Towards Fair Sharing of Block Storage in a Multi-tenant Cloud

Xing Lin, Yun Mao*, Feifei Li, Robert Ricci
Cloud Computing

Key Idea: Resource Sharing

- Ecomonies of scale
- High utilization

Typical setup
Performance Unpredictability

Sharing results in interference

- Listed as the **Number 5 obstacle** for Cloud Computing (Above the Cloud: a Berkeley View of Cloud Computing)
- CPU and memory sharing work well in practice
- A dedicated session for network performance yesterday
- Here, we are looking into disk I/O sharing
Disk I/O Sharing

Disk I/O sharing is problematic

- Interference between random and sequential workloads
- Conflicts between read and write workloads

Can we build a cloud storage system with more predictable performance?
Interference Analysis - Workloads

- Use FIO to investigate interference between:
  - Random Read (RR)
  - Sequential Read (SR)
  - Random Write (RW)
  - Sequential Write (SW)
- Real-world application
  - TPC-H
Interference Analysis - Setup

- Disk: Seagate Cheetah 10,000 RPM 146 GB SCSI disk (pc3000 in Emulab)
- FIO benchmarks
  - 10 GB partitions
  - Direct IO
  - Block size: 4 KB
  - IO depth: 32
  - Runtime: 120 s
- Metrics: IOPS for random workloads and throughput for sequential workloads
Interference Analysis Result - I

Co-locating same type of workloads

**Observation 1:** When co-locating the same type of workloads, each workload gets a fair share in performance and system resources.
Interference Analysis Results - II

Co-locating different types of workloads:

- 50% of the performance of a RR workload when run in isolation with a SR workload

![Graph showing performance comparison of co-located workloads](image)
Observation 2: Random workloads are destructive to sequential workloads.
Observation 3: Random write workload is destructive for all other types of workloads.
Interference Analysis Result - III

- Real-world application: TPC-H
  - 21 TPC-H queries (random read)
  - sequential scan of 9 tables (sequential read)
Goal: want to build a block storage system, similar to Amazon EBS, with more predictable performance

- Assumptions
  - Inexpensive commodity components: replication
  - Exclusive ownership of a virtual volume
  - No assumption about workloads within VM
FAST – System Design

- System Design:
  - Directs random reads and sequential reads to different replicas
  - Log-structure to convert random writes into sequential
FAST – Architecture

Legend:

- Control messages
- Data messages

**Computenode**
- VM
- VM
- VM

**FAST client**
- Mapping cache

**Namenode**
- Tenant info table
- Replica-group info table
- Chunk mapping table

**Replication group 1**
- Datanode 1 (Buffered)
- Datanode 2 (Buffered)
- Datanode 3 (Direct IO)

**IO type**
- (tenantID, chunkID)
- (group, perm, valid)

**Status**
FAST – Architecture

Legend:
- Control messages
- Data messages

Computenode
- VM
- VM
- FAST client
- Mapping cache

(tenantID,chunkID)
(R/W(chunkid))

IO type
(perm)
(valid)

Namenode
- Tenant info table
- Replica-group info table
- Chunk mapping table

Replication group1
- Datanode1
  (Buffered)
- Datanode2
  (Buffered)
- Datanode3
  (Direct IO)

status

...
FAST – Disk Layout and Strategy

Chain Replication: Disk Layout and Write Policies

- **Default-with-steal strategy**
  - By default, random reads go to head node and sequential reads go to middle node.
  - Allows idle or lightly-loaded replicas to steal "requests" from other replicas
Initial Results – Simulation Setup

• Workloads:
  • One replication group
  • 30 tenants, each running one workload
  • 10 random read of 16 MB each
  • 10 sequential read of 19 MB each
  • 5 random write of 20 MB each
  • 5 sequential write of 20 MB each

• Workload assignment
  • Baseline: round-robin
  • FAST: workload type-aware
Initial Results - Assignment

Workloads: 10 RRs, 10 SRs, 5 RWs and 5 SWs

Baseline: (round-robin)

FAST

Replication group
Result 1: Write workloads in FAST get much better performance.
Result 2:

a). All SRs in FAST get similar performance

b). SRs in FAST get comparable or better performance than the baseline
Initial Results - Evaluations

Result 3:

a). All RRs in FAST get similar performance
b). RRs get worse performance in FAST

Replication group

Baseline

D1
4 RRs
3 SRs
5 RWs
5 SWs

D2
3 RRs
4 SRs
5 RWs
5 SWs

D3
3 RRs
3 SRs
5 RWs
5 SWs

Replication group

FAST

D1
10 RRs
5 RWs
5 SWs

D2
10 SRs
5 RWs
5 SWs

D3
5 RWs
5 SWs
1 SW
Future Work

- Modeling of effects of co-locating same type of workloads but with different I/O request characteristics
- Failure handling for datanode and namenode
- Load balancing among replication groups
- Tradeoff of chunk size
- System implementation
Conclusion

- Directs random and sequential reads to different replicas
- Introduce different write policies and disk layouts for chain replication
Thank you!

Questions?
Related Works and Contributions

- Related works
  - QoS-based resource allocation
    - Stonehege, Argon and Aqua
  - Support for latency control
    - SMART, BVT and pClock
  - Proportional share + limit and reservation
    - mClock

These work typically abstract the storage device to a single block device and rely on the lower layer to deal with replications.
IOPS – 1

From disk specification:
- Average (rotational) latency: 3.0 ms
- Average read seek time: 4.7 ms
- Average write seek time: 5.3 ms

For the whole disk:
- Theoretical read IOPS = $\frac{1000}{3+4.7} = 129.87$
- Theoretical write IOPS = $\frac{1000}{3+5.3} = 120.48$
- Measured read IOPS = 123
- Measured write IOPS = 222
From disk specification:

- Average (rotational) latency: 3.0 ms
- Average read seek time: 4.7 ms
- Average write seek time: 5.3 ms

For a 10GB partition:

- Theoretical read IOPS = \( \frac{1000}{3 + 4.7 \times \frac{10G}{146.8G}} \) = 301.19
- Theoretical write IOPS = \( \frac{1000}{3 + 5.3 \times \frac{10G}{146.8G}} \) = 297.53
- Measured read IOPS = 198
- Measured write IOPS = 339
RR with different think times
## SR with different block size

| Block Size | Throughput
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Isolation:</td>
</tr>
<tr>
<td>4k-SR</td>
<td>60.538 MB/s</td>
</tr>
<tr>
<td>256k-SR</td>
<td>73.755 MB/s</td>
</tr>
<tr>
<td>1m-SR</td>
<td>73.635 MB/s</td>
</tr>
</tbody>
</table>