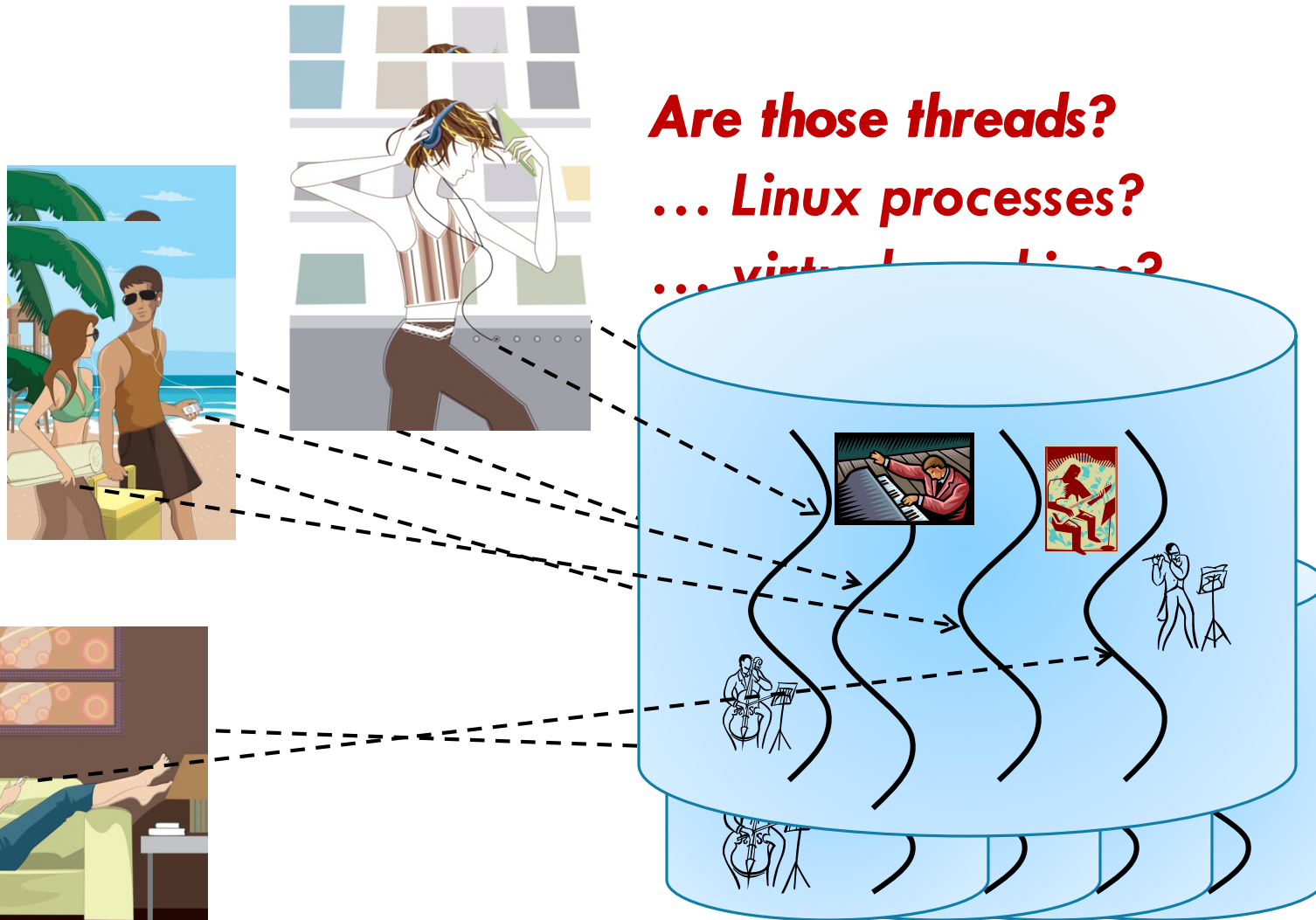
# WHAT YOU LEARNED IN O/S COURSE

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

With true virtualization, programs run on private virtual machines.

Today, a recent alternative is “containers”, which give the illusion of a private virtual machine in a Linux process address space, not a true VM.

# ... EVEN OUR "EASY" CLOUD POSES CHOICES!



Slight digression

# HOW WOULD YOU DECIDE BETWEEN THEM?

Basically, we have four options:

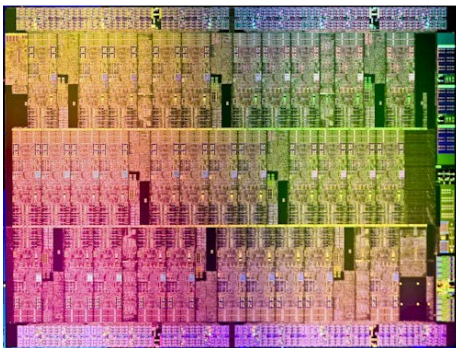
1. Keep my server busy by running one multithreaded application on it
2. Keep it busy by running N unthreaded versions of my application as virtual machines, sharing the hardware
3. Keep it busy by running N side by side processes, but don't virtualize
4. Keep it busy by running N side by side processes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



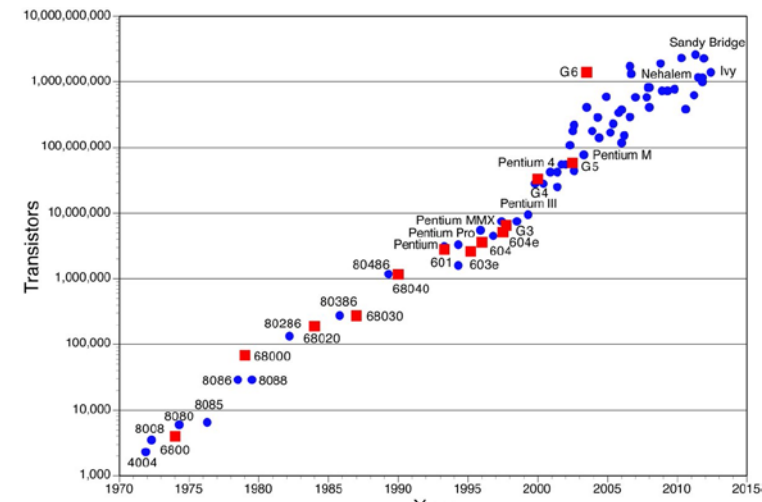
# WHY DOESN'T A MULTI-THREADED SOLUTION PERFORM BEST?

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!



Slight digression

# 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, 1.5x or more

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

# MULTITHREADING ON A NUMA.



On a NUMA architecture, many threaded programs *slow down* on  $> 1$  cores! Many reasons:

- Locking and NUMA memory coherency,
- Weak control over “placement” (which memory is on which DRAM?),
- Higher scheduling delays,
- Issues of reduced L2 cache hit rate

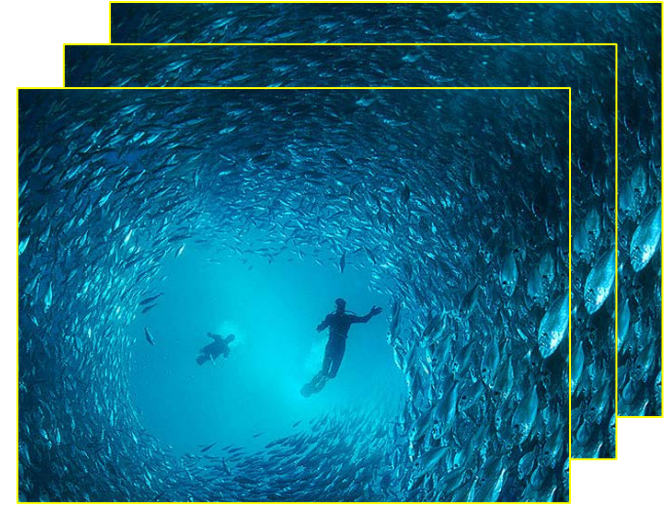
# OTHER OPTIONS: VMS AND CONTAINERS.



Another approach is to just run multiple programs side by side. But how can we avoid risk of interference?

- **Virtual machines (VMs)** are a tool for making one compute emulate a cluster of computers and even run several different O/S instances.
- **Containers** are a different tool that does less but has lower overheads. Each container has one or a set of processes that are isolated from other containers, but not with a full VM.

# DEEP DIVE ON THAT QUESTION



How does true virtualization work?

- The VM solution (like VMWare or Oracle VirtualBox) has a way to package a whole computer as a single executable.
- Now we won't have memory contention: each program lives in an isolated context. We use more memory, because sharing is harder.
- But the virtualization layer causes page-table translation slowdown, and I/O operations might also be slower. DMA might not work.

# DEEP DIVE ON THAT QUESTION



A container is a normal Linux process with a library that mimics a full VM.

- The system looks “private” but without full virtualization.
- Eliminates the 10% or so performance overheads seen with true VMs.
- Also, containers launch and shut down much faster than a full VM, because we don’t need to load the whole OS.
- We won’t see NUMA memory contention problems.
- Security and “isolation” are nearly as good as for VMs.

Popular options? Kubernetes and Docker.



# CHOICES AS A TABLE

Option	Speed	Complexity	Costs
1. <b>Multithreaded server</b>	Poor	Poor	<b>Development: expensive.</b> Use of resources: best. But may be <b>hard to administer.</b>
2. <b>Single-thread + VM</b>	Poor	Good	Inexpensive development, but <b>inefficient use of memory resources, high overheads</b>
3. <b>Single-threaded process model</b>	Very good if interference can be avoided	Least complex!	Inexpensive development, but administering to ensure that the processes won't somehow interfere can be tricky.
4. <b>Single-threaded, containers</b>	<b>Best of all.</b>	<b>Just like a single-threaded process model.</b>	<b>Inexpensive development. The approach helps by protecting containerized apps from most forms of interference.</b>



# WHY CONTAINER MODEL “WINS”...

We want the edge of the cloud to be as cost-effective as possible.

Development and management complexity is one kind of cost.

Also think about CPU load, memory, and context switching overheads:

- Best would be a single program with multiple threads
- Containers offer isolation and can share code pages, saving memory

# AHA!

*Build single threaded server, optimized to run on behalf of a single client.*

*Run lots of copies on each NUMA server (perhaps hundreds).*

*Use containers for isolation, container O/S smart about DRAM memory issues.*

- *Share read-only pages only between cores that share the same DRAM*
- *Make one copy per DRAM for read-only shared data, like code pages!*

# MORE TOPICS WE WILL TALK ABOUT

## Azure's IoT Edge

- Sensors and actuators: what are they?
- How smart are they likely to be?
- Customizing the IoT Edge
- Filtering and transforming data

## Fault Tolerance

### Challenges of dealing with real-time data

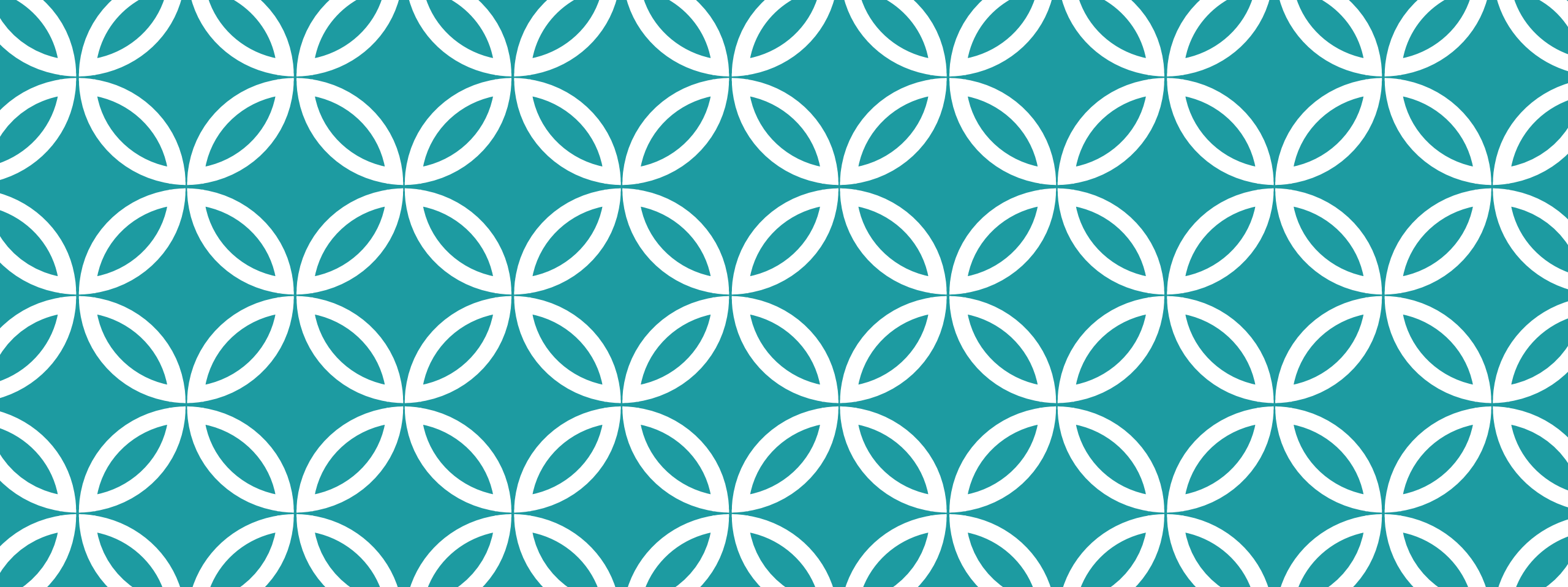
- Time synchronization, temporal storage
- Concepts of consistency for the cloud edge

## Azure's Intelligent IoT Cloud

- Details of the  $\mu$ -services concept
- Customizing the intelligent cloud
- Roles played by edge  $\mu$ -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

# YOUR FINAL GRADE

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

Formula?

- Half 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.
- Half from exams (quizzes, then a final). 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.

Many people want suggestions. In spring 2020 we are teaming up with some “smart farming” researchers and they have great ideas:

- Various cloud-computing + IoT applications seen on dairy farms.
- Drone and robot image processing that arises in greenhouses.

# CS5999 (MENG PROJECT CREDITS)

Some people expand CS5412 projects by adding 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.

# 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.