Data Centers
Part II

Lecture 8, Computer Networks (198:552)
Fall 2019
What are data centers?

• Large facilities with 10s of thousands of networked servers
  • Compute, storage, and networking working in concert
  • “Warehouse-Scale Computers”
Cloud Computing: The View from Apps

• 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
  • SLAs on performance & reliability

• Dynamic Management
  • Resiliency: isolate failure of servers and storage
  • Workload movement: move work to other locations
What’s different about DCNs?

• Single administrative domain
  • Change all endpoints and switches if you want
  • No need to be compatible with outside world

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

• Latency and tail-latency critical
  • Network is a backplane for large-scale parallel computation

• Together, serious implications for the transport, network, link layer designs you can (and should) use
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

---

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

- Session state and storage constraints
  - If the world were stateless servers, life would be good
Goal: *Agility* -- any service, any server

- Turn the servers into a *single large fungible 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 filesystems (e.g., HDFS, blob stores)

• Network and Routing management
  • Means for communicating with other servers, regardless of where they are in the data center
Agility means that the DCN needs...

- Massive *bisection bandwidth*
  - Topologies
  - 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 network scheduling (e.g., for big data)
- Support for next-generation hardware & apps
  - ML, RDMA, rack-scale computing, memory disaggregation
Conventional DC network

### Key
- **CR** = Core Router (L3)
- **AR** = Access Router (L3)
- **S** = Ethernet Switch (L2)
- **A** = Rack of app. servers

~ 1,000 servers/pod == IP subnet

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 cloud utilization (and cost-efficiency)
Conventional DC Network Problems

- Resource fragmentation, significantly lowering cloud utilization (and cost-efficiency)
## Google’s DCN Challenges & Approaches

<table>
<thead>
<tr>
<th>Challenge</th>
<th>Our Approach (Section Discussed in)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introducing the network to production</td>
<td>Initially deploy as bag-on-the-side with a fail-safe big-red button (3.2)</td>
</tr>
<tr>
<td>High availability from cheaper components</td>
<td>Redundancy in fabric, diversity in deployment, robust software, necessary protocols only, reliable out of band control plane (3.2, 3.3, 5.1)</td>
</tr>
<tr>
<td>High fiber count for deployment</td>
<td>Cable bundling to optimize and expedite deployment (3.3)</td>
</tr>
<tr>
<td>Individual racks can leverage full uplink capacity to external clusters</td>
<td>Introduce Cluster Border Routers to aggregate external bandwidth shared by all server racks (4.1)</td>
</tr>
<tr>
<td>Incremental deployment</td>
<td>Depopulate switches and optics (3.3)</td>
</tr>
<tr>
<td>Routing scalability</td>
<td>Scalable in-house IGP, centralized topology view and route control (5.2)</td>
</tr>
<tr>
<td>Interoperate with external vendor gear</td>
<td>Use standard BGP between Cluster Border Routers and vendor gear (5.2.5)</td>
</tr>
<tr>
<td>Small on-chip buffers</td>
<td>Congestion window bounding on servers, ECN, dynamic buffer sharing of chip buffers, QoS (6.1)</td>
</tr>
<tr>
<td>Routing with massive multipath</td>
<td>Granular control over ECMP tables with proprietary IGP (5.1)</td>
</tr>
<tr>
<td>Operating at scale</td>
<td>Leverage existing server installation, monitoring software; tools build and operate fabric as a whole; move beyond individual chassis-centric network view; single cluster-wide configuration (5.3)</td>
</tr>
<tr>
<td>Inter cluster networking</td>
<td>Portable software, modular hardware in other applications in the network hierarchy (4.2)</td>
</tr>
</tbody>
</table>

Source: Jupiter rising: a decade of Clos topologies and centralized control in Google’s data center network. SIGCOMM’15
Building a high-speed switching fabric
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 \((n \times m)\)-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*.

Electrical/mechanical/electronic crossover

Crossbar
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

• Q: How is this relevant to the data center network fabric?
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.

- How?
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.
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
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
  • No need to measure DCN traffic patterns before deciding on routing
  • Extremely simple to implement: no global state

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

• Very beneficial in practice
  • Performance isolation: other flows don’t matter
  • High capacity (“bisection bandwidth”) between any two ToR ports
Is ToR-port to ToR-port high capacity enough?

• VLB + Clos provides high capacity if no ToR port demands more than its bandwidth.

• But is that enough?
Example: Web Search

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

Demand may exceed ToR port bandwidth!
Hose traffic 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.  
  • Q: do you see any problems?
VL2: Virtual Layer 2
Backup slides
VL2 goals
VL2 design principles

• Randomizing to cope with volatility
  • Tremendous variability in traffic matrices

• Separating names from locations
  • Any server, any service

• Embracing end systems
  • Leverage the programmability & resources of servers
  • Avoid changes to switches

• Building on proven networking technology
  • Build with parts shipping today
  • Leverage low cost, powerful merchant silicon ASICs, though do not rely on any one vendor
Single-chip “merchant silicon” switches

Image courtesy of Facebook
### Specific objectives and solutions

<table>
<thead>
<tr>
<th>Objective</th>
<th>Approach</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Layer-2 semantics</td>
<td>Employ flat addressing</td>
<td>Name-location separation &amp; resolution service</td>
</tr>
<tr>
<td>2. Uniform high capacity between servers</td>
<td>Guarantee bandwidth for hose-model traffic</td>
<td>Flow-based random traffic indirection (Valiant LB)</td>
</tr>
<tr>
<td>3. Performance Isolation</td>
<td>Enforce hose model using existing mechanisms only</td>
<td>TCP</td>
</tr>
</tbody>
</table>
Addressing and routing: Name-location separation

Cope with host churns with very little overhead

Servers use flat names

Switches run link-state routing and maintain only switch-level topology
Addressing and routing: Name-location separation

Cope with host churns with very little overhead

**VL2**

Switches run link-state routing and maintain only switch-level topology

- Allows to use low-cost switches
- Protects network and hosts from host-state churn
- Obviates host and switch reconfiguration

Servers use flat names
Example topology: Clos network

Offer huge aggr capacity and multi paths at modest cost

VL2

Aggr

Int

TOR

\( K \) aggr switches with \( D \) ports

\( 20\text{-}(DK/4) \) Servers

\( 20\text{-} \) Servers
Example topology: Clos network

Offer huge aggr capacity and multi paths at modest cost

<table>
<thead>
<tr>
<th>D (# of 10G ports)</th>
<th>Max DC size (# of Servers)</th>
</tr>
</thead>
<tbody>
<tr>
<td>48</td>
<td>11,520</td>
</tr>
<tr>
<td>96</td>
<td>46,080</td>
</tr>
<tr>
<td>144</td>
<td>103,680</td>
</tr>
</tbody>
</table>

Example: Clos network
Traffic forwarding: Random indirection

Cope with arbitrary TMs with very little overhead

Links used for up paths
Links used for down paths

\[
\begin{align*}
I_{\text{ANY}} & \quad I_{\text{ANY}} & \quad I_{\text{ANY}} \\
T_1 & \quad T_2 & \quad T_3 & \quad T_4 & \quad T_5 & \quad T_6 \\
I_{\text{ANY}} & \quad T_5 & \quad z & \quad \text{payload} \\
x & \quad y & \quad z
\end{align*}
\]
Traffic forwarding: Random indirection

Cope with arbitrary TMs with very little overhead

[ ECMP + IP Anycast ]

- Harness huge bisection bandwidth
- Obviate esoteric traffic engineering or optimization
- Ensure robustness to failures
- Work with switch mechanisms available today

Links used for up paths _any_
Links used for down paths _any_
Some other DC network designs…

BCube [SIGCOMM’10]

Jellyfish (random) [NSDI’12]