Data Transport in Datacenter

Clos data center network provides multiple paths between pairs of ToR switches

Randomized load balancing cannot achieve full bisection bandwidth: flows collide with high probability

Centralized flow scheduler can only run periodically, due to monitoring and schedule computation and instantiation overhead ⇒ works well only for large flows and only if flows are network, not host or NIC, limited

Consequently, flows manage only 10% of potential throughput and total network utilization is < 50%

Multipath TCP (MPTCP)

Balance load by path selection and congestion control:
• explore multiple paths simultaneously
• link congestion response of subflows on different paths
• move traffic away from congested links

MPTCP opens multiple subflows (TCP connections) per application-level connection:
• subflows can be differentiated by port numbers or by assigning source and/or destination host multiple IP addresses
• number of subflows negotiated in the initial SYN exchange
• subflows are assigned paths by ECMP
• data delivery is striped across subflows

Each MPTCP subflow has its own sequence space and maintains its own congestion window ($cwnd$)
• on receiving an ACK, a subflow $r$ increases its $cwnd$ by a function of total $cwnd$ size across all subflows: $\text{MIN}(a/w_{\text{total}}, 1/w_r)$, $a$ an “aggressiveness” constant
• on loss, a subflow halves its own $cwnd$ only: $w_r = \frac{1}{2}$
• as a result, MPTCP moves traffic away from congested paths

Use of MPTCP is transparent to the app
Evaluation

Uses two kinds of simulation: packet-level and flow-level, numerical analysis to model throughput as a function of loss rate

On Fat-tree, VL2, and BCube topologies

**VL2**: a Clos network, like Fat-tree, but with order of magnitude higher core link bandwidth and randomized (ECMP) routing instead of static routing

**BCube**: a hypercube with servers connecting ethernet pods

Traffic Workload

Permutation matrix: each host is paired with a random host in a 1-1 mapping

Each flow is bulk-transfer with infinite data?

Flow-level simulation can simulate larger networks but is less accurate, does not model loss timeouts, for example

Also studied many-to-one (incast) matrix, not studied: all-to-all matrix

Link Rate and Statistical Multiplexing

VL2’s higher capacity core links allow for better statistical multiplexing than the smaller core links of BCube/Fat-tree

Flow Size and Statistical Multiplexing

To increase statistical multiplexing, and utilization, on small links, need larger number of smaller flows (each routed to a different core link)
Locality and Oversubscription

Full bisectional bandwidth: nonsensical goal?
• no app constantly sends at full-interface rate
• rack locality further reduces bisectional traffic

Allow for core oversubscription of potential load

512-node Fat-tree with 4:1 oversubscription, 1 connection per host, local destination in the same rack as source

Throughput and Oversubscription

Random traffic matrix: contention on access links
MPTCP increases throughput when core links are congested

0.25 is the minimum load to fill 4:1 oversubscription

Which Part of MPTCP Is Effective?

Multipathing improves performance, even when cwnd is not linked, but obtains different loss rates
• UNCOUPLED: data striped across multiple TCP connections
• Equal-weighted: smaller increase if more subflows, but doesn't move traffic away from congestion
• Packet scatter/spraying: per-packet, instead of per-flow, ECMP (under TCP)

Short-Flows’ Finish Times

Packet scatter/spray (under TCP) has lowest FCT, but attains low utilization because long flows back off due to transient congestion caused by short flows

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Short Flow Finish Time (mean/stdev)</th>
<th>Network Core Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>SINGLE-PATH TCP</td>
<td>78 ± 108 ms</td>
<td>25%</td>
</tr>
<tr>
<td>PACKETSCATTER</td>
<td>42 ± 63 ms</td>
<td>30%</td>
</tr>
<tr>
<td>EWTCP</td>
<td>80 ± 89 ms</td>
<td>57%</td>
</tr>
<tr>
<td>MPTCP</td>
<td>97 ± 106 ms</td>
<td>62%</td>
</tr>
<tr>
<td>UNCOUPLED</td>
<td>152 ± 158 ms</td>
<td>65%</td>
</tr>
</tbody>
</table>
Self Interference

For multi-sender applications, if there are multiple paths with different lengths, EWTCP and Packet scatter can cause long-path flows, with multiple congested links, to congest short-path flows.

MPTCP concentrates traffic on short paths, moving it away from long congested ones.

<table>
<thead>
<tr>
<th>Traffic Type</th>
<th>Throughput (Mb/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single-path</td>
<td>297</td>
</tr>
<tr>
<td>EW TCP</td>
<td>229</td>
</tr>
<tr>
<td>MPTCP</td>
<td>272</td>
</tr>
<tr>
<td>Packet Scatter</td>
<td>115</td>
</tr>
</tbody>
</table>

Dual-homed Fat-tree

Realistic traffic does not fill full-bisection bandwidth:

- Can oversubscribe core links, or
- If bottleneck is at host NICs: most hosts have 2 NICs, connect both to ToR switches, reduce ToR to aggregation switch connectivities

ToR switch redundancy also helps eliminate the biggest single cause of correlated node failures.

Single-path TCP cannot take advantage of this topology.

Dual-homed Fat-tree

Some apps can take advantage of rack locality.

Some flows are host limited.

On Amazon EC2

Doesn’t know topology or background traffic.

Hosts are virtual machines, may share a physical host.

65% of flows have 2 (50%), 3 (25%), up to 9 alternate paths.

For these, MPTCP with 4 subflows achieves 3x the throughput of single-path TCP.