Computer Networks

Datacenter Networks

Material based on courses at Princeton, MIT
What are Data Centers?

Large facilities with 10s of thousands of networked servers
- Compute, storage, and networking working in concert
- “Warehouse-Scale Computers”
- Huge investment: ~ 0.5 billion for large datacenter
## 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% utilization considered “good” in most data centers!

Uneven application fit
  – Each server has CPU, memory, disk: most applications exhaust one resource, stranding the others

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 fungible pool
  – Dynamically expand and contract service footprint as needed

Benefits
  – Lower cost (higher utilization)
  – Increase developer productivity
  – Achieve high performance and reliability
Achieving Agility

Workload 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
– Means for communicating with other servers, regardless of where they are in the data center
Datacenter Networks

Provide the illusion of “One Big Switch”

10,000s of ports

Compute

Storage (Disk, Flash, ...)

Raw Text Only:

Datacenter Networks

Provide the illusion of “One Big Switch”

10,000s of ports

Compute

Storage (Disk, Flash, ...)

RAW TEXT END
Datacenter Traffic Growth

Today: Petabits/s in one DC
- More than core of the Internet!

Conventional DC Network Problems
Conventional DC Network

Reference – “Data Center: Load balancing Data Center Services”, Cisco 2004
Conventional DC Network Problems

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

- Dependence on high-cost proprietary routers
- Extremely limited server-to-server capacity
- Resource fragmentation
And More Problems ...

Poor reliability
Lack of performance isolation
VL2 Paper

Measurements

VL2 Design
- Clos topology
- Valiant LB
- Name/location separation
  (precursor to network virtualization)

Measurements
DC Traffic Characteristics

Instrumented a large cluster used for data mining and identified distinctive traffic patterns

Traffic patterns are **highly volatile**
  - A large number of distinctive patterns even in a day

Traffic patterns are **unpredictable**
  - Correlation between patterns very weak

Traffic-aware optimization needs to be done frequently and rapidly
DC Opportunities

DC controller knows **everything** about **hosts**

Host OS’s are easily **customizable**

**Probabilistic** flow distribution would work well enough, because ...

- Flows are numerous and not huge – few elephants
- Commodity switch-to-switch links are substantially thicker (~10x) than the maximum thickness of a flow

DC network can be made simple
Intuition

Higher speed links improve *flow-level* load balancing (ECMP)

- **20×10Gbps Uplinks**
  - 11×10Gbps flows (55% load)

- **2×100Gbps Uplinks**

**Prob of 100% throughput** = 3.27%

**Prob of 100% throughput** = 99.95%
Virtual Layer 2
VL2 Goals

The Illusion of a Huge L2 Switch

1. L2 semantics
2. Uniform high capacity
3. Performance isolation
Clos Topology

Offer huge capacity via multiple paths (scale out)
Building Block: Merchant Silicon Switching Chips

Switch ASIC

6 pack

Facebook Wedge

Image courtesy of Facebook
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
## VL2 Goals 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

**VL2**
Switches run link-state routing and maintain only switch-level topology.

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

Servers use flat names

**Directory Service**

**Lookup & Response**
Figures V.8.1: VL2 Agent in Action

Why use hash for Src IP? Why anycast & double encapsulation?
Other details

How does L2 broadcast work?

How does Internet communication work?
VL2 Directory System

Read-optimized Directory Servers for lookups

Write-optimized Replicated State Machines for updates

Stale mappings?
Data Center Congestion Control
100Kbps–100Mbps links
~100ms latency

10–40Gbps links
~10–100μs latency

Transport inside the DC
Interconnect for distributed compute workloads
What’s Different About DC Transport?

Network characteristics
  – Very high link speeds (Gb/s); very low latency (microseconds)

Application characteristics
  – Large-scale distributed computation

Challenging traffic patterns
  – Diverse mix of mice & elephants
  – Incast

Cheap switches
  – Single-chip shared-memory devices; shallow buffers
Data Center Workloads

Mice & Elephants

Short messages
(e.g., query, coordination) → Low Latency

Large flows
(e.g., data update, backup) → High Throughput
Incast

- Synchronized fan-in congestion

Worker 1
Worker 2
Worker 3
Worker 4

Aggregator

\[ \text{RTO}_{\text{min}} = 300 \text{ ms} \]

TCP timeout

\[ \text{Vasudevan et al. (SIGCOMM'09)} \]
Incast in Bing

Jittering trades of median for high percentiles
Requests are jittered over 10ms window.

Jittering switched off around 8:30 am.

Incast in Bing

Jittering trades of median for high percentiles
Baseline fabric latency (propagation + switching): **10 microseconds**
High throughput requires buffering for rate mismatches
... but this adds significant queuing latency
Data Center TCP
TCP in the Data Center

TCP [Jacobsen et al.’88] is widely used in the data center
  – More than 99% of the traffic

Operators work around TCP problems
  – Ad-hoc, inefficient, often expensive solutions
  – TCP is deeply ingrained in applications

Practical deployment is hard
  → keep it simple!
Review: The TCP Algorithm

Additive Increase:
\[ W \rightarrow W+1 \text{ per round-trip time} \]

Multiplicative Decrease:
\[ W \rightarrow W/2 \text{ per drop or ECN mark} \]

ECN = Explicit Congestion Notification
TCP Buffer Requirement

Bandwidth-delay product rule of thumb:

− A single flow needs $C \times RTT$ buffers for 100% Throughput.
Reducing Buffer Requirements

Appenzeller et al. (SIGCOMM ‘04):
- Large # of flows: $C \times \frac{RTT}{\sqrt{N}}$ is enough.
Reducing Buffer Requirements

Appenzeller et al. (SIGCOMM ’04):
  – Large # of flows: $C \times \frac{RTT}{\sqrt{N}}$ is enough

Can’t rely on stat-mux benefit in the DC.
  – Measurements show typically only 1-2 large flows at each server

Key Observation:
Low variance in sending rate $\Rightarrow$ Small buffers suffice
DCTCP: Main Idea

- Extract multi-bit feedback from single-bit stream of ECN marks
  - Reduce window size based on \textit{fraction} of marked packets.

<table>
<thead>
<tr>
<th>ECN Marks</th>
<th>TCP</th>
<th>DCTCP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0 1 1 1 0 1 1 1</td>
<td>Cut window by 50%</td>
<td>Cut window by 40%</td>
</tr>
<tr>
<td>0 0 0 0 0 0 0 0 1</td>
<td>Cut window by 50%</td>
<td>Cut window by 5%</td>
</tr>
</tbody>
</table>

![Graphs showing TCP and DCTCP window size variations over time](image-url)
DCTCP: Algorithm

Switch side:
- Mark packets when Queue Length > K.

Sender side:
- Maintain running average of fraction of packets marked ($\alpha$).
  
  each RTT:  
  
  \[ F = \frac{\# \text{ of marked ACKs}}{\text{Total \# of ACKs}} \quad \Rightarrow \quad \alpha \leftarrow (1 - g)\alpha + gF \]

  
  \[ W \leftarrow (1 - \frac{\alpha}{2})W \]

- Note: decrease factor between 1 and 2.
**DCTCP vs TCP**

**Experiment:** 2 flows (Win 7 stack), Broadcom 1Gbps Switch

DCTCP mitigates Incast by creating a large buffer headroom

Buffer is mostly empty
Why it Works

1. Low Latency
   ✓ Small buffer occupancies → low queuing delay

2. High Throughput
   ✓ ECN averaging → smooth rate adjustments, low variance

3. High Burst Tolerance
   ✓ Large buffer headroom → bursts fit
   ✓ Aggressive marking → sources react before packets are dropped
Bing Benchmark (baseline)

Background Flows

Query Flows

Flow Completion Time (ms)

Flow Size

DCTCP

TCP

Query Completion Time (ms)

Mean 95th 99th 99.9th

DCTCP

TCP
Bing Benchmark (scaled 10x)

Incast
Deep buffers fix incast, but increase latency
DCTCP good for both incast & latency

Completion Time (ms)

Query Traffic (Incast bursts)
Short messages (Delay-sensitive)

TCP/ShallowBuf
TCP/DeepBuf
DCTCP/ShallowBuf