Silo: Predictable Message Latency in the Cloud

Keon Jang – Intel Labs
Justine Sherry – UC Berkeley
Hitesh Ballani – Microsoft Research
Toby Moncaster – University of Cambridge

Presenter: Yinda Zhang
Public cloud datacenters
Public cloud datacenters

Predictable network performance across \textit{diverse requirements}

\begin{itemize}
\item Blue circle: Message driven apps (e.g., Online game server)
\item Purple circle: OLDI (Online Data Intensive) apps (e.g., search workload)
\item Red circle: Data-parallel (e.g., Map reduce)
\end{itemize}
Public cloud datacenters

Shared resources
Unpredictable network performance

10GbE switch

- Memcached
- Netperf

CDF

- Memcached alone
- Memcached with netperf

Latency (μs)

0 1000 2000 3000 4000

9x at 50th and 99th %-ile
TCP Timeout at 99.9th
Providing predictable message delay

Given:
1. Tenants know the size of message.
2. Tenants know their desired SLA.
What contributes to the latency?

Packet path at high-level:

- **Application**
- **Guest OS**
- **Hypervisor**
- **NIC**
- **Switch**
- **Network**

Latency contributors:
- Guest OS scheduling
- vCPU scheduling
- Batching
- Queuing delay
What contributes to the latency?

Packet path at high-level:

1. Application
2. Guest OS
3. Hypervisor
4. NIC
5. Switch
6. Network

Silo’s guarantee is NIC to NIC
Deconstructing message latency in network

Requirement 1: bandwidth guarantee
Requirement 2: packet delay guarantee
Requirement 3: burst allowance

Scenario: App generates 100kB message every 2ms with 1ms deadline
→ Avg bandwidth: 400Mbps

- No burst
- Burst 100KB at 1Gbps
Silo’s goal and requirements

Predictable network performance for wide range of applications

→ Predictable message latency

Requirements

1) Bandwidth guarantee
2) Packet delay guarantee
3) Burst allowance
Silo network guarantee

1. Bandwidth per VM (Hose model)
2. Latency between VMs
3. Burst allowance per VM
How Silo works?
Silo overview

- Traffic policing at the end host
  - No change to the network
  - Fine-grained pacing

- VM Placement
  - Ensure packet delay, bandwidth, and burst guarantees
  - Admission control
End host traffic policing

VM

Traffic

Silo Pacer

\( B_1, S \) \( B_2, S \) \( \ldots \) \( B_N, S \)

Bandwidth (B), Burst (S)

Per destination VM
Ensures hose model
Ensure burst of S
Average rate of B
Need for fine-grained pacing

Scenario: Software pacer with batching (64KB)

10 flows each with 500Mbps rate limit

300KB buffering

10Gbps

With coarse-grained pacing microburst from few servers overflows the queue

→ Need fine-grained pacing
Fine-grained pacing using void packets

![Diagram showing two VMs (VM1 2Gbps and VM2 1Gbps) connected to a SILO Pacer, which in turn is connected to a 10GbE NIC. The diagram includes a batch of 40 packets with one void packet indicated by 'V'. The batch size is calculated as 49.2 µs. The packet intervals are annotated as 2Gbps → 5 packet interval. Void packets are denoted by 'V'.]
Fine-grained pacing using void packets

Example void packet: *source mac address = dest mac address*
VM Placement
Queuing-aware VM placement

Tenant VM requirements:

- Bandwidth: **1Gbps**
- Burst: **100KB**

350KB buffering per port
10Gbps link

8 VM flows \(\rightarrow 400KB\) queuing

Packet loss
Queuing-aware VM placement

Tenant VM requirements: 350KB buffering per port 10Gbps link

8 VM flows → 400KB queuing

Bandwidth: 1Gbps Burst: 100KB

350KB buffering per port 10Gbps link

6 VM flows → 300KB queuing

The maximum queue buildup should be less than the buffer size:

\[ Q\text{-bound}_p \leq Q\text{-capacity}_p, \quad \forall p \in Path(i, j) \]
The sum of queue bounds across the path between them should be less than the delay guarantee:

$$\sum_{p \in Path(i,j)} Q\text{-capacity}_p \leq d,$$

The sum of bursts

$$S(t)$$

and

$$R\text{sum}(t)$$

q: max queuing delay at port

time

# of packets
VM Placement

The maximum queue buildup should be less than the buffer size:

\[ Q\text{-bound}_p \leq Q\text{-capacity}_p, \quad \forall p \in Path(i, j) \]

The sum of queue bounds across the path between them should be less than the delay guarantee:

\[
\sum_{p \in Path(i, j)} Q\text{-capacity}_p \leq d,
\]
VM Placement and admission control

- Generate a placement for a given request
- Check for constraints
  - Bandwidth and latency using network calculus
  - If fails generate a new placement
- If no placement meets constraint, reject the request
How Silo performs?
Evaluation

- Small scale: testbed experiments
  - Memcached with competing traffic

- Mid scale: packet-level simulation (NS2)
  - Comparison against the state of the art
    - HULL, DCTCP ...

- Large scale: flow level simulation
  - Quantify utilization tradeoff
Testbed experiment setup

- **Memcached** (3 x 5 = 15VMs)
  - 1 server and 14 clients
  - Mimic Facebook’s ETC trace [1]
  - Pareto inter-arrival time dist.

- **Netperf** (3 x 5 = 15VMs)
  - TCP all to all bulk transfer

[1] Workload Analysis of a Large-scale Key-value Store, SIGMETRICS’12
Memcached tail latency with Netperf

Silo guarantee: 2ms
Memcached tail latency with Netperf

Latency (ms)

99th %-ile

With Silo

Without Silo

Silo guarantee: 2ms

217.43

1.99

0.56

With Silo

Without Silo

29
NS2 – cloud emulation benchmark

- **Setup**
  - 10 Racks, 40 servers per rack, 8 VMs per server
  - 10Gbps / server and 40Gbps / ToR (1:10 oversubscription)

- **Workload**
  - **Class A tenants**: models online search query traffic (50%)
  - **Class B tenants**: models data parallel application (50%)

<table>
<thead>
<tr>
<th></th>
<th>Class A tenants</th>
<th>Class B tenants</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comm. pattern</td>
<td>All to 1</td>
<td>All to All</td>
</tr>
<tr>
<td>Traffic</td>
<td>Short repeated</td>
<td>Long bulk</td>
</tr>
<tr>
<td>Bandwidth</td>
<td>0.25Gbps</td>
<td>2Gbps</td>
</tr>
<tr>
<td>Burst</td>
<td>15KB</td>
<td>1.5KB</td>
</tr>
</tbody>
</table>

Exp. distribution
## NS2 – comparison with existing solutions

<table>
<thead>
<tr>
<th>Transport</th>
<th>Placement Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP</td>
<td>Packing (locality aware)</td>
</tr>
<tr>
<td>DCTCP</td>
<td>Packing (locality aware)</td>
</tr>
<tr>
<td>HULL</td>
<td>Packing (locality aware)</td>
</tr>
<tr>
<td>Okto</td>
<td>Bandwidth guarantee</td>
</tr>
<tr>
<td>Okto+</td>
<td>Bandwidth guarantee</td>
</tr>
</tbody>
</table>

Okto+ is extension to Oktopus with *burst allowance*
Message latency for Class A tenants

- **Silo**
- **TCP**
- **DCTCP**
- **HULL**
- **Okto**
- **Okto+**

The chart shows the median and 95th and 99th percentiles for message latency in milliseconds across different protocols.
Message latency for Class A tenants

Message latency (ms)

Silo ensures low latency even at tail.
Message latency for Class A tenants

Bandwidth guarantee *without burst* is slow
Message latency for Class A tenants

- Burst lowers latency but still suffers from packet loss and TCP Timeout without *delay aware placement*.
Message latency for Class B tenants

CDF (% of tenants)

Message latency / Estimated latency

Silo
HULL
TCP
Okto
Both
Burst
None
Placement

0 1 2 3 4

0 20 40 60 80 100

CDF (% of tenants)
Message latency for Class B tenants

Silo and Okto completes as estimate with bandwidth guarantee

CDF (% of tenants)

Message latency / Estimated latency

- Silo
- HULL
- TCP
- Okto

Placement:
- Both
- Burst
- None
- None

37
Message latency for Class B tenants

TCP and HULL completes message slower with fair-share bandwidth
Message latency for Class B tenants

TCP and HULL completes some messages faster with *work conservation*
Utilization tradeoff – large scale simulation

Silo
→ Bandwidth and delay aware placement

Okto
→ Bandwidth aware placement

Locality (TCP):
→ Packing placement
Utilization tradeoff – large scale simulation

Silo
→ Bandwidth and delay aware placement

Okto
→ Bandwidth aware placement

Locality (TCP):
→ Packing placement

Packing placement reduces network utilization

Network Utilization (%) vs. Datacenter occupancy (%)
Utilization tradeoff – large scale simulation

Cost of delay guarantee

Silo
→ Bandwidth and delay aware placement

Okto
→ Bandwidth aware placement

Locality (TCP):
→ Packing placement
Summary

- **Silo ensures predictable message latency**
  - Today: latency-critical workloads are restricted to private clusters
  - Silo: they can co-exist with bandwidth hogs

- **Simple design**
  - VM Placement satisfying both *latency* and *bandwidth* requirement
  - Hypervisor-based *fine-grained packet pacing* using *void packets*
  - *Unmodified* tenant stacks
  - *Unmodified* network switches