EECS 489
Computer Networks

Winter 2024

Mosharaf Chowdhury

Material with thanks to Aditya Akella, Sugih Jamin, Philip Levis, Sylvia Ratnasamy, Peter Steenkiste, and many other colleagues.

Agenda

- Video streaming
- Datacenter applications

Why is video important?

- Dominates the global Internet traffic landscape
  - About 65%, i.e., every 2 of 3 bytes in 2023!
- Major sources
  - Netflix
  - YouTube
  - …

How to watch a video?

- Download and watch
  - Often too large to send in one GET
  - Doesn't even make sense even if its possible
    - » Users have to wait too long
    - » Users may skip forward! ⇒ bandwidth waste
    - » User's connection quality may change (e.g., switching from WiFi to LTE) ⇒ lower resolution to match bandwidth

- Our focus is *not* live video

The video medium

- Video is a sequence of images/frames displayed at a constant rate (moving pictures)
- Digital image is an array of pixels, each pixel represented by bits
- Examples:
  - Single frame image encoding: 1024x1024 pixels, 24 bits/pixel ⇒ 3 MB/image
  - Movies: 24 frames/sec ⇒ 72 MB/sec
  - TV: 30 frames/sec ⇒ 90 MB/sec

The video medium (cont'd)

- Compression is key
  - Lots of algorithms to compress
- The same video can be (and typically is) compressed to multiple quality levels
  - E.g., 480p, 720p, 1080p, 4K
- Why multiple resolutions?
  - Adapt to conditions

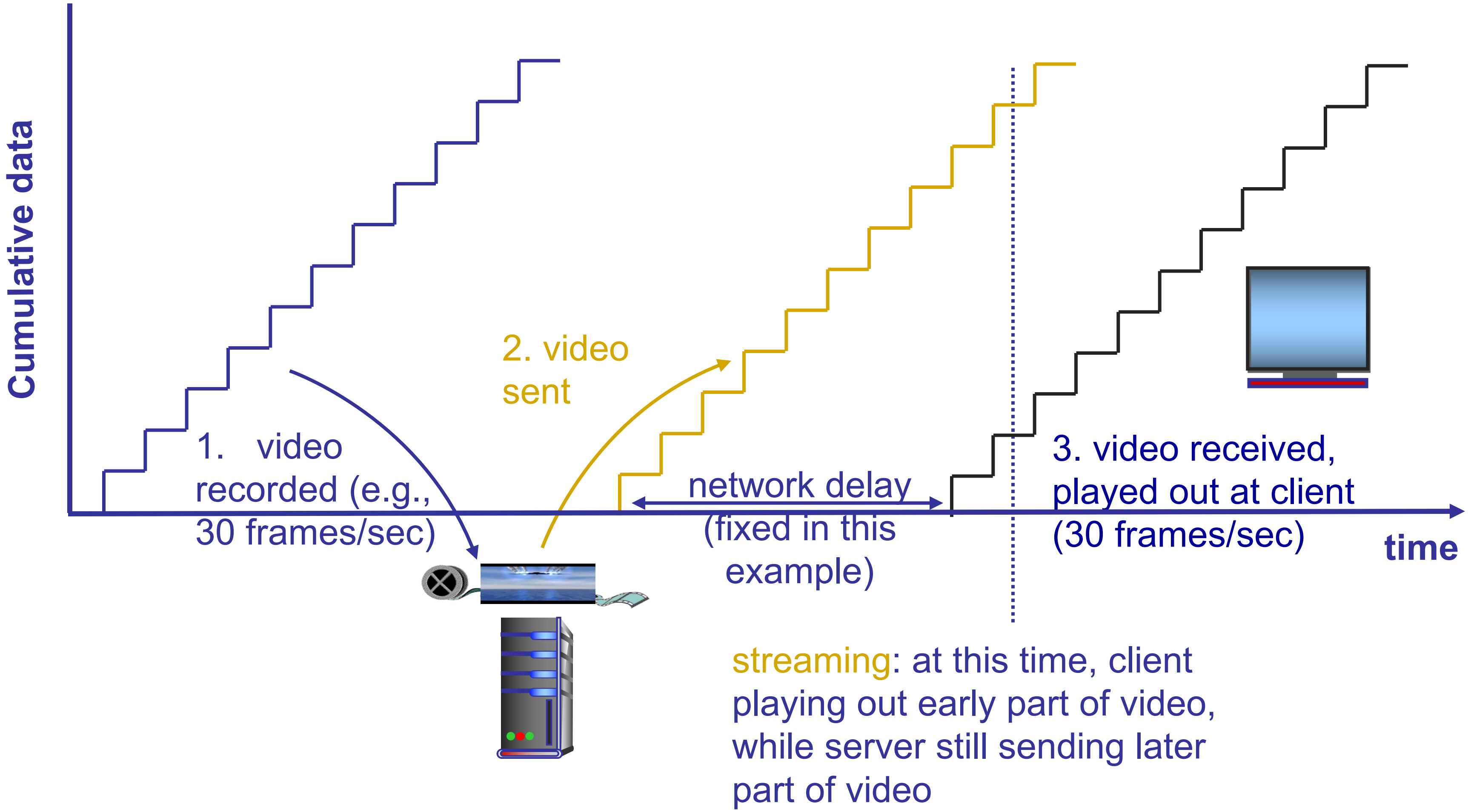
How do we serve video?

- It's in the name!
  - Video streaming

HTTP streaming

- Video is stored at an HTTP server with a URL
- Clients send a GET request for the URL
- Server sends the video file as a stream
- Client first buffers for a while. Why?
  - To minimize interruptions later
- Once the buffer reaches a threshold
  - The video plays in the foreground
  - More frames are downloaded in the background

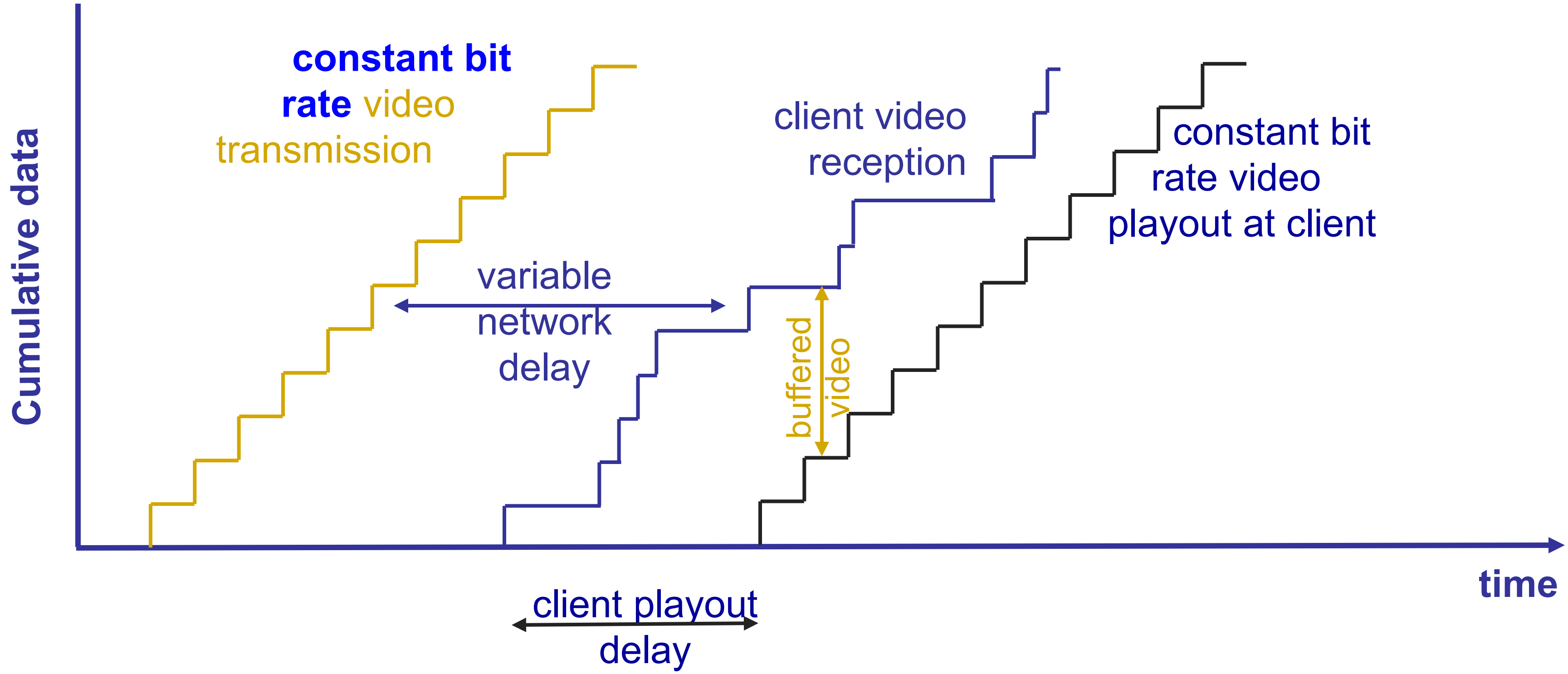
HTTP streaming



Challenges

- Absorb network delay variations
- Handle user interactions
  - Jump forward, fast-forward, rewind, pause
- Handle packet loss, retransmission etc.

HTTP streaming: Revisited



- Client-side buffering and playout delay: compensate for network-added delay, delay jitter

Issues with HTTP streaming

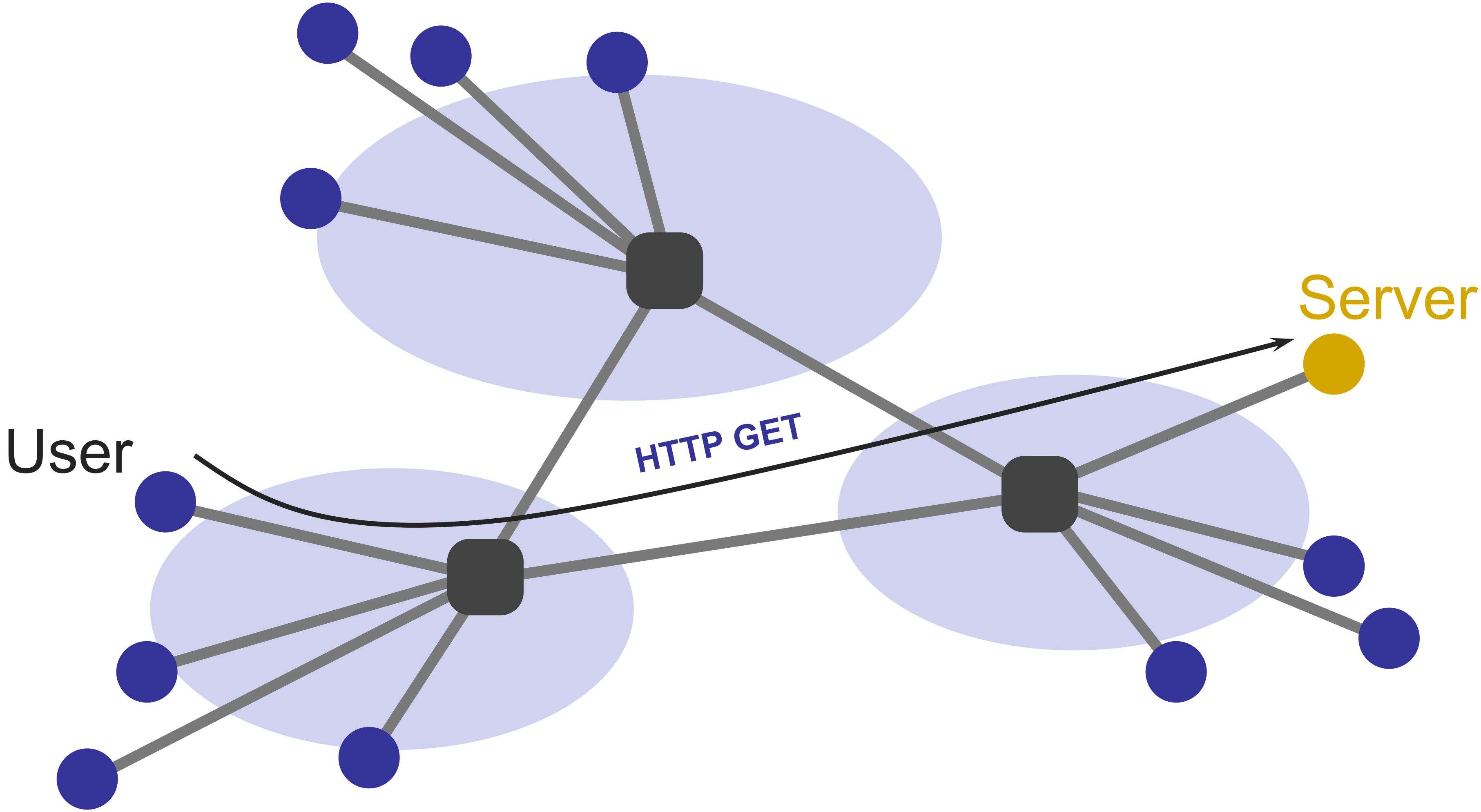
- Same bitrate for all clients
  - Clients can have very different network conditions
  - Clients network conditions can change over time
- Cannot dynamically adapt to conditions

DASH : Dynamic Adaptive Streaming over HTTP

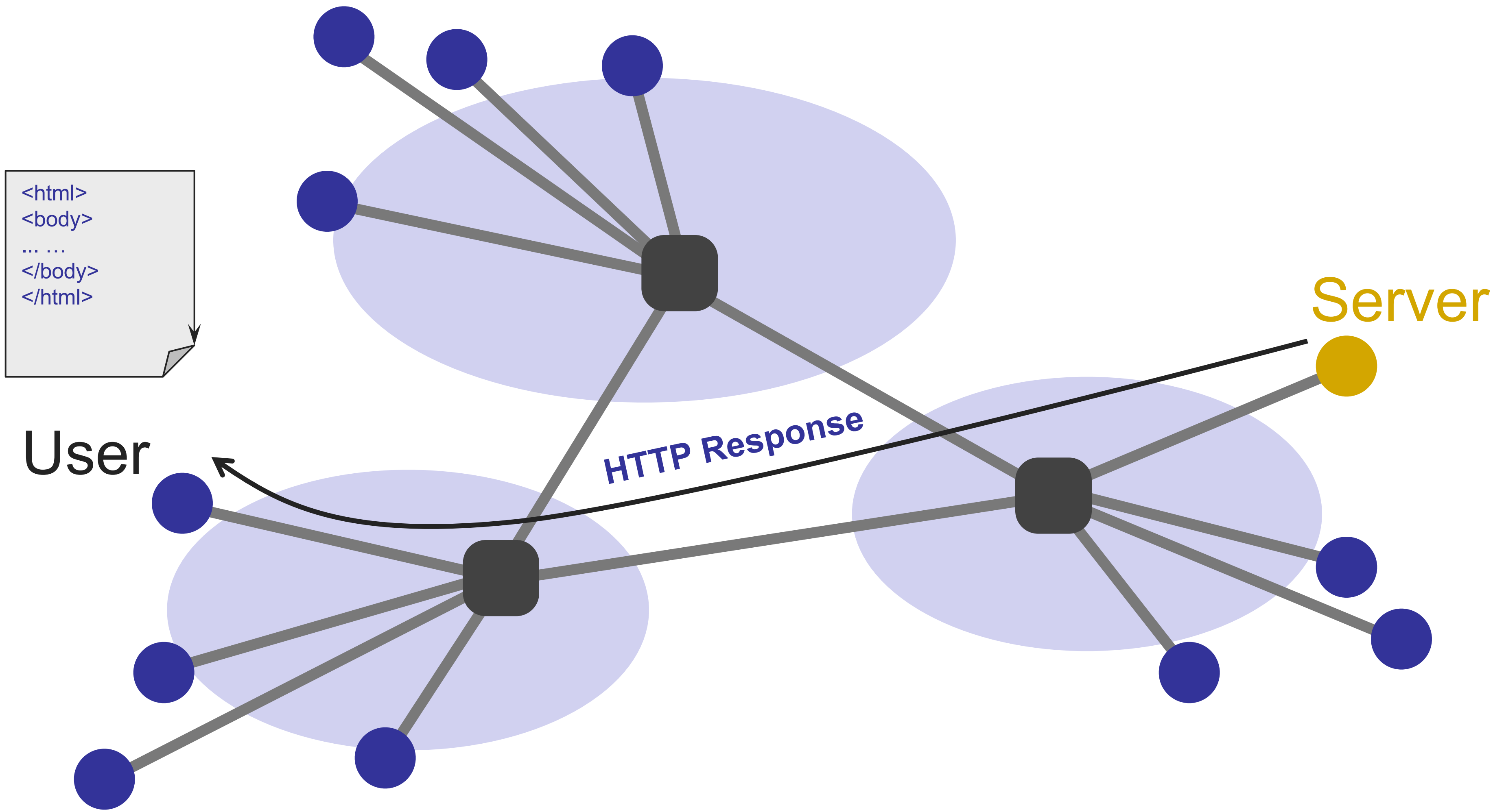
- Keep multiple resolutions of the same video
  - Stored in a manifest file in the HTTP server
- Client asks for the manifest file first to learn about the options
- Asks for chunks at a time and measures available bandwidth while they are downloaded
  - Low bandwidth ⇒ switch to lower bitrate
  - High bandwidth ⇒ switch to higher bitrate
- Adaptive bit rate (ABR)

CLOUD SYSTEMS

Who’s serving Web services?

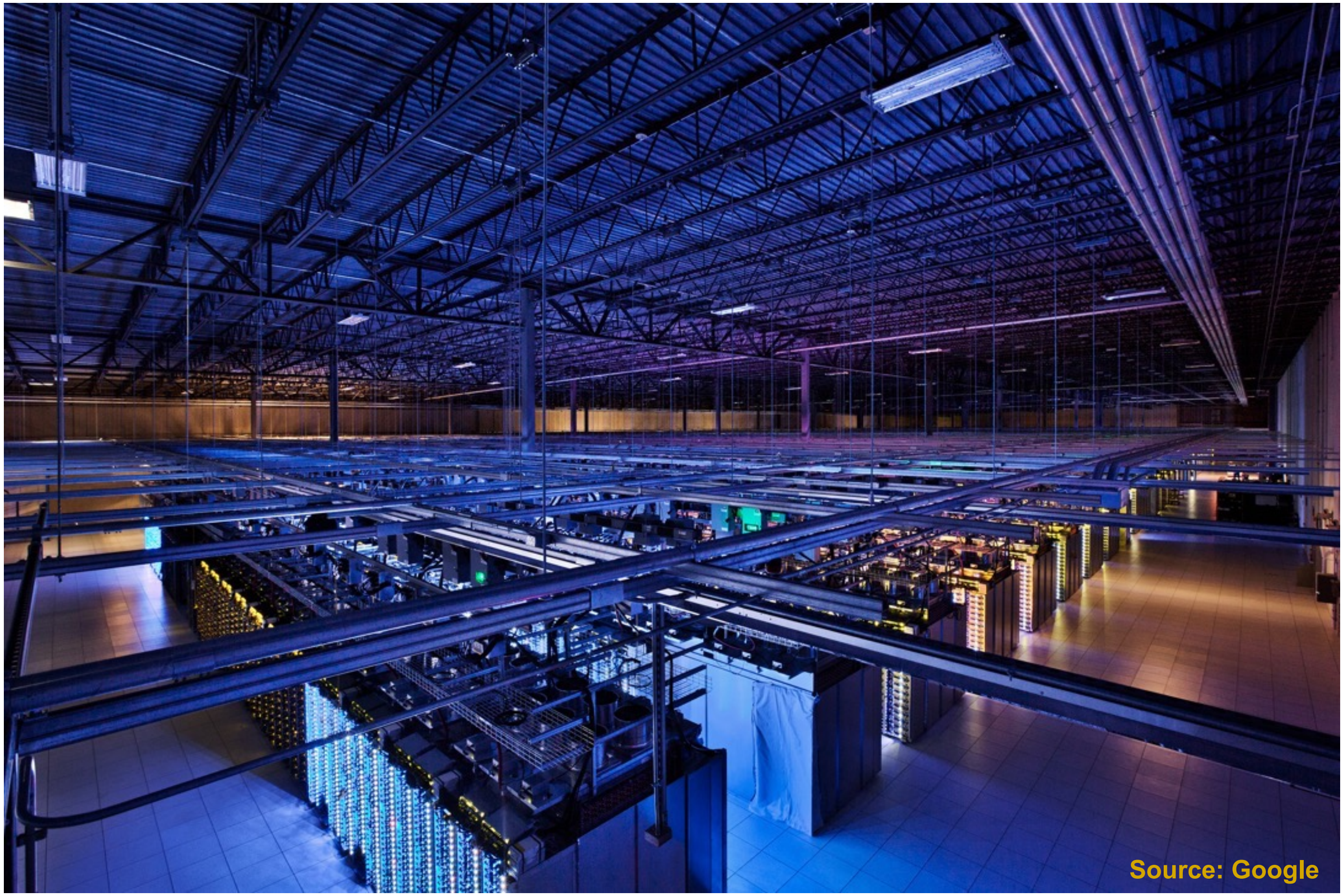


Who's serving Web services?



Cloud datacenters run the world

Cloud datacenters run the world



Cloud datacenters run the world

How big is a datacenter (DC)?

- 1M servers/site [Microsoft/Amazon/Google]
- > $1B to build one site [Facebook]
- >$20M/month/site operational costs [MS’09]
- Data center hardware spending grew to $185 billion in 2021. [Synergy Research Group report]

- But only O(100) sites

Implications (1)

- Scale
  - Need scalable designs
  - Low-cost designs: e.g., use commodity technology
  - High utilization (efficiency): e.g., >60% avg. utilization
    - » Contrast: avg. utilization on Internet links often ~30%
  - Tolerate frequent failure
    - » Large number of (low cost) components
  - Automate

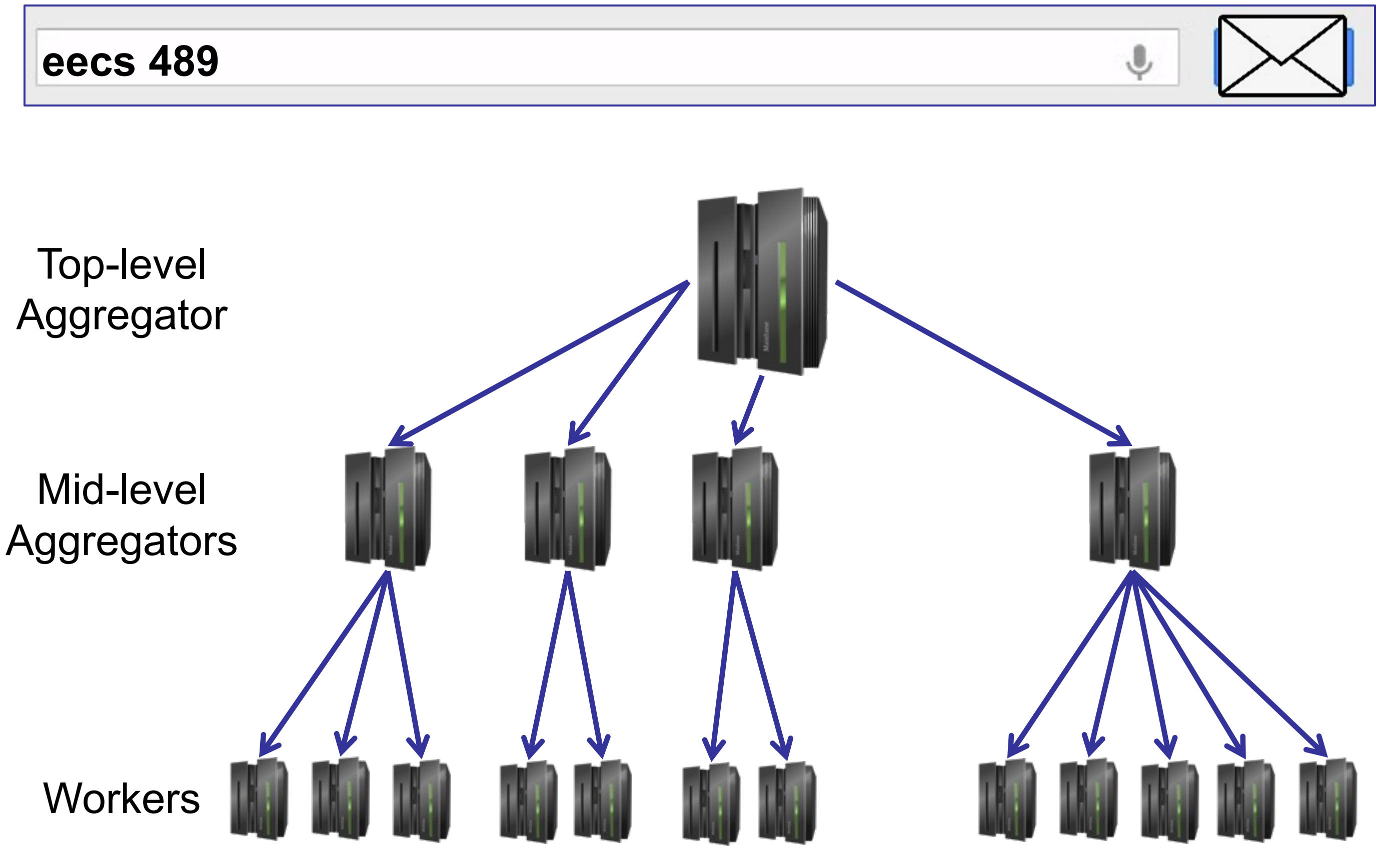
Implications (2)

- Service model: clouds / multi-tenancy
  - Performance guarantees
  - Isolation guarantees
  - Portability

Applications

- Common theme: parallelism
  - Applications decomposed into tasks
  - Running in parallel on different machines
- Two common paradigms
  - Partition-Aggregate
  - Map-Reduce

Partition-Aggregate



Partition-Aggregate

eecs 489

GitHub - mosharaf/eecs489: EECS 489: Computer Networks @ the ...
https://github.com/mosharaf/eecs489 ▾
EECS 489: Computer Networks (F'18) ... **EECS 489** takes a top-down approach to explore how networks operate and how network applications are written. ... Kurose and Ross, Computer Networking: A Top-Down Approach, 7th.

UM EECS 489: Computer Networks
www.eecs.umich.edu/courses/eecs489/ ▾
Lecture: MWF 9:30 - 10:30 in 1500 **EECS**. Discussion/Lab: W 12:30 - 1:30 in 2166 DOW or W 4:30 - 5:30 in 1014 DOW. The discussion sessions will mostly be ...

EECS 489
www.eecs.umich.edu/courses/eecs489/f99/ ▾
News group umich.**eecs**.class.**489**. Everything posted here will be automatically forwarded to the **eecs489staff@eecs**.umich.edu mailing list hourly.

EECS 489 - EECS @ Michigan - University of Michigan
https://www.eecs.umich.edu/eecs/academics/courses/eecs-489.html ▾
Course Homepage: http://www.**eecs**.umich.edu/courses/**eecs489**/w10/. Coverage We study how networks operate and how network applications are written.

Revamping EECS 489: A Retrospective | Mosharaf Chowdhury
https://www.mosharaf.com/blog/2017/05/07/revamping-eecs-489-a-retrospective/ ▾
May 7, 2017 - A couple of weeks ago, we wrapped up the Spring 2017 offering of the **EECS 489**:

End-to-end response time

- Less than 200 milliseconds between receiving user query in the browser and displaying the results
  - RTT = O(10) to 100 milliseconds
  - What remains?

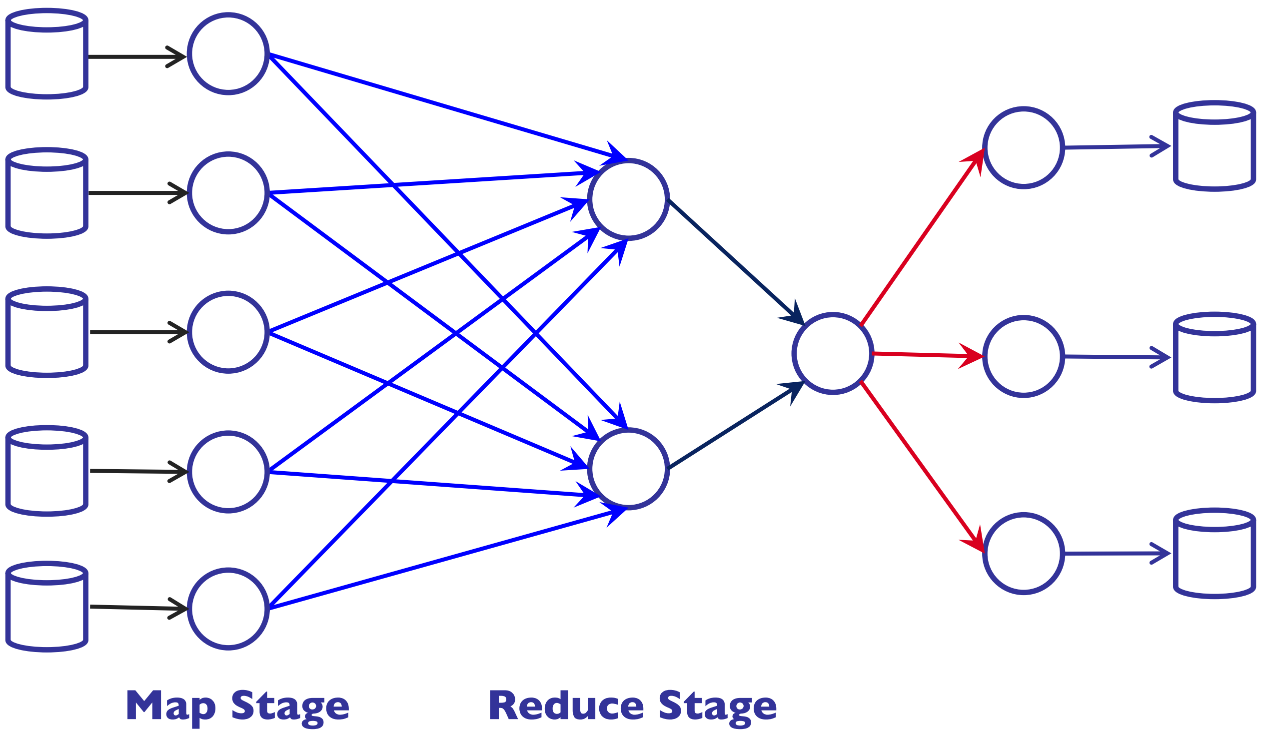
- Next time, when the page is not loading fast enough, think about the poor servers working for you ☺

5-MINUTE BREAK!

Applications

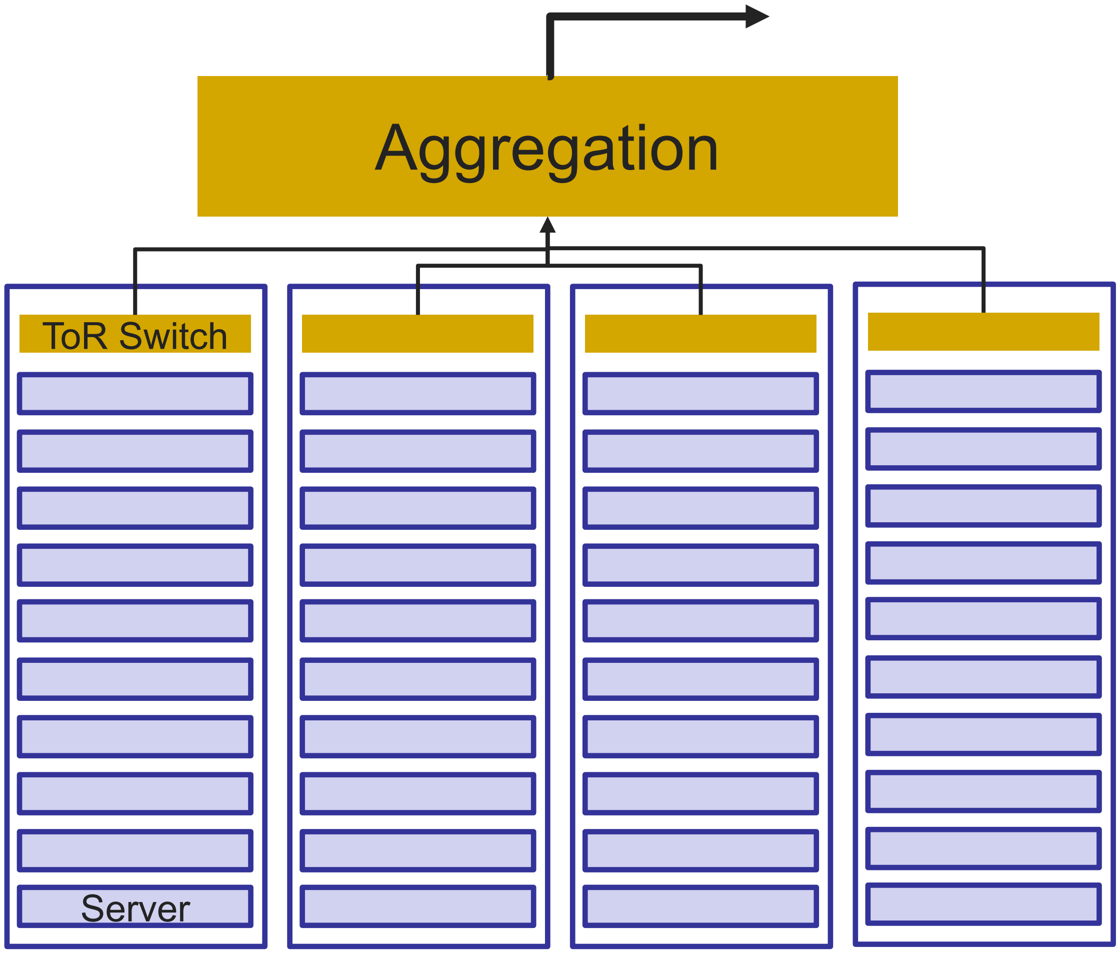
- Common theme: parallelism
  - Applications decomposed into tasks
  - Running in parallel on different machines
- Two common paradigms
  - Partition-Aggregate
  - Map-Reduce

Map-Reduce

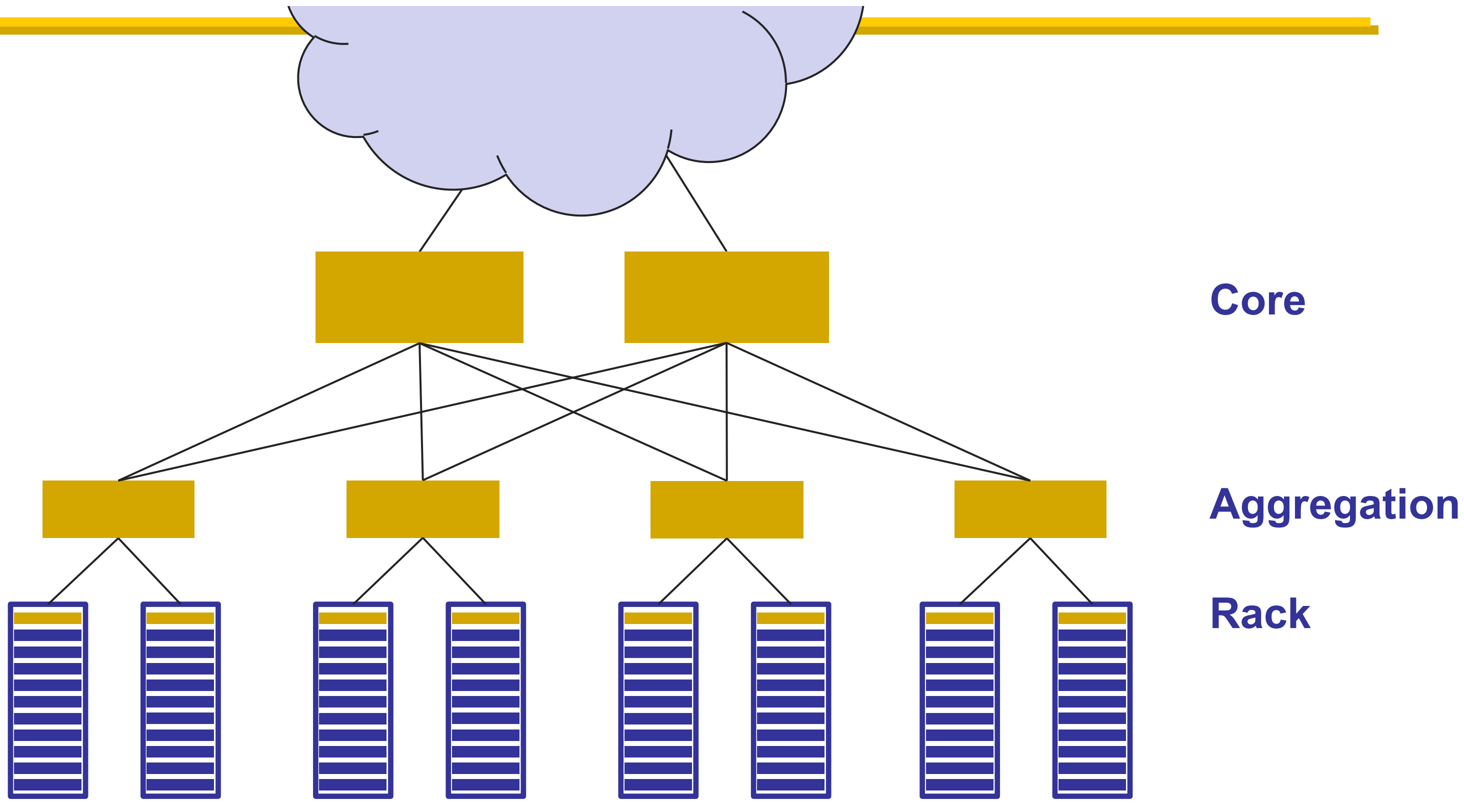


The most popular software that follows this paradigm is Apache Spark

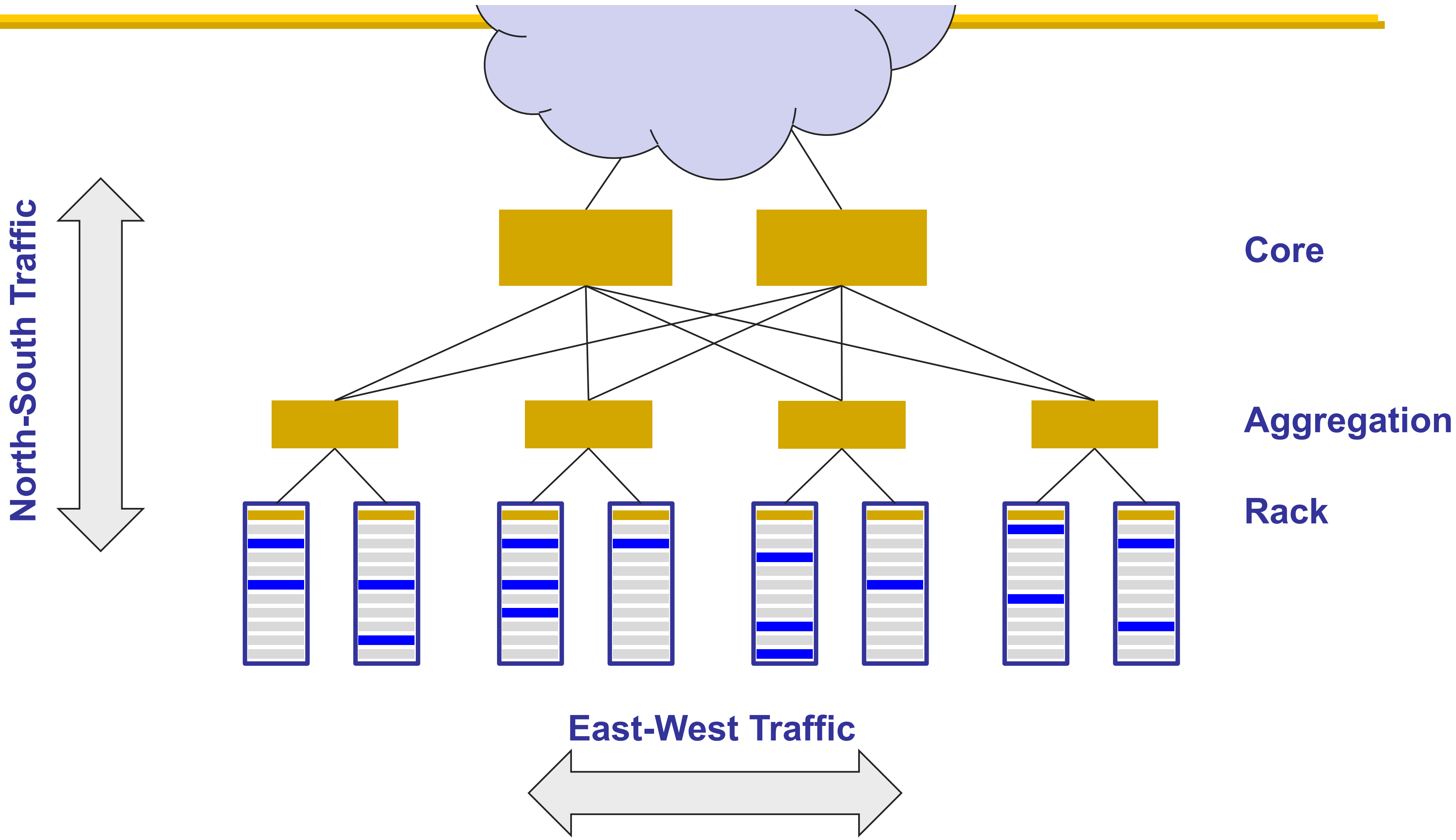
Datacenter networks



Datacenter networks (Cont.)



Datacenter traffic



East-West traffic

- Traffic between servers in the datacenter
- Communication within “big data” computations
- Traffic may shift on small timescales (< minutes)

Datacenter traffic characteristics

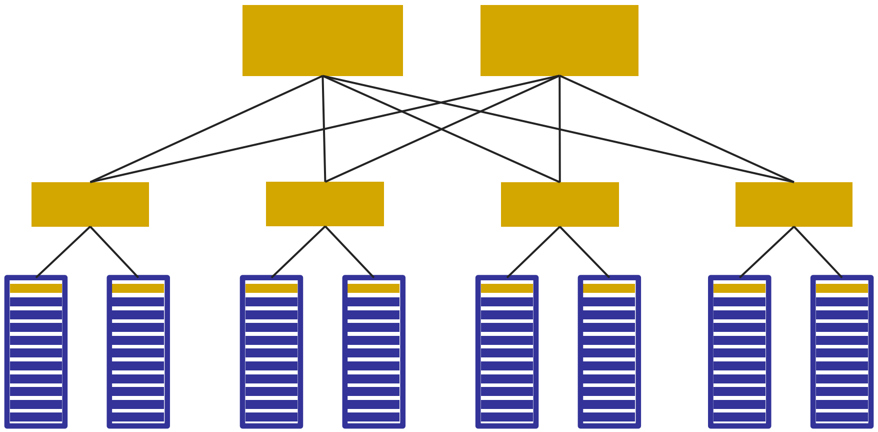
- Two key characteristics
  - Most flows are small
  - Most bytes come from large flows

- Applications want
  - High bandwidth (large flows)
  - Low latency (small flows)

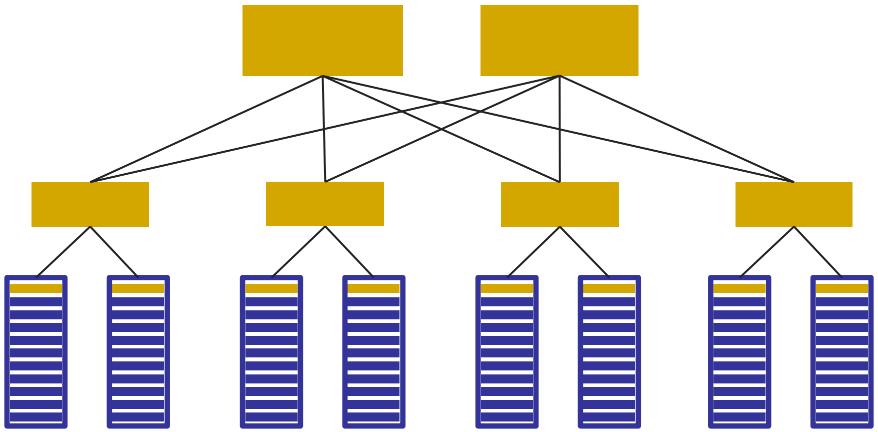
High bandwidth

- Ideal: Each server can talk to any other server at its full access link rate
- Conceptually: Datacenter network as one giant switch

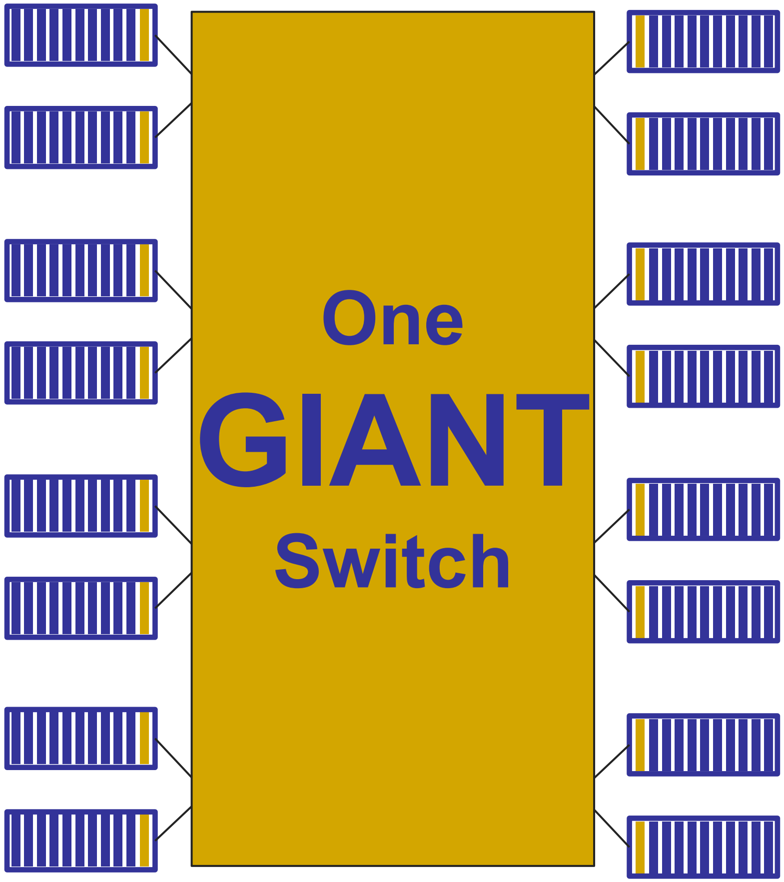
Datacenter network as one giant switch



Datacenter network as one giant switch



Datacenter network as one giant switch



High bandwidth

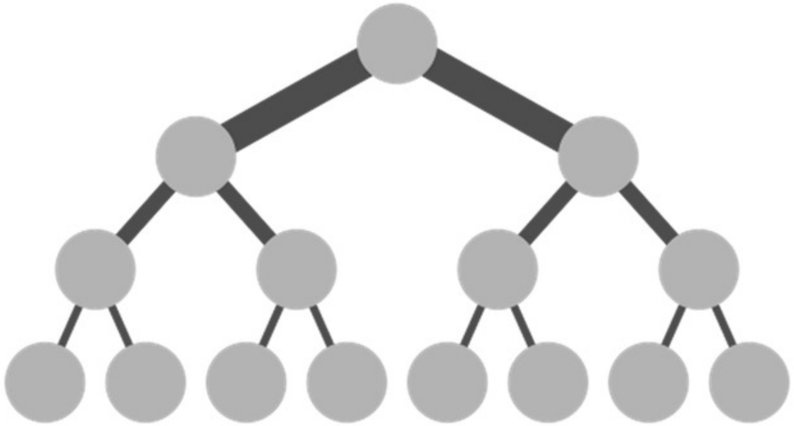
- Ideal: Each server can talk to any other server at its full access link rate
- Conceptually: Datacenter network as one giant switch
  - Would require a 10 Pbits/sec switch!
    - 1M ports (one port/server)
    - 10Gbps per port
- Practical approach: build a network of switches (“fabric”) with high “bisection bandwidth”
  - Each switch has practical #ports and link speeds

Bisection bandwidth

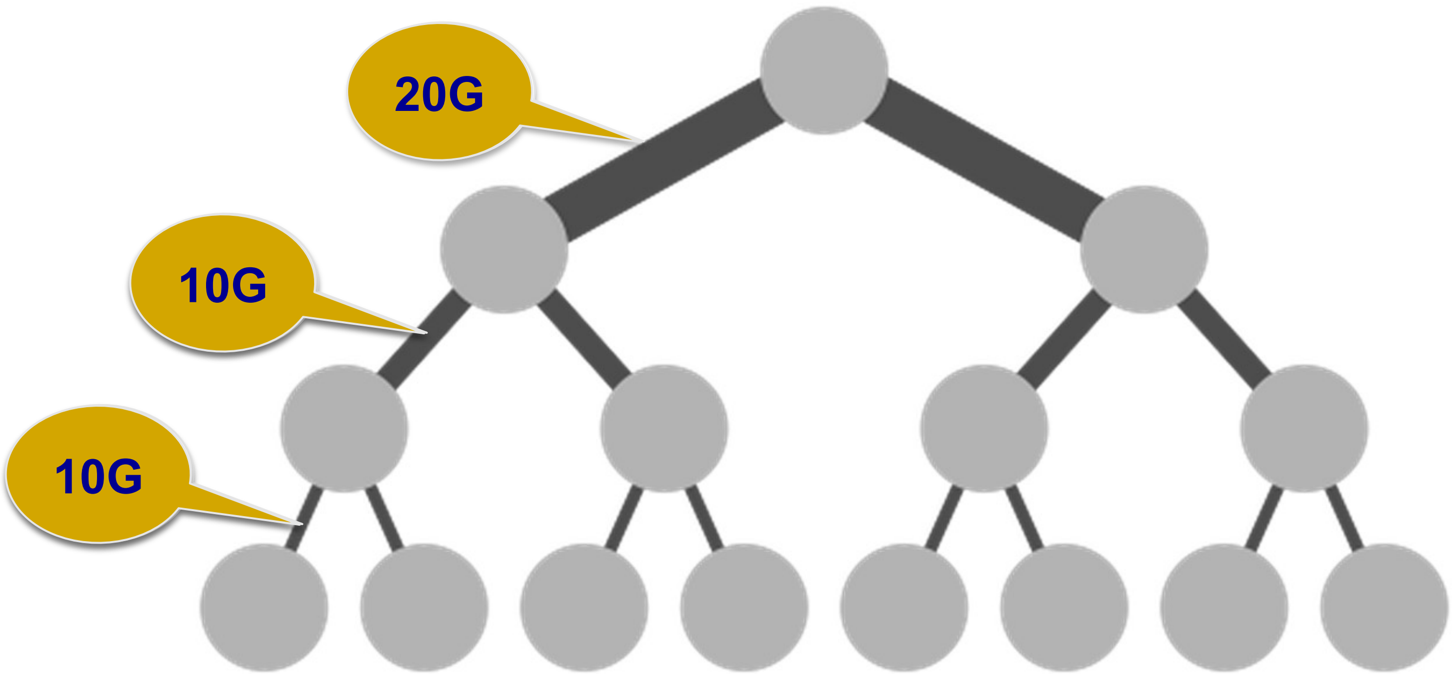
- Partition a network into two equal parts
- Minimum bandwidth between the partitions is the bisection bandwidth
- Full bisection bandwidth: bisection bandwidth in an N node network is N/2 times the bandwidth of a single link
  - Nodes of any two halves can communicate at full speed with each other

Achieving full bisection bandwidth

- Scale up
  - Make links fatter toward the core of the network
- Problem: Scaling up a traditional tree topology is expensive!
  - Requires non-commodity / impractical / link and switch components
- Solutions?
  - Over-subscribe (i.e., provision less than full BBW)
  - Better topologies



Oversubscription

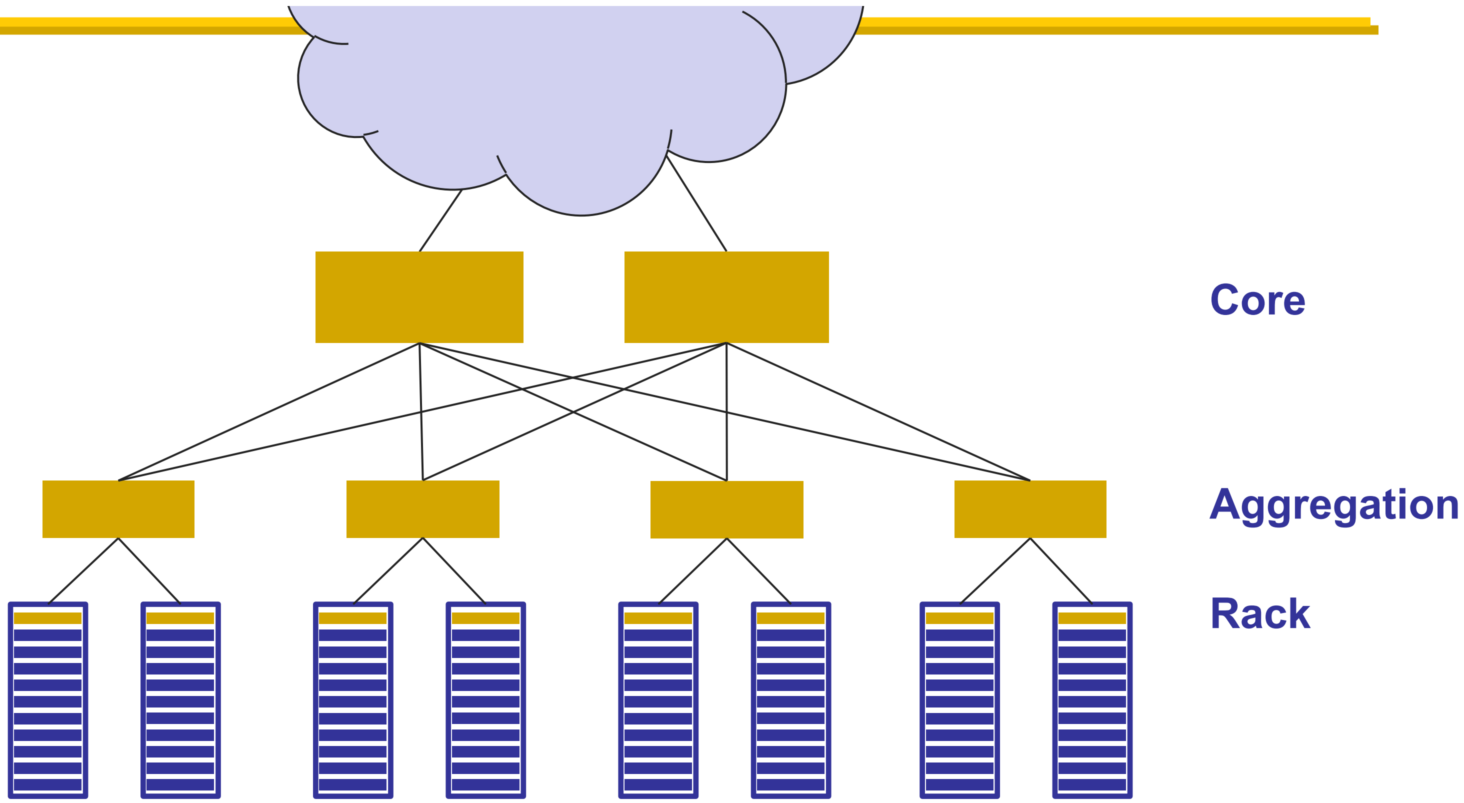


Need techniques to avoid congesting oversubscribed links!

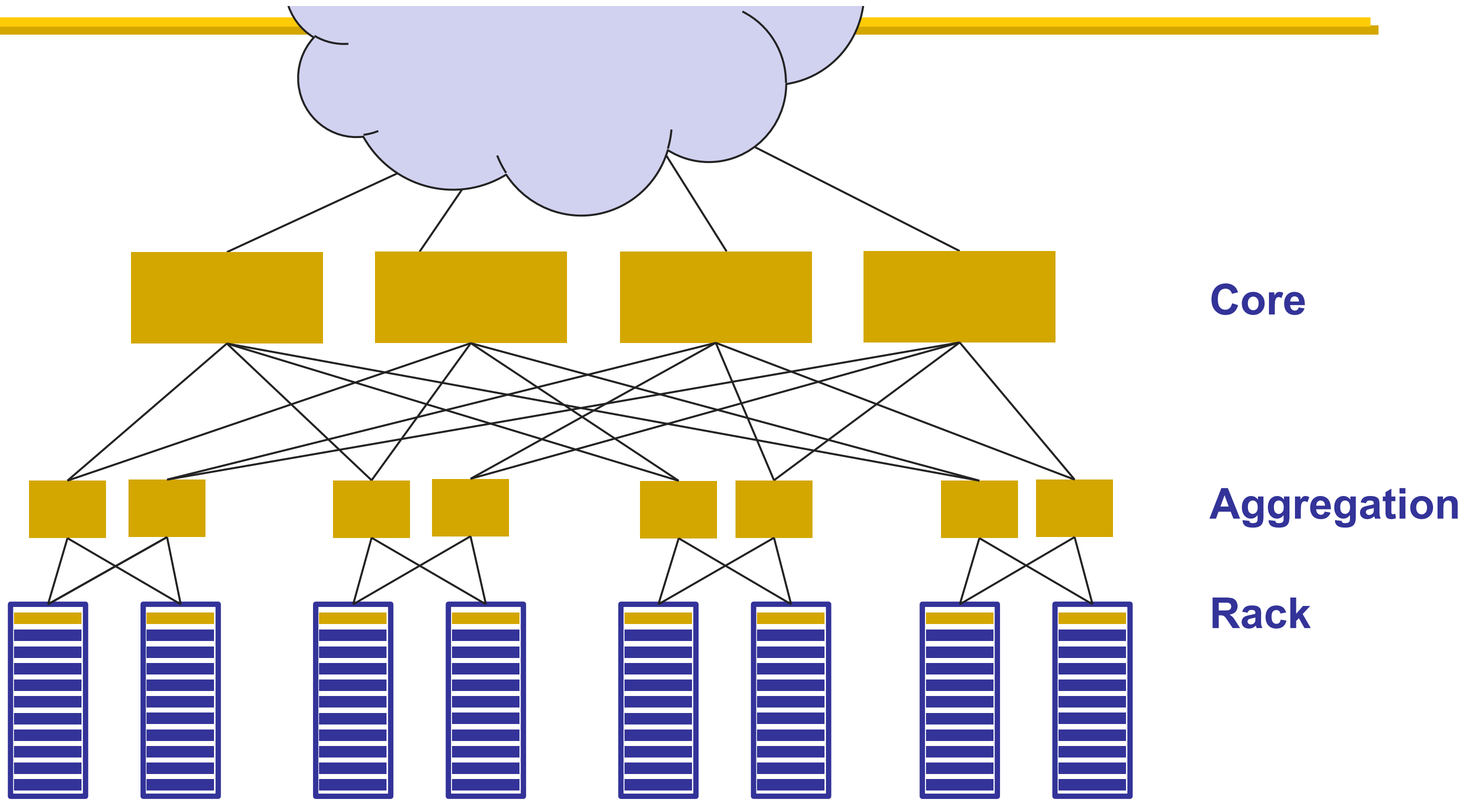
Oversubscription

- Not enough bandwidth
  - Oversubscription: Less bandwidth in the ToR-Agg links than all the servers bandwidth in the rack
  - Oversubscription ratio: Ratio between bandwidth underneath and bandwidth above
- Not enough paths between server pairs
  - Load balancing issues
  - Failure recovery issues

Better topologies



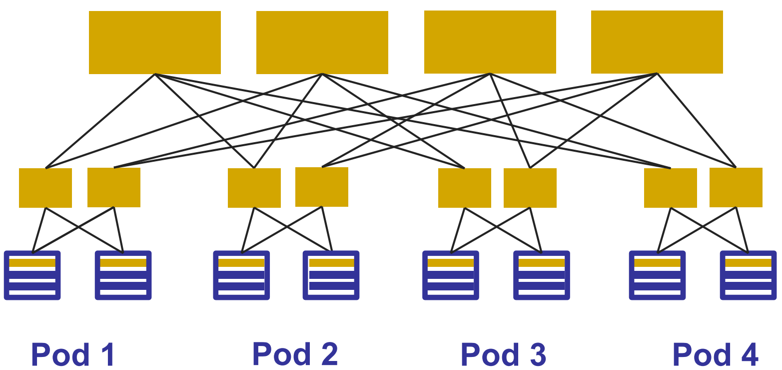
Better topologies



Clos topology

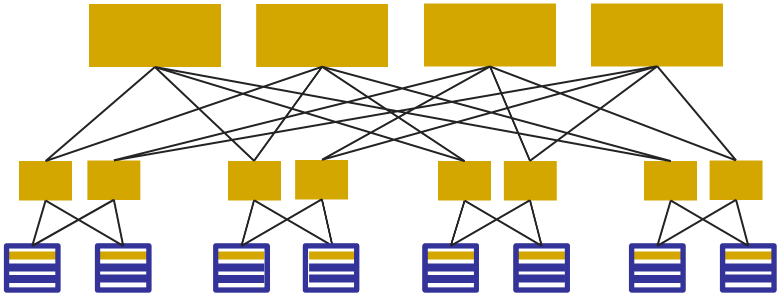
- Multi-stage network
- k pods, where each pod has two layers of k/2 switches
  - k/2 ports up and k/2 down
- All links have the same b/w
- At most $k^3/4$ machines

- Example
  - k = 4
  - 16 machines
- For k=48, 27648 machines



Challenges in scale-out designs?

- Topology offers high bisection bandwidth
- All other system components must be able to exploit this available capacity
  - Routing must use all paths
  - Transport protocol must fill all pipes (fast)



Summary

- Video streaming
  - Too large to send, so stream it
  - Dynamically adapt to the network and users
- Cloud systems
  - Forms the backend of modern web services
  - Runs in datacenters where all the processing happens