Storage Quality of Service for Enterprise Workloads

Tom Talpey
Eno Thereska
Microsoft
Outline

- Problem Statement
- Classification, Control and Costing
- Controller and Policies
- Demo
- Details
- Q&A
Background: Enterprise data centers

- General purpose applications
- Application runs on several VMs
  - E.g. a multi-tier service
- Separate networks for VM-to-VM traffic and VM-to-Storage traffic
- Storage is virtualized
- Resources are shared
  - Filesharing protocols deployed
The Storage problem

- First things first
  - In a datacenter, storage is a shared resource
  - Many tenants (customers)
    - Contention between tenants in the datacenter
  - Many workloads
  - Many backend device types and throughputs
  - Network resources are also shared
  - Provisioning key to providing cost-effective SLAs
More storage problems

- Storage throughput (capacity) changes with time
- Storage demand also changes with time
- SLAs do not

- Variety of storage protocols
  - SMB3, NFS, HTTP, …
SMB3 challenges

- Multichannel – Many to Many
  - Many SMB3 sessions (flows) on one channel
  - Many SMB3 channels used for each session
- SMB Direct
  - RDMA offload of bulk transfer – stack bypass
- Live Migration
  - Memory-to-memory with time requirements
  - Starvation of other flows must be avoided
Service Level Agreement (SLA) Examples

- Minimum guaranteed (“Not less than…”)
  - Storage bandwidth
  - Storage I/Os per second
- Maximum allowed (“Not more than…”)
- Opportunistic use (“If available…”)
  - Bandwidth/IOPS can use additional resources when available (and authorized)
- All limits apply across each tenant’s/customer’s traffic
Storage SLA Challenges

- Diverse operations mix
  - E.g. SMB3 Create, Read/Write, metadata
  - Resources, latencies, IO size diversity
- Dynamic, diverse workload
  - E.g. database tier vs web frontend vs E-mail
  - Small-random, large-sequential, metadata
- Bi-directional
  - Read vs write differing costs, resources
Can’t We Just Use Network Rate Limits?

- No. Consider:
  - The network flow is a 4-tuple which names only source and destination
  - Does not distinguish:
    - Read from write
    - Storage resources such as disk, share, user, etc
    - Even deep packet inspection doesn't help
      - Can’t go that deep – not all packets contain the full context
      - Might be encrypted
  - SMB multichannel and connection sharing
    - Many-to-many – mixing of adapters, users, sessions, etc
  - RDMA
    - Control and data separated, and offloaded
Classification – Storage “Flow”

1. Classify storage traffic into Flow buckets
   - Pre-defined “tuple” of higher-layer attributes
     - Can have *any number* of elements, and wildcards
     - Can therefore apply across many connections, users, etc.
   - For example (SMB3 and virtual machine):
     - VM identity
     - `\Servername\Sharename`
     - Filename
     - Operation type

2. Identify each operation as member of Flow

3. Apply policy to Flow
Elements of an IO classification

- Simple classifier tuple: (VM4, \share\dataset)

- Share appears as a block device Z: (\share\dataset)
- Block device maps to a VHD \serverX\RM79.vhd
- Maps to packets Dst IP: ServerX’s IP Dst port: 445
- VHD file maps to drive H:\RM79.vhd
- Maps to device \device\ssd5
Control (Rate Limiting)

- Apply limits (max) / guarantees (min) to each flow
- Rate-control operations to meet these
- Control at:
  - Initiator (client)
  - Target (server)
  - Any point in storage stack
Rate Limiting along the Flow

- Simple classifier tuple: { VM4, \share\dataset }
Control based on simple Classification?

- Again, no
- Taking the example SLA
  - `<VM 4, \share\dataset> --> Bandwidth B`
  - With contention from other VMs
- Comparing three strategies
  - By “payload” – actual request size
  - By “bytes” – read and write size
  - By “IOPS” – operation rate
Classification-only Rate Limit Fairness?

By payloads - reads beat writes by dominating queue

By bytes – writes beat reads by dominating (e.g. SSD) expense

By IOPS - large beats small by dominating bandwidth
Solution - Cost

- Compute a **cost** to each operation within a flow, and to a flow itself.
- "Controller" function in the network constructs empirical cost models based on:
  - Device type (e.g. RAM, SSD, HDD)
  - Server property
  - Workload characteristics (r/w ratio, size)
    - "Normalization" for large operations
    - Client property
  - Bandwidths and latencies
    - Network and storage property
  - Any other relevant attribute
- Cost model is assigned to each rate limiter
- Cost is managed globally across flows by distributed rate limiters
Cost of operations along the Flow

- Simple classifier tuple: \{ VM4, \share\dataset \}
Outline

- Problem Statement
- Classification, Control and Costing
- Controller and Policies
- Demo
- Details
- Q&A
Demo of storage QoS prototype

- Four tenants: Red, Blue, Yellow, Green
  - Each tenant
    - Rents 30 virtual machines on 10 servers that access shared storage
    - Pays the same amount
    - Should achieve the same IO performance
  - Tenants have different workloads
    - Red tenant is aggressive: generates more requests/second
Demo setup

Mellanox ConnectX-3 40Ge RNIC (RoCE)
120 one-Core VMs across 10 Hyper-V servers
- Run IoMeter 64K Read from 3GB VHD
Things to look for

- Enforcing storage QoS across 4 competing tenants
  - Aggressive tenant under control
- Inter-tenant work conservation
  - Bandwidth released by idle tenant given to active tenants
- Intra-tenant work conservation
  - Bandwidth of tenant’s idle VMs given to its active VMs
Demo: Red tenant dominates! 😞

Tenant performance is not proportional to its payment.

By being aggressive, red tenant gets better performance.
Research Prototype: With Storage QoS 😊

- Tenant performance is proportional to its payment

Algorithm notices red tenant’s performance

Tenants get equal performance
Behind the scenes: how did it work?

Data-plane Queues + Centralized controller

Programmable queues along the IO path

IO Packets

Queue 1 Queue n

High-level SLA

Controller

Control API
Data plane queues

1. Classification
   - [IO Header -> Queue]

2. Queue servicing
   - [Queue -> <rate, priority, size>]
Controller: Distributed, dynamic enforcement

- \( \{\text{Red VMs 1-4}, *, * \} //\text{share/dataset} \) --> Bandwidth 40 Gbps

- SLA needs per-VM enforcement
- Need to control the aggregate rate of VMs 1-4 that reside on different physical machines
- Static partitioning of bandwidth is sub-optimal
Work-conserving solution

- VMs with traffic demand should be able to send it as long as the aggregate rate does not exceed 40 Gbps

- Solution: Max-min fair sharing
Max-min fair sharing (intuition)

- Well-accepted notion of allocating a shared resource
  - Basic idea: Maximize the minimum allocation

- Formally, VMs [1..n] sharing rate $B$
  - Demand $D_i$ for VM $i$
  - Max-min fair share $f$ ensures that:
    - if the rate allocated to VM $i$ is $R_i = \min(f, D_i)$
    - then $\sum R_i \leq B$
      - No VM is given more than its demand
      - Total allocated rate doesn’t exceed $B$
Distributed rate limiting algorithm

- Allocate rate to VMs based on their demand
- No VM get a rate higher than its demand
- VMs with unmet demand get the same rate

VM Demand

<table>
<thead>
<tr>
<th>VM</th>
<th>Demand</th>
</tr>
</thead>
<tbody>
<tr>
<td>VM1</td>
<td>1</td>
</tr>
<tr>
<td>VM2</td>
<td>2</td>
</tr>
<tr>
<td>VM3</td>
<td>5</td>
</tr>
<tr>
<td>VM4</td>
<td>6</td>
</tr>
</tbody>
</table>

Unallocated Rate = 10
Unallocated VMs = 4
Current fair share = 10/4 = 2.5

Unallocated Rate = 7
Unallocated VMs = 2
Current fair share = 7/2 = 3.5

Algorithm achieves a max-min fair allocation

Aggregate rate = 10
Max-min fair sharing (recap)

- Well studied problem in networks
  - Existing solutions are distributed
    - Each VM varies its rate based on congestion
    - Converge to max-min sharing
  - *Drawbacks*: complex and requires congestion signal

- But we have a centralized controller
  - Converts to simple algorithm at controller
Controller decides where to enforce

Minimize # times IO is queued and distribute rate limiting load

**SLA constraints**
- Queues where resources shared
- Bandwidth enforced close to source
- Priority enforced end-to-end

**Efficiency considerations**
- Overhead in data plane ~ # queues
- Important at 40+ Gbps
Centralized vs. decentralized control

Centralized controller allows for simple algorithms that focus on SLA enforcement and not on distributed system challenges.

Analogous to benefits of centralized control in software-defined networking (SDN).
Summary and takeaways

- Storage QoS building blocks
  - Traffic classification
  - Rate limiters
  - Logically centralized controller
References / Other Work

- Predictable Data Centers
  - [http://research.microsoft.com/datacenters/](http://research.microsoft.com/datacenters/)

- Storage QoS - IOFlow (SOSP 2013)

- Network + storage QoS (OSDI 2014)

- SMB3 Rate Limiter (Jose Barreto’s blog)
  - [http://smb3.info](http://smb3.info)