Data Center Networking

Lecture 9
Srinivas Narayana

http://www.cs.rutgers.edu/~sn624/553-S23

Parts of this lecture were heavily adapted from slides by Mohammad Alizadeh and Changhoon Kim
What are data centers?

• Large facilities with ~100K servers
  • Compute, storage, and networking working in concert
  • “Warehouse-Scale Computers”
Types of Data Centers

- Specialized data centers built for one or a few big apps
  - Social networking: Facebook, Insta
  - Web Search: Google, Bing

- “Cloud” data centers
  - Amazon EC2, Microsoft Azure
  - Google App Engine
Data Centers with 100,000+ Servers

Microsoft

Google

Facebook
Scale of mega-data centers (circa 2016)

- Each DC hosts ~100K servers
- 100s of Petabytes of storage
- 100s of Terabits/s of Bandwidth (more than core of Internet)
- 10-100MW of power (1-2% of global energy consumption)
- Cost upwards of $1--4 billion per (mega) data center
- 100s of millions of users per data center app
Datacenter traffic growth

How a data center looks on the inside
Review: Basics of data center topology

• Servers arranged into racks
  • Some people call the individual server in a rack a blade
  • Each slot also called a pizza box or 1U (“1 unit”)

• Switch interconnecting all servers at the top of the rack
  • Forms the edge of the network
  • You’ll see the name Top-Of-Rack (ToR) switch

• ToR switches are interconnected by the rest of the network fabric
Half-filled rack

• Cabling from servers to the switch

• Power, network backplane running on top
Fully filled rack

- With this kind of density, you need effective *cooling*

- Different kinds of cooling possible.
  - “Free” cooling with external air, pumped refrigerants, etc.
What’s different about DCNs?

• Single administrative domain
  • Change all endpoints and switches if you want
  • Limited interfaces with the outside world

• Unique network properties
  • Tiny round trip times (microseconds)
  • Massive multipath topologies
  • Shallow-buffered switches

• Latency and tail-latency critical
  • Tail latency: high %-ile (e.g., 99.9) of a latency distribution
  • Network is a backplane for large-scale parallel computation

• Together, serious implications for the application, transport, network, link layer designs one can use
Goal: Support cloud app requirements

• On-demand
  • Use resources when you need it; pay-as-you-go
  • Elastic: Scale up & down based on demand

• Multi-tenancy
  • Multiple independent users share infrastructure
  • Security and resource isolation
  • SLOs on performance & reliability

• Dynamic Management
  • Resiliency: isolate failure of servers and storage
  • Workload movement: move work to other locations
Example: Web Search

Partition/Aggregate App Structure

- Strict deadlines
- Tail Latency Matters
Challenges in DCNs
## Data center costs

<table>
<thead>
<tr>
<th>Amortized Cost*</th>
<th>Component</th>
<th>Sub-Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>~45%</td>
<td>Servers</td>
<td>CPU, memory, disk</td>
</tr>
<tr>
<td>~25%</td>
<td>Power infrastructure</td>
<td>UPS, cooling, power distribution</td>
</tr>
<tr>
<td>~15%</td>
<td>Power draw</td>
<td>Electrical utility costs</td>
</tr>
<tr>
<td>~15%</td>
<td>Network</td>
<td>Switches, links, transit</td>
</tr>
</tbody>
</table>

*3 yr amortization for servers, 15 yr for infrastructure, 5% cost of money

**The Cost of a Cloud: Research Problems in Data Center Networks.**

Server costs

30% server utilization considered “good” in data centers

• Application demands uneven across the resources
  • Each server has CPU, memory, disk: most applications exhaust one resource, stranding the others

• Long provisioning timescales
  • New servers purchased quarterly at best

• Uncertainty in demand
  • Demand for a new service can spike quickly

• Risk management
  • Not having spare servers to meet demand brings failure just when success is at hand
Goal: Agility: any service, any server

- Turn the servers into a single large pool
- Dynamically expand and contract service footprint as needed
- Place workloads where server resources are available
- Easier to maintain availability
  - If one rack goes down, machines from another still available

- Want to view DCN as a pool of compute connected by one big high-speed fabric
Steps to achieving Agility

• Workload (compute) management
  • Means for rapidly installing a service’s code on a server
  • Virtual machines, disk images, containers

• Storage management
  • Means for a server to access persistent data
  • Distributed global filesystems (e.g., HDFS, blob stores)

• Network and Routing management
  • Communicate efficiently with other servers, regardless of where they are in the data center
Achieving agility requires DCN to have...

- Massive **bisection bandwidth**
  - Bandwidth between any two “halves” of the network across a cut
  - Topologies, addressing, routing (Multiple paths ➔ Load balancing)

- **Ultra-Low latency** (<10 microseconds)
  - The right transport? Switch scheduling/buffer management?
  - Schedule packets or control transmission rates?
  - Centralized or distributed control?

- **Effective Resource Management** (across servers & switches)
  - Multi-tenant **performance isolation**
  - App-aware packet or flow scheduling
Conventional DC network

Source: “Data Center: Load balancing Data Center Services”, Cisco 2004
Layer 2 vs. Layer 3

• Ethernet switching (layer 2)
  ✔ Fixed IP addresses and auto-configuration (plug & play)
  ✔ Seamless mobility, migration, and failover
  x Broadcast limits scale (ARP)
  x Spanning Tree Protocol: no multipath routing

• IP routing (layer 3)
  ✔ Scalability through hierarchical addressing
  ✔ Multipath routing through equal-cost multipath
  x More complex configuration
  x Can’t migrate w/o changing IP address
Conventional DC Network Problems

- Dependence on high-cost proprietary routers
- Extremely limited server-to-server capacity
Conventional DC Network Problems

- Resource fragmentation, significantly lowering server utilization and cost-efficiency
Conventional DC Network Problems

- Resource fragmentation, significantly lowering server utilization and cost-efficiency
Building a high-speed switching fabric
Interconnecting fabric is key to agility
A single \((n \times m)\)-port switching fabric

- Different designs of switching fabric possible
- Assume \(n\) ingress ports and \(m\) egress ports, half duplex links
A single (n X m)-port switching fabric

• We are OK with any design such that:

  • Any port can connect to any other directly if all other ports free

  • **Nonblocking:** if input port x and output port y are both free, they should be able to connect
    • Regardless of other ports being connected.
    • If not satisfied, switch is **blocking.**
Nonblocking designs are nontrivial
High port density + nonblocking == hard!

- Low-cost nonblocking crossbars are feasible for small # ports

- However, it is costly to be nonblocking with a large number of ports

- If each crossover is as fast as each input port,
  - Number of crossover points == n * m
  - Cost grows quadratically on the number of input ports

- Else, crossover must transition faster than the port
  - ... so that you can keep the number of crossovers small
Nonblocking switches with many ports

• Key principle: Every fast nonblocking switch with a large number of ports is built out of many fast nonblocking switches with a small number of ports.

• How to build large nonblocking switches?
  • The subject of *interconnection networks* from the telephony era
3-stage Clos network (r*n X r*n ports)
How Clos networks become nonblocking

• if \( m > 2n - 2 \), then the Clos network is strict-sense nonblocking.

• That is, any new demand between any pair of free (input, output) ports can be satisfied without re-routing any of the existing demands.
Need at most \((n-1) + (n-1)\) middle stage

At most \(n-1\) existing demands
Surprising result about Clos networks

• if \( m \geq n \), then the Clos network is rearrangeably nonblocking

• That is, any new demand between any pair of free (input, output) ports can be satisfied by suitably re-routing existing demands.

• It is easy to see that \( m \geq n \) is necessary
  • The surprising part is that \( m \geq n \) is sufficient
Rearrangeably nonblocking Clos built with identical switches
Modern data center network topologies are just folded Clos topologies.

VL2: a scalable and flexible data center network
(sigcomm’09)
How does one design a Clos DCN?

• Switches are usually n X n with full-duplex links

• **Fold** the 3-stage Clos into 2-stages

• Share physical resources between ingress and egress stages

• Share ports and links across the two “sides” of the middle stage
From/to hosts

ToR/aggregation layer

n/2 X n/2

n/2 X n/2

n/2 X n/2

n X n

n X n

n X n

n/2 X n/2

n/2 X n/2

n/2 X n/2

ToR/aggregation layer

Spine layer

n X n

n/2 switches n X n full duplex

n X n full duplex

n X n full duplex

n X n full duplex
Consequences of using folded Clos

- 2-stage high throughput data center topology
- All can use the same switches! (port density and link rates)
What about routing?

• We said that the Clos topology above is rearrangeably nonblocking.

• So, how to rearrange existing demands when a new packet arrives, so that it can get across as quickly as possible?

• How to do it without “interference” to (ie: rerouting) other pkts?

• VL2: We don’t need to rearrange anything.
Valiant Load Balancing (VLB)

- Designed to move data quickly for shuffling in parallel computing

- Setting: Connectivity is sparse ("hypercube" topology): log n links per node in a network with n nodes

- Key idea: pick a random node to redirect a message to from the source, then follow the shortest path to the destination from there

- Guarantee: With high probability, the message reaches its destination very quickly (log n steps)
  - Practically, this means there is very less queueing in the network
VLB in data center networks

• VLB is more general than data center networks or Clos
  • It is a form of oblivious routing
  • e.g., no need to measure traffic patterns before choosing routes
  • Extremely simple to implement: no global state

• VLB is handy in folded Clos topologies due to the numerous options to pick the first-hop from ToR switch
  • Balance load across many paths

• Very beneficial in practice
  • Performance isolation: other flows don’t matter
  • High capacity (“bisection bandwidth”) between two ToR ports
VLB requirements: Hose model

• The guarantees of VLB + Clos only hold under the hose model:
  • Demands for any one ToR port (send or receive) must not exceed its bandwidth.

• Very hard to enforce especially on the receiver side without sender-side rate limits.

• VL2 uses TCP convergence as a way of ensuring that aggregate ToR port demand is within its bandwidth
Requirements for VLB to work well

- VLB + Clos provides high capacity if no ToR port demands more than its bandwidth (hose model)

A bunch of results arrive at MLAs and TLAs in a short period.

Demand may exceed ToR port bandwidth!
Enforcing the hose model

• Hose model is hard to enforce especially on the receiver side without sender-side rate limits

• VL2 uses TCP’s convergence to bottleneck link rate as a loose method to enforce that the demand for port bandwidth is met by the available capacity.