A Scalable, Commodity Data Center Network Architecture


Presenter: William Beyer

Paper Goals

• Point out faults with current data center designs
• Propose new architecture based on fat-tree
  – Scalable interconnection bandwidth
  – Economies of scale
  – Backward compatibility

A Typical Data Center

• Data center topology is typically 2-3 level tree of switches and routers

Oversubscription

• Ratio of worst-case achievable aggregate bandwidth among end-hosts to the total bisection bandwidth of the network topology
  – Ability of hosts to fully utilize their uplink capacity
• 1:1 – All hosts can use full uplink capacity
• 5:1 – Only 20% of host bandwidth may be available
• Typical ratio is 2.5:1 (400 Mbps) to 8:1 (125 Mbps)

Figure 1: Common data center interconnect topology. Host to switch links are GigE and links between switches are 10 GigE.
Multi-path Routing

- “Multi-rooted” tree required to communicate at full bandwidth for large clusters
  - Otherwise limited to max bandwidth of a single expensive switch (128-port 10 GigE)
- Use multi-path routing technique such as ECMP
  - Performs static load splitting, cannot account for flow sizes
  - Routing tables become very large with multiple paths

Cost Analysis

<table>
<thead>
<tr>
<th>Year</th>
<th>Hierarchical design</th>
<th>Fat-tree</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10 GigE</td>
<td>Hosts</td>
</tr>
<tr>
<td>2002</td>
<td>28-port 4,480</td>
<td>$25.3K</td>
</tr>
<tr>
<td>2004</td>
<td>32-port 7,680</td>
<td>$4.4K</td>
</tr>
<tr>
<td>2006</td>
<td>64-port 10,240</td>
<td>$2.1K</td>
</tr>
<tr>
<td>2008</td>
<td>128-port 20,480</td>
<td>$1.8K</td>
</tr>
</tbody>
</table>

Table 1: The maximum possible cluster size with an oversubscription ratio of 1:1 for different years.

Fat-tree Architecture

- k-ary fat-tree: three-layer topology (edge, aggregation, core)
  - k pods, each consists of (k/2)^2 hosts and two layers (edge/aggregate) each with k/2 k-port switches
  - Each edge switch connects to k/2 hosts and k/2 aggregate switches
  - Each aggregate switch connects to k/2 edge and k/2 core switches
  - (k/2)^2 core switches: each connects to k pods
  - Supports k^3/4 hosts!
Fat-tree Topology with $k = 4$

Issues with Fat-tree Topologies

- Backwards compatible with IP/Ethernet
  - Good thing, but routing algorithms will naively choose a single shortest path to use between subnets
  - Leads to bottlenecks quickly
  - $(k/2)^2$ shortest paths available, should use them all equally
- Complex wiring due to lack of high speed ports

Addressing in Fat-tree

- Use 10.0.0.0/8 private addressing block
- Pod switches have address 10.pod.switch.1
  - Pod and switch in $[0, k-1]$ based on position
- Core switches have address 10.k.j.i
  - $i$ and $j$ denote core position in $(k/2)^2$ core switches
- Hosts have address 10.pod.switch.ID
  - ID is host ID in switch subnet ($[2, (k/2) + 1]$)
  - $k < 256$, this scheme does not scale indefinitely

Two-Level Lookup Table

- Prefixes used for forwarding intra-pod traffic
- Suffixes used for forwarding inter-pod traffic

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Output port</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.2.0.0/24</td>
<td>0</td>
</tr>
<tr>
<td>10.2.1.0/24</td>
<td>1</td>
</tr>
<tr>
<td>0.0.0.0/0</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Suffix</th>
<th>Output port</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0.0.2/8</td>
<td>2</td>
</tr>
<tr>
<td>0.0.0.3/8</td>
<td>3</td>
</tr>
</tbody>
</table>

Figure 3: Simple fat-tree topology. Using the two-level routing tables described in Section 3.3, packets from source 10.0.1.2 to destination 10.2.0.3 would take the dashed path.

Figure 4: Two-level table example. This is the table at switch 10.2.2.1. An incoming packet with destination IP address 10.2.1.2 is forwarded on port 1, whereas a packet with destination IP address 10.3.0.3 is forwarded on port 3.
Two-Level Lookup Implementation

• Implemented in hardware using a TCAM
  – Can perform parallel lookups across table
  – Stores don’t care bits, suitable for storing variable length prefixes

• Prefixes preferred over suffixes

Routing Algorithm

• Prefixes in two-level table prevent intra-pod traffic from leaving pod

• Inter-pod traffic handled by suffix table
  – Suffixes based off host IDs, ensures spread of traffic across core switches
  – Prevents packet reordering by having static path

• Each host-to-host communication has a single static path
  – Better than having a single path between subnets

Routing Algorithm (cont.)

• Core switches contain (10.pod.0.0/16, port) entries
  – Statically forwards inter-pod traffic on specified port

• Aggregate switches contain (10.pod.switch.0/24, port) entries
  – Switch value is the edge switch number

• Assumes a central entity with full knowledge of topology generates these routing tables
  – Also responsible for detecting switch failures and re-routing traffic

Routing Algorithm Example

Figure 3: Simple Fat-tree topology. Using the two-level routing tables described in Section 3.3, packets from source 10.0.1.2 to destination 10.2.0.3 would take the dashed path.
Dynamic Routing Techniques

- Alternatives to two-level routing table
  - Attempt to classify and schedule flows rather than use static routing
- Flow Classification
  - Periodically reassigns flow output ports
  - Prevents competition between flows for a single port
- Flow Scheduling
  - Identify large flows and establish reserved paths for them
  - Requires communication between edge switches and a central flow scheduler

Fault Tolerance

- Many possible paths between hosts leads to “easy” fault tolerance
- Each switch maintains Bidirectional Forwarding Detection session with neighbors
  - Allows switch to determine when neighbors fail
- Two primary types of link failure
  - Between lower and upper switches
  - Between upper and core switches

Router Power and Heat Dissipation

- Bar charts showing power and heat dissipation for different router models:
  - Catalyst 4948
  - ProCurve 2900-48G
  - Nortel G4S 7352S
  - Catalyst 6509-E
  - BigIron RX-32
  - BlackDiamond 10808

Topology Power/Heat Dissipation

- Bar charts showing total power and total heat dissipation for hierarchical and fat-tree designs:
Cafarella, 2013

Power & Related Costs

- Some numbers from James Hamilton...
  - Facility: ~$200M for 15MW facility
  - Servers: ~$2k/ea, about 50k of them
  - Power draw at 30% util: 80%
  - Commercial power: 0.07/kWh

Monthly Costs

<table>
<thead>
<tr>
<th>Category</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Servers</td>
<td>$284,686</td>
</tr>
<tr>
<td>Power &amp; Cooling Infrastructure</td>
<td>$1,296,902</td>
</tr>
<tr>
<td>Power</td>
<td>$2,997,090</td>
</tr>
<tr>
<td>Other Infrastructure</td>
<td>$1,042,440</td>
</tr>
</tbody>
</table>

Note: power & related hardware costs high
Servers going down in price; power flat or up

Hamilton, 2008

Cafarella, 2013

Heat

- Servers account for barely half of power
  - 1W of cooling per 1.5W of IT load
- Managing energy consumption means, to a large extent, managing heat

Emerson Network Power, 2007

Software Implementation

- Validated in software using Click
  - Click is a modular software router architecture
  - Implement routers on PCs, supports experimental router designs
- Click modules called “elements”
  - Each element performs a specified task
  - Routing table lookup, decrement packet TTL, etc...
- Implemented elements for two-level table, flow classifier, and flow scheduler

Cafarella, 2013

Datacenter Energy Growth

- Installed base grows 11%/year
- In 2011, 2.5% of all US energy use

EPA, 2007
Evaluation Setup

- Uses a 4-port fat-tree as seen previously
  - Two-level table and flow-based schemes analyzed
  - Compared against hierarchical tree with oversubscription ratio of 3.6:1
- Both evaluated using Click
  - Emulate switches and hosts on PCs
- All hosts generate 96 Mbit/s of outgoing traffic
  - This value prevents CPU from throttling test

Evaluation Results

- Percentages indicate aggregate network bandwidth
  - Measured as amount of incoming traffic received by hosts

<table>
<thead>
<tr>
<th>Test</th>
<th>Tree</th>
<th>Two-Level Table</th>
<th>Flow Classification</th>
<th>Flow Scheduling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>53.4%</td>
<td>75.0%</td>
<td>76.3%</td>
<td>93.5%</td>
</tr>
<tr>
<td>Stride (1)</td>
<td>100.0%</td>
<td>100.0%</td>
<td>100.0%</td>
<td>100.0%</td>
</tr>
<tr>
<td>Stride (2)</td>
<td>78.1%</td>
<td>100.0%</td>
<td>100.0%</td>
<td>99.5%</td>
</tr>
<tr>
<td>Stride (4)</td>
<td>27.9%</td>
<td>100.0%</td>
<td>100.0%</td>
<td>100.0%</td>
</tr>
<tr>
<td>Stride (8)</td>
<td>28.0%</td>
<td>100.0%</td>
<td>100.0%</td>
<td>99.9%</td>
</tr>
<tr>
<td>Staggered Prob (1.0, 0.0)</td>
<td>100.0%</td>
<td>100.0%</td>
<td>100.0%</td>
<td>100.0%</td>
</tr>
<tr>
<td>Staggered Prob (0.5, 0.3)</td>
<td>83.6%</td>
<td>82.0%</td>
<td>86.2%</td>
<td>93.4%</td>
</tr>
<tr>
<td>Staggered Prob (0.2, 0.3)</td>
<td>64.9%</td>
<td>75.6%</td>
<td>80.2%</td>
<td>88.5%</td>
</tr>
</tbody>
</table>

Worst cases:
- Inter-pod Incoming: 28.0% 50.6% 75.1% 99.9%
- Same-ID Outgoing: 27.8% 38.5% 75.4% 87.4%

Flow Scheduler Requirements

- Minimal time and memory requirements for flow scheduler
- Feasible to use at least until k grows extremely large

Packaging Problem

- Fat-tree has significant cabling overhead
  - 1 GigE switches used to reduce cost
  - Lack of 10 GigE ports leads to more cabling
- Present a packaging solution for k=48
  - Generalizes to other values of k

<table>
<thead>
<tr>
<th>k</th>
<th>Hosts</th>
<th>Avg Time/ Req (µs)</th>
<th>Link-state Memory</th>
<th>Flow-state Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>16</td>
<td>50.9</td>
<td>64 B</td>
<td>4 KB</td>
</tr>
<tr>
<td>16</td>
<td>1,024</td>
<td>55.3</td>
<td>4 KB</td>
<td>205 KB</td>
</tr>
<tr>
<td>24</td>
<td>3,456</td>
<td>116.8</td>
<td>14 KB</td>
<td>691 KB</td>
</tr>
<tr>
<td>32</td>
<td>8,192</td>
<td>237.6</td>
<td>33 KB</td>
<td>1.64 MB</td>
</tr>
<tr>
<td>48</td>
<td>27,648</td>
<td>754.43</td>
<td>111 KB</td>
<td>5.53 MB</td>
</tr>
</tbody>
</table>

Table 3: The flow scheduler’s time and memory requirements.
Packaging Solution

Strengths

• Fat-tree architecture seems to outperform hierarchical solution
• Excellent power and heat reductions over hierarchical approach
• Evaluation methods were good overall with tests performed
• Data centers can easily switch to this new method

Weaknesses

• Language used in paper was confusing at times
  – Referred to pod switches as “aggregate switch”, “upper-layer switch”, and “upper pod switch” at various points
• Evaluation performed with small value of k=4
  – Would have been nice to see higher values of k tested
  – Academic project and resources were obviously a factor for evaluation

References

• Cafarella, M. (2013, April 20). Datacenters. EECS 485. Lecture conducted from University of Michigan, Ann Arbor.