Adaptation in distributed NoSQL data stores

Kostas Magoutis
Department of Computer Science and Engineering
University of Ioannina, Greece
Institute of Computer Science (ICS)
Foundation for Research and Technology – Hellas (FORTH)
Workload variations lead to SLO violations

Impact of workload increase
Adapt by adding a node

However, rebalancing has an impact
Background tasks lead to SLO violations

Throughput

Impact of background-task induced overload at leader node

Adapt by reorganizing replica groups (changing leader)

SummerSOC 2018
Agenda

• Workload or resource variations ⇒ SLO violations
  – Need to adapt to maintain SLO
  – Examples: Elasticity, rebalancing, reconfiguration

• Feedback-loop based adaptation
  – Performance modeling via systematic measurements
  – Importance of fast, light rebalancing actions

• Adapting via overhead-hiding operations
  – Replica group leadership change
  – Hide overhead at the leader
NoSQL data stores: overview

Data model

Horizontal partitions (shards)

Servers

Replicas

mapping

backup

primary

backup

Indexing

B+-tree

LSM

SummerSOC 2018
QoS architecture for NoSQL data stores

Maria Chalkiadaki and Kostas Magoutis, Managing Service Performance in the Cassandra Distributed Storage System, in *Proc. of 5th IEEE International Conference on Cloud Computing Technology and Science (CloudCom 2013)*
Provisioning methodology

- Prediction of service capacity requirements
- Tables of measured performance results
  - Response time
  - Throughput

<table>
<thead>
<tr>
<th># Clients</th>
<th>Workload: $W$; Server type: $S$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>clients₁</td>
<td>$r_1, t_1$</td>
</tr>
<tr>
<td>clients₂</td>
<td>$r_1, t_1$</td>
</tr>
<tr>
<td>clients₃</td>
<td>$r_1, t_1$</td>
</tr>
<tr>
<td>...</td>
<td>$r_1, t_1$</td>
</tr>
</tbody>
</table>

SummerSOC 2018
Provisioning methodology

QoS specification

- 100% reads
- Zipf distribution
- Load: 512 threads
- Resp. time: 35ms
Exploring the accuracy of different regression approaches

- Interpolation exhibits 70-80% (avg) prediction accuracy in most cases – can we improve on this?

- Evaluate prediction accuracy using more advanced regression methods
  - Multivariate adaptive regression splines (MARS)
  - Support vector regression (SVR)
  - Artificial neural networks (ANN)

Overall results

Predict performance for different cluster sizes

<table>
<thead>
<tr>
<th>Regression Method</th>
<th>Case 1 Accuracy</th>
<th>Case 2 Accuracy</th>
<th>Case 3 Accuracy</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>MARS</td>
<td>96.93</td>
<td>97.81</td>
<td>98.82</td>
<td>97.85</td>
</tr>
<tr>
<td>SVR</td>
<td>93.97</td>
<td>92.42</td>
<td>96.11</td>
<td>94.16</td>
</tr>
<tr>
<td>ANN</td>
<td>92.11</td>
<td>89.85</td>
<td>89.22</td>
<td>90.39</td>
</tr>
</tbody>
</table>


SummerSOC 2018
Overall results

Predict performance for different load levels

<table>
<thead>
<tr>
<th>Regression Method</th>
<th>Case 1 Accuracy</th>
<th>Case 2 Accuracy</th>
<th>Case 3 Accuracy</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>MARS</td>
<td>96.93</td>
<td>97.81</td>
<td>98.82</td>
<td>97.85</td>
</tr>
<tr>
<td>SVR</td>
<td>93.97</td>
<td>92.42</td>
<td>96.11</td>
<td>94.16</td>
</tr>
<tr>
<td>ANN</td>
<td>92.11</td>
<td>89.85</td>
<td>89.22</td>
<td>90.39</td>
</tr>
</tbody>
</table>
Overall results

Predict performance for different update settings

<table>
<thead>
<tr>
<th>Regression Method</th>
<th>Case 1 Accuracy</th>
<th>Case 2 Accuracy</th>
<th>Case 3 Accuracy</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>MARS</td>
<td>96.93</td>
<td>97.81</td>
<td>98.82</td>
<td>97.85</td>
</tr>
<tr>
<td>SVR</td>
<td>93.97</td>
<td>92.42</td>
<td>96.11</td>
<td>94.16</td>
</tr>
<tr>
<td>ANN</td>
<td>92.11</td>
<td>89.85</td>
<td>89.22</td>
<td>90.39</td>
</tr>
</tbody>
</table>


SummerSOC 2018
Overall results

MARS provides better accuracy in all test cases

<table>
<thead>
<tr>
<th>Regression Method</th>
<th>Case 1 Accuracy</th>
<th>Case 2 Accuracy</th>
<th>Case 3 Accuracy</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>MARS</td>
<td>96.93</td>
<td>97.81</td>
<td>98.82</td>
<td>97.85</td>
</tr>
<tr>
<td>SVR</td>
<td>93.97</td>
<td>92.42</td>
<td>96.11</td>
<td>94.16</td>
</tr>
<tr>
<td>ANN</td>
<td>92.11</td>
<td>89.85</td>
<td>89.22</td>
<td>90.39</td>
</tr>
</tbody>
</table>

- MARS provides excellent accuracy
- SVR, ANN involve tuning (kernel, activation function)


SummerSOC 2018
Feedback-based control

$r(k)$: desired value of measured output, e.g., 66% CPU utilization

Difference between reference input and measured output

$u(k)$: setting of parameter(s) that manipulate the system

$y(k)$: measurable characteristic of target system (e.g. CPU)

Transform the measured output so that it can be compared to reference input (e.g., smoothing)

Behavior of a stable system

Measured output, eventually converges to $y_{ss}=3$

Steady-state error $e_{ss}=r_{ss}-y_{ss}=-1$

Reference input $r_{ss}$ changes from 0 to 2

Settling time $k_s$

Maximum overshoot

Goals: Stability, Accuracy, Short settling times, does not Overshoot (SASO)

Integral control

Integral controller: provides incremental adjustments to $u(k)$

$$u(k + 1) = u(k) + K_I e(k)$$

$y(k)$: measurable characteristic of target system (e.g. CPU)

Transform the measured output so that it can be compared to reference input (e.g., smoothing)

Reducing the impact of data rebalancing via incremental elasticity

Results in smoother elasticity action

Processing capacity at joining node

Antonis Papaioannou and Kostas Magoutis, Incremental elasticity for NoSQL data stores, in Proc. of 36th Symposium on Reliable Distributed Systems (SRDS 2017), Hong Kong, China, September 27-29, 2017 (full paper)

Antonis Papaioannou and Kostas Magoutis, Incremental elasticity for NoSQL data stores, in Proc. of 37th IEEE International Conference on Distributed Computing Systems (ICDCS 2017), Atlanta, GA, USA, June 5-8, 2017 (poster)
Impact on response-time SLO

Further response-time increase

Smoother transition to new state

- YCSB Workload B (95%-5%), SLO 50ms
- Load surge 20->30 YCSB threads
- Elasticity action 5 mins after surge
Adapting to impact of background tasks

Impact of background-task induced overload at leader node

Adapt by reorganizing replica groups (changing leader)
Replica-group leadership change as a performance enhancing mechanism

- Proactive replica group reorganizations provide rapid remedy to upcoming performance issues
  - Lightweight adaptation actions

- Replica group management increasingly possible via programmable APIs in NoSQL data stores
  - Examples: MongoDB, RethinkDB (both primary-backup)

A. Papaioannou, K. Magoutis, “Replica-group leadership change as a performance enhancing mechanism in NoSQL data stores”, 38th IEEE International Conference on Distributed Computing Systems (ICDCS’18), Vienna, Austria, Jul 6-9, 2018

High-level view of replicated data store

Client

Client Requests

Primary

Replication group

Backup

key data

key data

key data

key data

Replication group

Client

LSM

LSM

LSM

LSM

LSM

SummerSOC 2018
Log-structured merge (LSM) trees

An LSM-tree of K+1 components

Disk

Memory

put (write)

commit

flush

Disk

SSTables

Write-ahead log (WAL)

Compactions

SummerSOC 2018
High-level view of replicated data store

Key idea: Change leader before it starts a compaction

SummerSOC 2018
When to change a leader, whom to elect

A. Papaioannou, K. Magoutis, “Replica-group leadership change as a performance enhancing mechanism in NoSQL data stores”, 38th IEEE International Conference on Distributed Computing Systems (ICDCS’18), Vienna, Austria, Jul 6-9, 2018
Experimental results

Standard MongoDB RocksDB

90% reads, 10% writes

50% reads, 50% writes

SummerSOC 2018
Experimental results

MongoDB RocksDB with leadership changes

90% reads, 10% writes

50% reads, 50% writes
Data backup

MongoDB RocksDB

MongoDB RocksDB with leadership changes

SummerSOC 2018
Cross-layer management of data stores

YCSB

RethinkDB

Management API

Cluster

SummerSOC 2018
Cross-layer management of data stores

RethinkDB

Management API

Cluster

Pod
monitor

Autoscaler

ReplicaSet

YCSB

kubernetes

Google Cloud Platform
Cross-layer management of data stores

- **RethinkDB**
  - Management API
  - Cluster
  - Pod
    - monitor
  - hook
  - Events

- **YCSB**

- **kubernetes**
  - Container hooks
  - Autoscaler
  - ReplicaSet

- **Events:**
  - Starting up
  - Going down
  - Reduced performance

- **Manager**

- **Monitoring**

- **Google Cloud Platform**

*SummerSOC 2018*
Cross-layer management of data stores


Experimental testbed

*shard = horizontal partition*

Google Cloud Engine (GCE)

- 2 vCPUs 2.6GHz Intel Xeon E5
- 13GB RAM
- SSD

YCSB settings
- 50-50 reads/writes
- 16 client threads
- target 1000 operations/sec

SummerSOC 2018
Temporary offload via RG reorganization

- Fewer resources available for a short period of time
- Reconfigure replica groups, resume service

- Move $S_1$ primary out of Node 1
Temporary offload via RG reorganization
Summary

• Proactive replica group reorganizations provide rapid remedy to upcoming performance issues
  – Lightweight adaptation actions

• Functionality previously unavailable as infrastructure-level events invisible to NoSQL middleware
  – Richer feedback useful: How long is impact expected to last?
References

- A. Papaioannou, K. Magoutis, “Replica-group leadership change as a performance enhancing mechanism in NoSQL data stores”, 38th IEEE International Conference on Distributed Computing Systems (ICDCS’18), Vienna, Austria, Jul 6-9, 2018
- Antonis Papaioannou and Kostas Magoutis, “Incremental elasticity for NoSQL data stores”, in Proc. of 36th Symposium on Reliable Distributed Systems (SRDS 2017), Hong Kong, China, Sep 27-29, 2017
Questions?

H2020 GA no. 731846 EU project