eecs589

ADVANCED COMPUTER NETWORKS

Guo, C., et al., "Pingmesh: A Large-Scale System for Data Center Network Latency Measurement and Analysis" *Proc. of ACM SIGCOMM '15*, 45(4):139-152, Oct. 2015

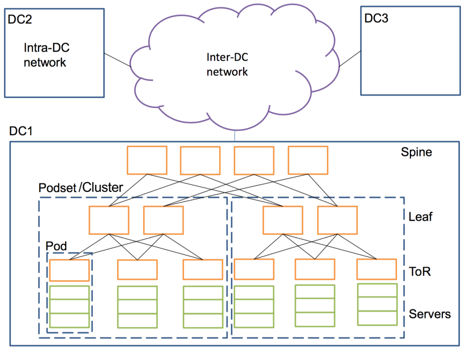
Microsoft's Datacenter Latency Diagnosis Tool

Goal: to "know" the latency between any two servers in a datacenter at any time

Purposes:

1. to diagnose whether any observed service degradation is caused by network performance
2. to track whether network performance meets service level agreement (SLA) with clients
3. to automate network troubleshooting

Constraints

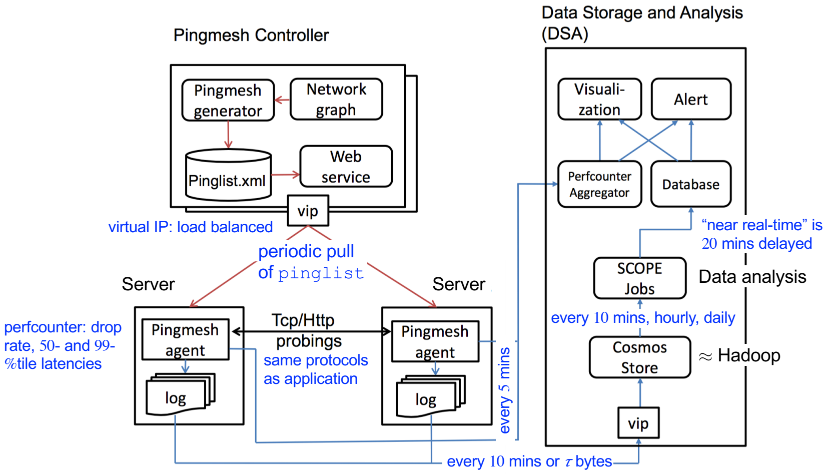


Must scale to 10^5's to millions of servers, 10^5's of switches, and millions of connections in a datacenter

Design decisions:

1. always-on or on-demand? it needs to be always on
2. all servers or only between certain pairs? use of ECMP load-balancing means the exact path of a connection is not known ⇒ we don't know which pairs to track to diagnose a given switch

Pingmesh Architecture



pinglist

Centrally computed, lists a pingmesh agent's probe targets, based on network topology

- a probe yields a RTT measure from TCP SYN/SYNACK
- each probe is a new TCP/HTTP connection with a new source port
- about 2K-5K targets per server

Scalability obtained by hierarchical probing:

1. per rack: all-pairs probing
   ⇒ complete graph of servers
2. intra-DC: 1-1 (*i*-to-*i*) probing across racks
   ⇒ complete graph of racks
3. inter-DC: several (unspecified) servers selected per cluster
   ⇒ complete graph of datacenters

Pingmesh Agent

Safety features:

1. CPU and memory usage capped
2. 10 seconds minimum probe interval, with maximum probe payload of 64 KB
3. stop probing after 3 tries or if no pinglist
4. if data upload fail after several tries, discard in-memory data; local logging of data is also size-capped
5. watchdogs to watch over every components

Pingmesh Agent

Overhead:

1. memory footprint < 45 MB [> MS DOS 640 KB RAM]
2. average CPU usage is 0.26% of Intel Xeon E5-2450
3. probe traffic averages 10's Kbps
4. total data upload: 24 TB/day or 2 Gbps
5. written in C++ not C# or Java to avoid runtime library and virtual machine overhead

Datacenter Latency

DC1: distributed storage and MapReduce, servers are throughput intensive:

- transmit and receive 100's Mbps
- 90% average CPU utilization

DC2: interactive search service, latency sensitive, servers have:

- high fan-in/fan-out, with low but bursty network traffic
- average CPU utilization moderate

Some results:

- inter-rack latencies higher than rack-internal latencies
- probes carrying payload have higher latencies than probes without payload, due to extra transmission delays

Datacenter Latency

Latencies below 90%-tile not that different between the two datacenters

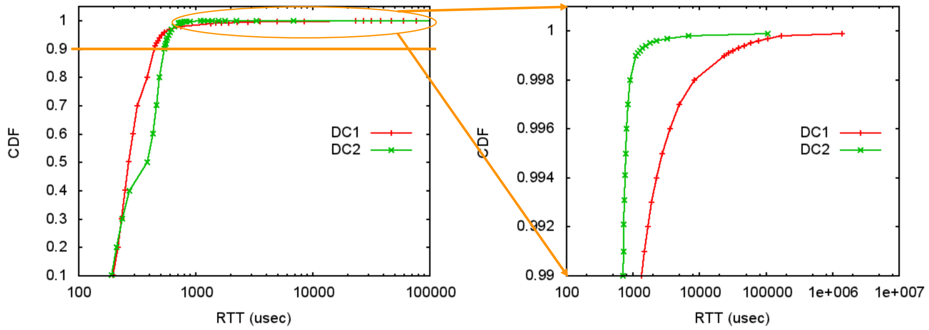
Transient long queues due to bursty traffic:

at 99.9%-tile:

- DC1: 23.35 ms
- DC2: 11.07 ms

at 99.99%-tile:

- DC1: 1.397 secs
- DC2: 105.84 ms



Packet Drop Rate

Estimated from TCP SYN/SYNACK probe failure:

$$\frac{\#probes_{1failure} + \#probes_{2failures}}{\#probes\_successful}$$

where:

- $\#probes_{1failure}$: # SYN packets dropped with one retry
- $\#probes_{2failures}$: # SYN packets dropped with two retries, but counted only once
- $\#probes\_successful$: successful probe, including after retries
- in short, (total number dropped)/(total number that got through)

Packet drop rate on the order of 10^{-5}, with inter-rack drop rate 2-6× higher than rack-internal drop rate

Network Troubleshooting

Problem: silent packet drop:

- specific source-destination pair gets dropped ⇒ due to flow table hardware (TCAM) corruption
- specific source-destination-transport tuple gets dropped ⇒ perhaps related to ECMP hashing
- both can be fixed by rebooting the switch

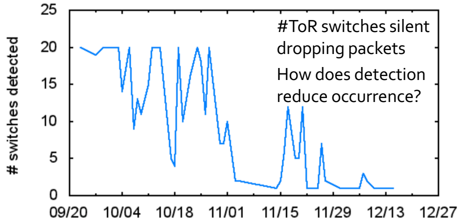
How to detect faulty switch?

Network Troubleshooting

How to detect faulty switch?

- if many servers under a ToR switch experience silent drop, the ToR switch is flagged
- if a small number of ToR switches in a cluster is flagged, they are probably faulty and are rebooted
- if a large number is flagged, a higher-level switch could be faulty ⇒ requires manual pinpointing, e.g., by using `traceroute`

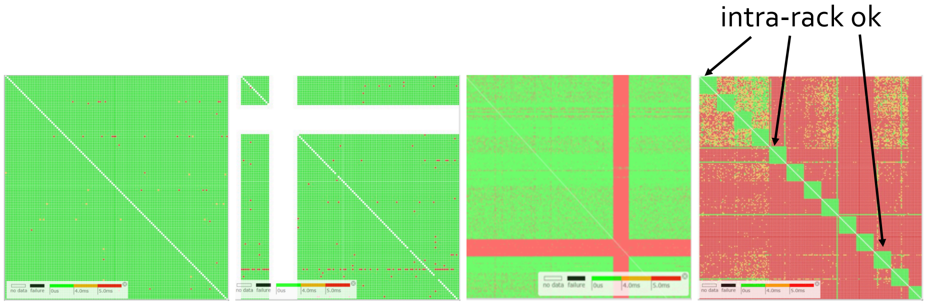
Pingmesh alone doesn't pinpoint faulty switch



#ToR switches silent dropping packets

How does detection reduce occurrence?

Fault Visualization



intra-rack ok

(a) Normal

(b) Cluster down

(c) Cluster failure

(d) Spine failure