Filo

consolidated consensus as a cloud service

Parisa Jalili Marandi, Christos Gkantsidis, Flavio Junqueira, Dushyanth Narayanan
Consensus

- Enables a set of distributed processes to reach agreement
  - Leader election, Membership
  - Coordinating access to shared objects
    - E.g., Paxos, Chain Replication, Two-Phase commit

- Many distributed systems need consensus
Many distributed systems are moving to cloud

How to implement consensus in a cloud environment?
Isolated consensus

Servers are dedicated to tenants

Underutilized Servers
Our Goal: Consolidated Consensus

- Lower $
- Efficient server utilization
- No management hurdles
Challenges with Consolidated Model

- Multi-tenancy
  - Performance isolation
  - SLA Guarantees: (requests/sec)
    - Users may misestimate their SLA

- Maximise resource usage on servers
  - CPU, Network, Storage

How to isolate performance and maximize resource usage?
1. Translate SLAs to raw resource usage
   e.g. 10K requests / s ➔ (10% CPU, 10K disk I/O, 80Mbps)
2. Monitor and adjust resource usage
Filo

1) Provides consensus as a shared multi-tenant service
2) Isolates Performance
3) Guarantees a minimum SLA
4) Optimizes resource usage
Filo at a high level

Admission Request

1. Durability mode
disk or memory
2. Replication degree
   3, 5, 7
3. Request size
   in bytes
4. Throughput SLA (High-level)
   in requests / second
1. Performance Analyser **initialization**

2. Admission Controller
   1. SLA Translation
   2. Placement

3. Resource controller
1. Performance Analyser

2. Admission Controller
   1. SLA Translation
   2. Placement

3. Resource controller
Performance Analyser

- Generates **performance profile**
  - Similar to [Quasar-SIGPLAN14], [Bazaar-SoCC12], [Matrix-ICAC14].
  - Large space to explore
  1. Control SLAs
  2. Translate high-level user SLAs to resource costs

- **Chain Replication** [OSDI-2004]
  - Or any other (e.g., Paxos)
1. Performance Analyser

2. Admission Controller
   1. SLA Translation
   2. Placement

3. Resource controller
Admission request
- Durability mode: in-memory
- SLA: 3000 requests/sec
- Request Size: 512 Bytes
- Number of Replicas: 4

Tenant is not limited to 512-B requests

Resource Budgets
- CPU: 10%
- Storage: 3000 Disk IO
- Network BW: 1.5 + MBs
1. Performance Analyser

2. Admission Controller
   1. SLA Translation
   2. Placement

3. Resource controller
Placement

- Multi-Resource Bin-Packing
  - Greedy approach
  - Respecting objectives and constraints:
    - Replicas of a consensus group on distinct servers

CostFunction()

Resource Budget

Storage IO  CPU  Network BW

Replica
However:
Tenant demand may be higher/lower than Resource Budget

Can we change Resource Budget at runtime?
Without violating others SLAs?
1. Performance Analyser
2. Admission Controller
   1. SLA Translation
   2. Placement
3. Resource controller
Resource Usage at Runtime
Centralized Resource Controller

Optimal resource usage but Slow
- Polynomial with # tenants
- Collect all information centrally

<table>
<thead>
<tr>
<th>Tenant</th>
<th>Granted Extra Requests</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>10 extra requests/sec</td>
</tr>
<tr>
<td></td>
<td>Size: 512 B</td>
</tr>
<tr>
<td>Bob</td>
<td>5 extra requests/sec</td>
</tr>
<tr>
<td></td>
<td>Size: 8KB</td>
</tr>
</tbody>
</table>
Distributed Resource Controller

- Slow computation
  - High resource usage

- Faster computation
  - resource usage?
Head-DRF
Dominant Resource Fairness [NSDI-2011]

Alice: 200 extra requests

Servers must have **consensus** on resource dissemination
ALL-DRF

Alice: 200 extra requests

Allocation Phase

Tune buckets

Resource stats

95

90

100
Evaluating Resource Controller

Head-DRF

All-DRF

Maximize resource usage Better

Better

Better

Number of Applicants

Utilized CPU(%)
Message Complexity

- Overhead is affordable given the many number of msgs exchanged for the service itself
1. Performance Analyser

2. Admission Controller
   1. SLA Translation
   2. Placement

3. Resource controller
Testbed

- 10 Dell servers each with 10-core Intel Xeon
- 10 Gbps Mellanox ConnectX-3 NIC
- 128 GB RAM
- Hyper threading enabled
- 2 HDDs
- Hierarchical Switches
Enable rate limiters
B becomes aggressive

Enable Resource controller
B demands more

C restores speed
A demands more

C slows down

Request size: 1 KB
Async disk IO

A-SLA: 6.5 K reqs/sec
B-SLA: 6.5 K reqs/sec
C-SLA: 6.5 K reqs/sec
Conclusions

- First system to provide consensus as a multi-tenant cloud service
  - A cheaper and convenient alternative for users
  - First distributed resource controller using DRF
Filo