Multi-Dimensional Storage Virtualization

Lan Huang  Gang Peng  Tzi-cker Chiueh

1st May, 2007
Talk Outline

- QoS specification
- Stonehenge architecture
- Achievement of QoS guarantees while trying to maximise utilisation
- Measurement-Based Admission Control (Related to Provisioning, also to guarantee delay bounds)
- Evaluations and Verification
Introduction to Storage QoS, Virtualisation

- QoS in Networks: studied extensively (e.g., FQ, WFQ ..., approximations of GPS, EDF scheduling variants)
- Typically, use leaky bucket to police flows and make bandwidth reservation easier
- Storage QoS: Harder because estimation of service time is hard
- Interference: Two workloads sharing same disk will see different bandwidth than if they had separate disks **Isolation** needed
- Fairness and Utilisation usually conflict. Need to trade off flexibly
A physical disk array is virtualised into several virtual disks - VDs.

User can attach any attributes as a real disk to a VD, i.e., bandwidth, access delay, capacity.

Each VD as tangible as a real disk.

Quintuple - ⟨ Availability, Bandwidth, Capacity, Delay, Elasticity ⟩. (“Multi-Dimensional”)

A and E not treated in this paper.

Mapping of VDs to physical disks - covered already in Polus.

Key Steps:

- All metrics converted into bandwidth numbers.
- Correlate worst-case delay bound and bandwidth reservation for a VD.
Stonehenge Architecture

1.2 Client asks for Virtual Disk

4 Storage Server Sends Cmd to Manager

3 Client Sends Disk Access Request to Storage Server

5 "Tags" requests and sends back to storage server

7 Workload Stats Sent to Manager Periodically

6 Storage Server Sends Result to Client

Storage Server

Storage Server Runs CVC scheduler

Lan Huang, Gang Peng, Tzi-cker Chiueh  Multi-Dimensional Storage Virtualization
Achieving QoS Guarantees

- Enforce QoS thro’ Disk Scheduling
- Admission Control: when to stop accepting Virtual Disks

Scheduler Requirements

- Should provide bandwidth guarantees
- Correlate delay specification, bandwidth reservation for each VD. Why? use bandwidth guarantee mechanisms to guarantee delay

- Virtual Clock (VC) Algorithm satisfies these requirements.
- Good property: Makes request streams independent
- Bad Property: Will penalise “misbehaving” streams - (Will affect spare bandwidth allocation)
- Compute timestamps $FT_{i,j}$ for request $i$ (size $= L_{i,j}$) from virtual stream $j$
- Keep track of Actual Arrival Time $AT_{i,j}$ of request
- Sort and dispatch job with smallest timestamp first

**Timestamp calculation for Network**

$$FT_{i,j} = \max(AT_{i,j}, FT_{i-1,j}) + L_{i,j}/B_j$$

**Timestamp calculation for Storage**

$$FT_{i,j} = \max(AT_{i,j}, FT_{i-1,j}) + (L_{i,j} + \delta)/B_j$$

$\delta = \text{avg\_overhead} \ast T$, $T = \text{total available bandwidth of physical disks}$.

- Storage algorithm takes into account seek and rotational latency. (Not queueing delay)
Increasing Utilisation

- We now have a service ordering. But it might not be optimal!
  ⇒ Utilisation might suffer
- Already Proved: VC reduces throughput by 40% vs CSCAN
- Use a modified Scheduler - CSCAN-based Virtual Clock
- 2 Queues maintained - “QoS Queue” and “Utilisation Queue”
- Utilisation Queue - CSCAN ordering.
- Scheduled only if FT violation not there at head of QoS Queue
- Checks only head of QoS Queue when deciding to visit queues

**FT Violation Check**

\[
\text{Service\_Time(HeadofQoSQueue)} + \text{Current\_Time} \leq LST_1
\]

*Service\_Time* is an estimate

*LST_i* is calculated for the whole QoS queue
Fairness Tradeoff vs. Disk Utilisation

- **Fairness?** CSCAN is fair, but *short-term unfairness can occur in spare bandwidth allocation*
- Because of VC property (the VC timestamp will have advanced, so *all* requests from other flows will be served)
- Still fair in the long term
- Can impose a limit on how many times Utilisation Queue can be serviced (consecutively)
- If this bound is $M$, large $M$ means unfairness in short term, but higher utilisation
- Stonehenge: Focus is on long-term fairness, so large $M$ used. (Every $M$ visits to Utilisation Queue, visit QoS Queue)
- Also have to account for Virtual Disk switching overhead (charge it to the right Virtual Disk) for VDs sharing physical disk
QoS well-studied in networks. How to apply it to Disks?

Network Formula - Express Delay as $f(\text{bandwidth})$

- $D_{\text{achievable}} \leq (\sigma_j + L_{\text{max}})/r_j + L_{\text{max}}/C_i$
- Constrained by leaky bucket with parameters $(\sigma_j, \rho_j)$
- $r_j$ is guaranteed rate for the connection, $C_i$ is link speed of the switch $i$, $L_{\text{max}}$ is the max pkt size
- Doesn’t take disk overhead into account, (“Effective Bandwidth”)
Bandwidth reservation for storage

- \( \text{avg\_overhead} = \frac{\text{inherent disk access overhead}}{\text{access count}} \)
- Expand request size to accommodate overhead in Delay bound calculation
- Certain time spent in seek+rotation, so certain portion of bandwidth lost
- \( \text{avg\_overhead} \) is measured, overhead time spent per request
- Could have gotten more equivalent no. of bytes out of disk in this time
- **Effective Bandwidth** is
  \[
  \frac{L}{B_{i\text{\_effective}}} = \frac{L + \text{avg\_overhead} \times T}{B_i}
  \]
New Delay Bound

- $D \leq \left( \text{overhead} \times T + \delta + L_{\text{max}} \right)/B_i + \left( L_{\text{max}} + \text{avg}\_\text{overhead} \times T \right)/T$
- $\delta = \text{max}\_\text{pending}\_\text{reqs} \times \text{avg}\_\text{req}\_\text{size}$
- $\text{overhead} = \text{avg}\_\text{overhead} \times (\text{max}\_\text{pending}\_\text{reqs} + 1)$

If we play down request size, then formula in terms of throughput:
$D \leq (\text{max}\_\text{pending}\_\text{reqs} + 1)/\text{IOPS}_i + 1/\text{IOPS}_{\text{max}}$
95% of the time, bandwidth usage is 58% of total bandwidth.

For 95% of the requests, the service time is 12% of estimated worst-case delays.
Measurement-Based Admission Control

- To avoid over-provisioning
- Because we made worst case assumptions for delay bound
- Because I/O rates cannot be estimated reliably
- Because r/w ratios of workloads not known
- Also to increase VDs that can be admitted
Measurement-Based Admission Control

- \( P^N_{\text{service}} \) - curve for \( N \) virtual disks.
- \( P^N_{\text{service}} \) used to find out \( P^{N+1}_{\text{service}} \)

The delay bound changes:
\[
D \leq (\frac{\text{max}_\text{pending}_\text{reqs} + 1}{\text{IOPS}_{N+1}} + 1/\text{IOPS}_{\text{max}}) \times
(\frac{Q^N_{\text{service}}(E_{N+1}) + s}{\text{E}_{N+1}})
\]

\( E_{N+1} \) is probability with Disk \( N + 1 \) needs delay bound to be met

\( s \) is an “adjustment factor” that will account for delay changes in the other admitted disks

A deterministic algorithm will assume all requests will see worst-case delay
Admission Control Decision

- If disk with spec \((IOPS', C, D, E)\) has a higher \(IOPS''\) (derived from delay bound), it is latency-bound
- Else it is throughput-bound
- \(P_{service}\) is used for delay-bound disks, runtime I/O rate measurements \((Q_{I/O\ Rate})\) are used for throughput-bound disks
- \(Q_{spare}(x) = IOPS_{max} - Q_{I/O\ Rate}\) spare bandwidth

Steps in Admission Control

- Calculate new Delay Bound for all Virtual Disks using \(Q_{service}\) (distribution has been measured)
- Group disks into latency-bound and throughput-bound
- Use measured I/O rate of throughput-bound disks to calculate \(Q_{spare}(x)\)
- For latency-bound disks, just sum up their \(IOPS_{max}\) and if \(\leq Q_{spare}(E)\), then admit new disk
Experimental Setup

- Small Network Delay ($\mu s$), iSCSI commands sent
- Clients run traces: TPC-C, Video Streams with 64KB request size, web search trace, financial trace
- VD0 - best-effort, VD1 - throughput and delay guarantees, VD2 - throughput guarantee

<table>
<thead>
<tr>
<th>Virtual Disk (VD)</th>
<th>Trace</th>
<th>Capacity [GB]</th>
<th>Throughput [IOPS]</th>
<th>Latency</th>
<th>I/O Rate Speed-up</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>TPC-C</td>
<td>9</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>1</td>
<td>Financial</td>
<td>15</td>
<td>128</td>
<td>120 msec</td>
<td>1.25</td>
</tr>
<tr>
<td>2</td>
<td>Web Search</td>
<td>96</td>
<td>675</td>
<td>N/A</td>
<td>2</td>
</tr>
</tbody>
</table>
QoS Guarantee

- Throughput under CSCAN vs. CVC (Input I/O rate exceeds throughput requirements and capacity)
- Latency bound violation for VD1 3% of requests
- For VD2, CSCAN doesn’t meet throughput requirements, but CVC does
Max. no. of VDs accepted under 3 schemes: Deterministic, MBAC, Oracle (assumes no limit of resources, keeps admitting until violation)

Oracle scheme will accept max. possible VDs

MBAC can produce 2-3 times higher admissions than Deterministic scheme
MBAC vs Deterministic: Deterministic Algorithm reserves 90% of disk throughput vs. 40% with MBAC

As QoS guarantee probability decreases, no. of VDs increases

Setting $s$: adding a VD increases $Q_{service}$ by 20%, so set $s = 0.2$
Spare bandwidth and Efficiency

- Vary resource reservation scaling factor $S$, Throughput reservations - 2$S$, 5$S$, 14$S$, $S=10$ means 0% spare bandwidth
- CVC 25% better than VC, 11% worse than CSCAN in utilisation for $S=2$
- As $S$ increases, CVC efficiency drops (not much choice), VC doesn’t change (load independent)
- VC reorders requests and spoils sequentiality
- Fully booked and overbooked systems: CVC can’t meet deadlines
Recall that $Service\_time(Head\ of\ QoSQueue)$ is used in scheduling.

If this estimate is conservative, lower delay bound violations, lower disk utilisation.

Comparison between CVC and CSCAN with various estimates (7ms - 20ms), utilisation equals CSCAN at 13.1.

Stonehenge: Dynamically measures distribution $P_{I/O\ Time}(x)$, probability that disk service time for request is $x$ ms.

Uses $P_{I/O\ Time}^{-1}(0.9)$ as estimate, gave same result as experiment, 13.1 ms.
THANK YOU!