Balancing fairness and efficiency in tiered storage systems with bottleneck-aware allocation

Hui Wang and Peter Varman
Rice University
FAST 2014
Motivation

- **Multi-tiered storage system**
  - Made up of SSDs and HDDs
  - Benefits
    - High performance
    - Low operating cost

- **Virtualized data centers**
  - Infrastructure-as-a-service
  - Benefits
    - Isolation
    - Consolidation

- However, using multi-tiered storage systems in virtualized data centers makes the already challenging problem of providing QoS more difficult.
Existing policies

• For single resource
  – Time-division based multiplexing
    • CFQ, Argon[FAST 2007], FIOS[FAST 2012]
  – Statistical multiplexing
    • SFQ(D)[SIGMETIRCES 2004], FlashFQ[ATC 2013]

• For multiple resources
  – DRF [NSDI 2011]
  – DRFQ [SIGCOMM 2012]

• However, existing policies focus on fairness more than performance.
Example 1

- Capacities
  - HDD 100 IOPS
  - SSD 500 IOPS

- Hit ratios
  - Client 1 0.5
  - Client 2 0.9

- Proportion fair
Example 1

- **Capacities**
  - HDD 100 IOPS
  - SSD 500 IOPS

- **Hit ratios**
  - Client 1 0.5
  - Client 2 0.9

- **DRF**

(c) Allocations using DRF. Allocations are in the ratio 9 : 25. Utilization of the SSD is 77%.
Example 1

- **Capacities**
  - HDD 100 IOPS
  - SSD 500 IOPS

- **Hit ratios**
  - Client 1 0.5
  - Client 2 0.9

- **Proportion fair**
- **DRF**

(b) Allocations with 100% utilization. Allocations are in the ratio 1 : 5.
• **Load of client i**
  - $h_{i}/C_{s}$ on SSD
  - $m_{i}/C_{d}$ on HDD

• **System load-balancing point**
  - $h_{bal} = C_{s} / C_{d} + C_{s}$
  - A workload with hit ratio equal to $h_{bal}$ will have equal load on both devices
  - If the hit ratio of a client is less than $h_{bal}$, it is said to be bottlenecked on the HDD
  - If the hit ratio is higher than $h_{bal}$, it is bottlenecked on SSD
Definition

• **Fair share**
  – Define the fair share of a client to be the throughput it gets if each of the resources are partitioned equally among all the clients.
  – Denote the fair share of client \( i \) by \( f \downarrow i \)

• Let \( A \downarrow i \) denote the allocation of (total IOPS done by) client \( i \) under some resource partitioning. The total throughput of the system is \( \sum \uparrow A \downarrow i \)
Analysis

• I/O mixing
  – Different clients have different hit ratio

• How to guaranteeing proportional fairness and full utilization?
  – The hit ratios are $\frac{\text{Capacity}_{\downarrow\text{SSD}}}{\text{Capacity}_{\downarrow\text{SSD}}} + \frac{\text{Capacity}_{\downarrow\text{HDD}}}{\text{Capacity}_{\downarrow\text{HDD}}}$

• Tradeoff
  – Fairness
  – Performance
Bottleneck-Aware Allocation

• Fairness policy

1. **Fairness between clients in** $D$:
   \[
   \forall i, j \in D, \quad \frac{A_i}{A_j} = \frac{f_i}{f_j}.
   \]
   Define $\rho_d = \frac{A_i}{f_i}$ to be the ratio of the allocation of client $i$ to its fair share, $i \in D$.

2. **Fairness between clients in** $S$:
   \[
   \forall i, j \in S, \quad \frac{A_i}{A_j} = \frac{f_i}{f_j}.
   \]
   Define $\rho_s = \frac{A_j}{f_j}$ to be the ratio of the allocation of client $j$ to its fair share, $j \in S$.

3. **Fairness between a client in** $D$ **and a client in** $S$:
   \[
   \forall i \in D, j \in S: \quad \frac{h_j}{h_i} \geq \frac{A_i}{A_j} \geq \frac{m_j}{m_i}.
   \]
   Note that if $h_i = 0$ then only the constraint $\frac{A_i}{A_j} \geq \frac{m_j}{m_i}$ is needed.
Performance optimization

• Maximize the throughput of the storage system, while satisfying the policy rules

\[ \sum_{k} A_k = (\sum_{i \in D} A_i + \sum_{j \in S} A_j) = (\rho_d \sum_{i \in D} f_i + \rho_s \sum_{j \in S} f_j) \]

The total number of IOPS made to the HD is:

\[ \rho_d \sum_{i \in D} f_i m_i + \rho_s \sum_{j \in S} f_j m_j. \]

The total number of IOPS made to the SSD is:

\[ \rho_d \sum_{i \in D} f_i h_i + \rho_s \sum_{j \in S} f_j h_j. \]
Performance optimization

• Maximize the throughput of the storage system, while satisfying the policy rules

Fairness rule 3 states that: \( \forall i \in D, j \in S, \)

\[
\frac{h_j}{h_i} \geq \frac{\rho_d f_i}{\rho_s f_j} \geq \frac{m_j}{m_i}
\]

\[
\frac{h_j f_j}{h_i f_i} \geq \frac{\rho_d}{\rho_s} \geq \frac{m_j f_j}{m_i f_i}.
\]

\[
\beta \geq \frac{\rho_d}{\rho_s} \geq \alpha.
\]

where

\[
\alpha = \max_{i,j} \left\{ \frac{m_j f_j}{m_i f_i} \right\} \quad \beta = \min_{i,j} \left\{ \frac{h_j f_j}{h_i f_i} \right\}
\]
Performance optimization

• Simplify the above equations

\[ \text{Maximize} \quad \rho_d \sum_{i \in D} f_i + \rho_s \sum_{j \in S} f_j \quad (1) \]

subject to:

\[ \rho_d \sum_{i \in D} f_i m_i + \rho_s \sum_{j \in S} f_j m_j \leq C_d \quad (2) \]

\[ \rho_d \sum_{i \in D} f_i h_i + \rho_s \sum_{j \in S} f_j h_j \leq C_s \quad (3) \]

\[ \beta \geq \frac{\rho_d}{\rho_s} \geq \alpha \quad (4) \]
Scheduling

- Algorithm

**Algorithm 1: Bottleneck-Aware Scheduling**

**Step 1.** For each client maintain statistics of its hit ratio over a configurable request-window $W$.

**Step 2.** Periodically invoke the BAA optimizer of Section 3.2 to compute the allocation of each client that maximizes utilization subject to fairness constraints.

**Step 3.** Use the allocations computed in Step 2 as relative weights to a proportional-share scheduler that dispatches requests to the array in the ratio of their weights.
Evaluation

- **Simulation**
  - FQ, DRF and BAA
  - Capacity
    - HDD 100 IOPS
    - SSD 5000 IOPS
  - Hit ratio
    - Client 1 0.5
    - Client 2 0.99

![Graph showing Throughputs for Fair Share, FQ, DRF, and BAA]

(a) Throughputs
Evaluation

• Simulation
  – FQ, DRF and BAA
  – Capacity
    • HDD 100 IOPS
    • SSD 5000 IOPS
  – Hit ratio
    • Client 1 0.5
    • Client 2 0.99

(b) Utilizations

SSD Utilization (%)
Evaluation

- **Simulation**
  - FQ, DRF and BAA
  - **Capacity**
    - HDD 100 IOPS
    - SSD 5000 IOPS
  - **Hit ratio**
    - Client 1 0.5
    - Client 2 0.99
    - Client 3 0.8

![Diagram showing throughput comparison between Fair Share, FQ, DRF, and BAA for different clients](image)
Evaluation

• Simulation
  – FQ, DRF and BAA
  – Capacity
    • HDD 100 IOPS
    • SSD 5000 IOPS
  – Hit ratio
    • Client 1 0.5
    • Client 2 0.99
    • Client 3 0.8
Evaluation

- Simulation
  - Capacity
    - HDD 200 IOPS
    - SSD 3000 IOPS
  - Hit ratio
    - Client 1 0.45
    - Client 2 0.95
  - Hit ratio window 60s
  - Allocation window 100s

Bottleneck-Aware Allocation
Evaluation

- **Linux experiments**
  - Throughput
  - Workloads
    - Financial workload, 0.3
    - Exchange workload, 0.95
    - 512B ~ 8MB

Figure 6: Throughputs using three schedulers. BAA achieves higher system throughput (1396 IOPS) than both DRF-based Allocation (810 IOPS) and Linux CFQ (1011 IOPS).
Evaluation

- **Linux experiments**
  - Device utilization
  - Workloads
    - Financial workload, 0.3
    - Exchange workload, 0.95
    - 512B ~ 8MB

Figure 7: System utilizations using three schedulers. The average utilization are: BAA (HD 94% and SSD 92%), DRF (HD 99% and SSD 78%), CFQ (HD 99.8% and SSD 83%)
Evaluation

- **Linux experiments**
  - Fairness
  - Workloads
    - Financial workload
    - Hit ratios, 0.2, 0.4, 0.98 and 1.0
    - 512B ~ 8MB

<table>
<thead>
<tr>
<th></th>
<th>Fair Share (IOPS)</th>
<th>Total IOPS</th>
<th>HD IOPS</th>
<th>SSD IOPS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Financial 1</strong></td>
<td>50</td>
<td>76</td>
<td>60.8</td>
<td>15.2</td>
</tr>
<tr>
<td><strong>Financial 2</strong></td>
<td>67</td>
<td>101</td>
<td>60.8</td>
<td>40.4</td>
</tr>
<tr>
<td><strong>Financial 3</strong></td>
<td>561</td>
<td>1068</td>
<td>21.4</td>
<td>1047</td>
</tr>
<tr>
<td><strong>Financial 4</strong></td>
<td>550</td>
<td>1047</td>
<td>0</td>
<td>1047</td>
</tr>
</tbody>
</table>

Table 3: Allocations for Financial workloads using BAA
Conclusion

• BAA
  – Multi-tiered storage system
  – Regard HDD and SSD as two types of resource
  – Balance fairness and performance