Cloud Native Database Systems at Alibaba: Opportunities and Challenges

FeiFei Li
VP of Alibaba Group
President of the Database Products Business Units, Alibaba Cloud Intelligence
Director of Database and Storage Lab, DAMO Academy
DB-Engines Database Trend (May 2019)

Relational DBMS (137) & NoSQL (210)

Number of systems per category, May 2019

- Wide column stores: 12
- Time Series DBMS: 30
- Search engines: 18
- Document stores: 48
- Event Stores: 3
- Graph DBMS: 13
- Key-value stores: 67
- Multivalue DBMS: 10
- Native XML DBMS: 7
- Spatial DBMS: 2
- Other oriented DBMS: 20

Ranking scores per category in percent, May 2019

- Wide column stores: 3.1%
- Time Series DBMS: 0.5%
- Search engines: 4.7%
- Document stores: 8.8%
- Graph DBMS: 1.5%
- Key-value stores: 4.9%
- Multivalue DBMS: 0.3%
- Native XML DBMS: 0.3%
- RDF stores: 0.4%

Commercial License (178) vs Open Source (169)

Popularity trend

The top 5 commercial systems, May 2019

1. Oracle 1286 1.
2. Microsoft SQL Server 1072 3.
3. IBM Db2 174 6.
5. Splunk 85 13.

The top 5 open source systems, May 2019

1. MySQL 1219 2.
2. PostgreSQL 479 4.
4. Elasticsearch 149 7.
5. Redis 148 8.
“The real battle will be in databases”

Source:
- “How Amazon Web Services aims to win cloud computing’s next big battle” SiliconANGLE
- “AWS to Oracle: Now it’s our turn and we got next” ZDNet
Alibaba Database Usage — Alibaba Group

Digital Media & Entertainment

Core E-Commerce

Local Services

Financial Service

Logistics

Marketing

Cloud Computing

870 Million
Active Users of Alipay
Around the Globe
Serving 15 million SMEs worldwide

100 Million
Packages Processed by Cai Niao
Logistic Network Each Day
Covering 15 hundred cities and
counties in mainland China

$768 Billion
GMV of FY2018
552 million monthly active users
Singles Day (11/11) 2018

Database workloads (~500,000 sales transactions per second at peak, which leads to a few million TPS)

122x

First second on 00:00:01, 11/11/2018

Hundreds of thousands of DB instances, >tens of PB of data
DB Revenue Report by Gartner

### Table 1. DBMS Cloud Services Revenue (2016-2018)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon</td>
<td>1,700.64</td>
<td>3,615.90</td>
<td>6,319.11</td>
<td>112.62%</td>
<td>74.76%</td>
</tr>
<tr>
<td>Microsoft</td>
<td>53.38</td>
<td>918.27</td>
<td>2,149.40</td>
<td>1620.11%</td>
<td>134.07%</td>
</tr>
<tr>
<td>Alibaba</td>
<td>96.93</td>
<td>213.44</td>
<td>460.55</td>
<td>120.21%</td>
<td>115.77%</td>
</tr>
<tr>
<td>Oracle</td>
<td>100.16</td>
<td>224.76</td>
<td>373.12</td>
<td>124.40%</td>
<td>66.01%</td>
</tr>
<tr>
<td>Google</td>
<td>101.47</td>
<td>164.36</td>
<td>285.49</td>
<td>61.98%</td>
<td>73.70%</td>
</tr>
<tr>
<td>Tencent</td>
<td>21.87</td>
<td>110.85</td>
<td>247.30</td>
<td>406.96%</td>
<td>123.09%</td>
</tr>
<tr>
<td>Huawei</td>
<td>77.42</td>
<td>70.99</td>
<td>137.87</td>
<td>-8.32%</td>
<td>94.22%</td>
</tr>
<tr>
<td>IBM</td>
<td>57.28</td>
<td>73.35</td>
<td>120.22</td>
<td>28.06%</td>
<td>63.90%</td>
</tr>
<tr>
<td>Cloudera</td>
<td>23.85</td>
<td>45.35</td>
<td>79.21</td>
<td>90.15%</td>
<td>74.66%</td>
</tr>
<tr>
<td>MongoDB</td>
<td>9.12</td>
<td>8.77</td>
<td>65.70</td>
<td>-3.90%</td>
<td>649.33%</td>
</tr>
<tr>
<td>Other</td>
<td>127.07</td>
<td>171.69</td>
<td>250.74</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Revenue 2016-2018

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Alibaba</td>
<td>96.93</td>
<td>213.44</td>
<td>460.55</td>
<td>120.21%</td>
<td>115.77%</td>
</tr>
</tbody>
</table>

Data sourced from "Market Share: Enterprise Platform as a Service, Worldwide, 2018"

Source: Gartner (June 2018)
Background - Do More with Less

More
- Availability
- Security
- Stability
- Scalability
- Performance

Less
- Human Labor
- Machines

> 100 BUs
> 1,000,000 DBs

< 50 DBAs

x1000
Challenges

Management at Scale
- Scheduling
- Protection
- Runtime Management
- Optimization
- Backup/Restore
- Security
  ...

Hotspot
- Scalability
- Elasticity
- Stability
- Cost
  ...

Workloads Diversity
- SLA-driven
- Workload-aware
- Agility
- >100 BUs
- >20,000 Developers
  ...

...
Journey to Cloud Native Database Systems

Human Labor
- DBAs

Commercial Database
- Oracle
- ~2010

Open Source Database
- AliSQL/MySQL
- 2010~2017

Large-scale DB Lifecycle Management Infra

DBPaaS

Cloud-Native Database
- POLARDB (VLDB 2018, SIGMOD 2019)
- AnalyticDB (VLDB 2019)
- 2017~now

SDDP
Self-Driving Database Platform
Three Different Architectures

- **Single Instance**: Local disk
  - Example: MySQL, PG

- **Shared Everything**: Shared storage
  - Example: POLARDB

- **Shared Nothing**: Disk
  - Example: PolarDB-X, AnalyticDB

Scale up and Scale out options are also indicated in the diagram.
Background of POLARDB

- Design of POLARDB – A Cloud Native Database

- Elasticity
- Low Cost
- Performance
- Continuous solution
POLARDB Key Features

1 M QPS
- 100k POLARDB
- 500k POLARDB
- 200k RDS for MySQL

Read Performance

130K TPS
- 150k POLARDB
- 100k RDS for MySQL
- 50k RDS for MySQL

Write Performance

2 min
- 10TB Data

RO Copies

3 min
- Primary
- Read Only

Remote Instance

Remote Backup

- 100TB Capacity
- 100% MySQL/PG Compatible
POLARDB Architecture -- Overview

- **Low Latency Oriented**
  - Pure Userspace I/O Stack
  - Zero Copy, RDMA

- **Design for Emerging Hardware**
  - RDMA NIC
  - NVMe SSD, Optane

- **Active R/W – Active RO**
  - One Write / Multiple Read

- **High Availability**
  - Three Replicas
  - ParallelRaft
POLARDB Architecture

Active-Active + Decouple of Reads and Writes + Serverless Storage
POLARDB Architecture

Intelligent proxy
PolarProxy

100% Compatible
POLARDB
PolarFS

Storage Optimized For Database
PolarStore

ECS

Cluster Address

Master Address

Master
RO
RO

PolarProxy

PolarStore
POLARDB Architecture

- **Primary**
- **Reader**
- **Reader**

**Distributed File System**

- **polarFS**

**Database cluster**

- **polarFS**

**Distributed shared storage**

- **A**
- **a**
- **a**

**RDMA&SPDK**

**ParallelRaft**

**Lockless storage**

**Excellent Elasticity**
- 2core vCPU upgrades to 32core in < 5mins
- 2 nodes scales out to 4 nodes < 5mins

**Cost reduction**

**Serverless on-demand billing**

**100T, max-capacity**

**Elasticity: scale-out**

**Lockless Backup**
- Snapshot backup without locking
Architecture of PolarFS (VLDB 2018) -- Inside

Architecture of PolarFS -- Components

- 1. libpfs
  - Library in Userspace
  - Enable I/O without trapping into Kernel
  - POSIX-like File System API
  - Easy to port different kernels to run on PolarFS

```c
int pfs_mount(const char *volname, int host_id)
int pfs_umount(const char *volname)
int pfs_mount_growfs(const char *volname)
int pfs_creat(const char *volpath, mode_t mode)
int pfs_open(const char *volpath, int flags, mode_t mode)
int pfs_close(int fd)
ssize_t pfs_read(int fd, void *buf, size_t len)
ssize_t pfs_write(int fd, const void *buf, size_t len)
off_t  pfs_lseek(int fd, off_t offset, int whence)
```
Architecture of PolarFS -- Components

- 2. PolarSwitch

- Proxy Daemon
  - Run on Compute Nodes

- Remap I/O Requests
  - Cache Mapping Metadata

- Send Remapped I/O Requests
  - To one or more Storage Node(s)

- Write Data to Primary Replica
- 3. **ChunkServer**

- **Run on Storage Nodes**
- **Each ChunkServer**
  - Manage One NVMe SSD Device
  - Bound to One CPU Core

- **Inner Chunk allocation**
  - Use WAL for A and D

- **Chunk Replication**
  - Parallel Raft Consensus Protocol
- 4. PolarCtrl

- Control Plane of PolarFS Cluster
  - ChunkServer Membership
  - Chunk Arrangement and Load Balancing
  - Metadata Center (push to PolarSwitch)
  - Monitor System Health
  - I/O Performance for Each Volume (Latency, IOPS, etc.)
  - Scheduling Data CRC Check Periodically

- Not on critical I/O Path
  - Most I/Os Only Involve PolarSwitch and ChunkServer
Put Them Together

- **A Write I/O Flow**

- All in Userspace

1. A request by Database
2. Libpfs make IO requests to PolarSwitch
3. PolarSwitch RDMA to ChunkServer
4. ChunkServer write SSD by SPDK
5. Main replica RDMA to Follower Node

- **Ultra-low Latency**

- No Syscall
- No Context Switch
- No Useless Data Copy
Failure Resilient of PolarFS -- ParallelRaft

- **ParallelRaft Motivation**

  - DB tolerate Out-of-Order I/O completion
  - Normal Storage Semantics

- **Loose Raft Sequential Restriction**

  - Commit and Apply out-of-orderly

- **Keep Logical Serialization**

  - Overlapped modification consistent on all node
Failure Resilient of PolarFS -- ParallelRaft

- Key Problem
- Missing Entry Conflict or Not??
- Solution: Look Behind Buffer

- Entry
  - Seq id 798
  - Page 1
- Entry
  - Seq id 799
  - Page 2
- Entry
  - Seq id 800
  - Page 3
  - LBF1 Page2 LBA
  - LBF2 Page1 LBA

- Missing Entry
  - Seq id 800
  - Page 3
  - LBF1 Page2 LBA
  - LBF2 Page1 LBA

- Look Behind Buffer
  Stores N Previous Entries’s LBA

- Entry
  - Seq id
  - Page
  - Look Behind Buffer 1
  - Look Behind Buffer N

Now Entry Know if conflict with the Missing Entry by
Look Behind Buffer (LBA: logical block address)

Entry 6 does NOT know if conflict with Entry 4/5 so CANNOT commit
Evaluation -- Database on PolarFS

- Three Systems
  1. Alibaba MySQL cloud service RDS
  2. POLARDB on PolarFS
  3. POLARDB on local Ext4

- OLTP workloads
  - Read-only, Write-only (update : delete : insert = 2:1:1), R/W-mixed (read : write = 7:2)
  - Test data size : 500GB,
  - Tables : 250,
  - Records of each table : 8,500,000

POLARDB on PolarFS
Comparable to Local SSD
With 3 Replicas over Nodes
POLARDB-X: shared-nothing layer for scaling out

- Stateless computation nodes
- SQL optimizer and distributed SQL execution engine
- Distributed storage engine
- Cross-AZ, LSM-tree based tiered storage
- Shared storage
- Distributed file system

Distributed relational database service

- SQL Parser
- SQL Optimizer
- SQL Router
- Transaction Manager

- POLARDB
  - plan executor
  - transaction service
  - X-engine (SIGMOD 2019)

- POLARDB
  - plan executor
  - transaction service
  - X-engine (SIGMOD 2019)

- POLARDB
  - plan executor
  - transaction service
  - X-engine (SIGMOD 2019)

- PolarFS
- PolarStore
Background

**Agility**
- Arbitrarily Join
- Arbitrarily Filter

**Real-time**
- READ COMMITTED
- 10M Records/s Insert

**Accuracy**
- 100%

**OLAP Requirements at Alibaba & Alibaba Cloud**
- **High concurrency**
  - 100K QPS
  - 10K Clients
- **Low Latency**
  - 95% 50ms
- **High availability**
  - 99.999%
Table Partitions

- Hash + List two level partitions for fact table
- Dimension table
- Partition pruning on both Hash and List partitions
Query is executed on both baseline data and incremental data.

Multi versioned delete bitsets are used to support snapshot read and delete/update.

Baseline and incremental data are merged in background.
Index for computing

- Indexes on all columns
- High performance for ad-hoc queries
- K-ways merge for row-id sets.
- Runtime indexes selection
- Support complex-typed data
- Full text and JSON index

```
SELECT ... From t WHERE
(NAME='Bob' AND (CITY = 'Hangzhou' OR CITY = 'Shanghai') AND
(SEX != Female) ) OR ( JSON_EXTRACT(ATTR,'time') > 0
OR ANN(VEC, [1,1,1,1], 2) )
```
Hybrid row-column store

Multi-dimensional analysis
✓ Any column Join
✓ Complex long computing tasks, ETL

Detail Query
✓ 1000+ columns extra wide table
✓ Semi-structured, large fields (JSON/ARRAY, etc.)

Real-time RW
✓ Live updates
✓ 6 million TPS
✓ 10000+ QPS
## Real Workload Evaluation

<table>
<thead>
<tr>
<th>Query Type</th>
<th>Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full Scan (Q1)</td>
<td><code>SELECT * FROM orders ORDER BY o trade time LIMIT 10</code></td>
</tr>
</tbody>
</table>
| Point Lookup (Q2)         | `SELECT * FROM orders
AND o trade prize BETWEEN 50 AND 60 AND o seller id=9999 LIMIT 1000` |
| Multi-table Join (Q3)     | `SELECT o seller id, SUM(o trade prize) AS c FROM orders
JOIN user ON orders.o user id = user.u id
GROUP BY o seller id ORDER BY c DESC LIMIT 10;`                   |

- Eight Physical Machines
- Intel Xeon Platinum 8163 CPU, @ 2.5 GHz
- 300GB main memory and 3TB SSD
- 10Gbps Ethernet network
- 4 coordinators
- 4 write nodes, and 32 read nodes
- Workloads
- 1TB and 10 TB
TPC-H Evaluation

- Pipeline-process
- All-column index
- Hybrid row-column storage
- Runtime cost-based index path selection
- K-ways merging and composite predicates pushdown
- Vectorized execution engine and optimized codeGen
Outline

Background

POLARDB

AnalyticDB

Self-Driving Database Platform

Conclusion
## Components of SDDP

### Self-Driving Database Platform (Portal)

<table>
<thead>
<tr>
<th><strong>Tuning</strong></th>
<th><strong>Auto admin</strong></th>
<th><strong>Protection</strong></th>
<th><strong>Elasticity</strong></th>
<th><strong>Operations</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimized SQL</td>
<td>Anomaly detection</td>
<td>Threat detection</td>
<td>Resource prediction</td>
<td>Resource management</td>
</tr>
<tr>
<td>Memory opt</td>
<td>Monitoring</td>
<td>Security proxy</td>
<td>Alert</td>
<td>Config upgrades</td>
</tr>
<tr>
<td>Config optimization</td>
<td>Diagnostics</td>
<td>Patch</td>
<td>Scheduling</td>
<td>HA, disaster recovery</td>
</tr>
<tr>
<td>Whole stage opt</td>
<td>Restore</td>
<td>Identification</td>
<td></td>
<td>Backup recovery</td>
</tr>
</tbody>
</table>

### Machine learning

Alibaba Self-Driving Database Platform (SDDP) provides cloud databases with automatic operation and maintenance.
Key features of SDDP

Knobs tuning
- e.g., OtterTune

Hot/cold separation
- Immutable Memtable
- Memtable
- Memory
- Persistent storage

ClouDBench
- Workload Generator
- Workload Replay

Anomaly detection
- Slow SQL → Throttling

NL2SQL
- Human Language
  - SQL
  - DBMS

DB Design recommendation
- Auto Index/sharding
SDDP: Self-Driving Database Platform

- **DB**
- **Model Prediction**
- **Control system**
  - Cold/hot model
  - Index model
  - Memory model
- **System metrics**
- **SQL & DB Metric**
- **Parameter update**
- **DB Advisor**
  - SQL & DB metrics
  - Autotuning
  - Slow SQL
  - Space analysis

**SDDP**:
- Resource scheduling
- SQL collection
- Anomaly detection
iBTune - individualized Buffer Tuning: Motivation

The memory uses at Alibaba product environment. Buffer pool is the largest memory consumer.

Table 1: Usage of different memory pools

<table>
<thead>
<tr>
<th>Memory Pool</th>
<th>buffer pool</th>
<th>insert buffer</th>
<th>log buffer</th>
<th>join buffer</th>
<th>key buffer</th>
<th>read buffer</th>
<th>sort buffer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg. Size</td>
<td>29609.98M</td>
<td>8.00M</td>
<td>200.00M</td>
<td>0.13M</td>
<td>8.00M</td>
<td>0.13M</td>
<td>1.25M</td>
</tr>
<tr>
<td>Percent</td>
<td>99.27%</td>
<td>0.03%</td>
<td>0.67%</td>
<td>0.00%</td>
<td>0.03%</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
</tbody>
</table>

Memory is bottleneck among the resources.
Reduce memory (buffer pool) while guaranteeing SLA (response time).

- DBA manually uses a small number of BP sizes (10 configurations in our case).
- Each instance’s BP size might be different as the query workload is different.
- Manual tuning is not scalable for large cloud databases since each instance has different BP size.

**iBTune: Individualized Buffer Tuning for Largescale Cloud Databases**
tolerate miss ratio 
\(t_{\text{miss\_ratio}}\)

Practical function
\[
\frac{\log(mr_{\text{target}}) - \log(mr_{\text{cur}})}{\log(hp_{\text{pi\_target}}) - \log(hp_{\text{cur}})} \approx -\alpha_t
\]

Pairwise DNN

Apply new BP size

DB instance

DB instance

Guarantee SLA

F1\(_{(t_{\text{miss\_ratio}})}\) = New BP size

F2\(_{(t_{\text{miss\_ratio}})}\) = Response time

Safe Response time (SLA)
iBTune: individualized Buffer Tuning (VLDB 2019)

- iBTune: deployed on production systems with more than 10,000 instances, memory saving of >20TB
10 representative instances. The memory saving ranges from 50% to 10%, which strongly supports that a single number does not fit all. Instance 1 has a large increase in RT after the adjustment. We find that there is one query that consumes 99.97% of the total response time. The lookup value in WHERE condition changes for this query.
Outline

Background

POLARDB

AnalyticDB

Self-Driving Database Platform

Conclusion
Recent publications

  [Link](http://sigmod2019.org/sigmod_industry_list)

  [Link](https://sigmod2018.org/sigmod_industrial_list.shtml)

  [Link](www.vldb.org/pvldb/vol11/p1849-cao.pdf)

- [VLDB’14] Realization of the Low Cost and High Performance MySQL Cloud Database.
  [Link](www.vldb.org/pvldb/vol7/p1742-alibaba.pdf)

  [Link](https://www.vldb.org/2019/?papers-industrial)

- [VLDB’19] iBTune: Individualized Buffer Tuning for Large scale Cloud Databases
  [Link](https://www.vldb.org/2019/?papers-industrial)
Conclusion and Future Work

- **Cloud-native:** more elasticity, high availability, and excellent scalability
- HTAP capability
- Multi-model
- New hardware support (software-hardware co-design)
- Security
- Self-driving databases
Thanks
Database Requirement is Changing

- Several Giant Business → A Wide Range of Enterprises

- In China, 70% of uprising enterprises are hampered by such Data Challenges.

<table>
<thead>
<tr>
<th>High Cost</th>
<th>A million $ license? Professional engineers?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weak Ability</td>
<td>Data Backup? Mining? Trouble Shooting?</td>
</tr>
<tr>
<td>Data Explosive</td>
<td>Hard to store, analyze, utilize.</td>
</tr>
</tbody>
</table>

- Everything is Online, so does Database

**Data**
- Generate
- Gather
- Extract
- ...

**Computing**
- Training
- Biding
- Anti-fraud
- ...

**Application**
- Working
- Social
- Entertainment
- ...

**Consumer**
- Global
- 7 X 24
- Digitalized
- ...
Cloud Database Features

- **More flexible**: marketing activity, hotspot
- **More economical**: small profits, globalization
- **More efficient**: sensitive to customer loss
- **More agile**: time & opportunity is money

Elastic

Low Cost

Performance

Continuous Solution
POLARDB Architecture

- **Excellent Elasticity**
  - 2core vCPU upgrades to 32core in < 5mins
  - 2 nodes scales out to 4 nodes < 5mins

- **Cost reduction**

- **Serverless on-demand billing**

- Distributed shared storage
  - Database Files
  - Redo Log File
Failure Resilient of PolarFS -- ParallelRaft

- **Why develop a variant of Raft**
- **At First Choose Raft**
  - For its implementation simplicity
- **Found not suitable for PolarFS**
  - Test shows scale problem under our highly concurrent environment

- **Sequential Commitment Limitation**
  - May get stuck when some entry is slow

From: https://raft.github.io/
Failure Resilient of PolarFS -- ParallelRaft

- Selecting a New Leader

- Candidate might lack committed entries
  - Holes permitted in the log

- Introduce an additional merge stage
  - Guarantee Leader Completion Property

- CP is CheckPoint
  - Logs old than CP are deletable

Other Details please refer the Paper
- Storage can play an important role in Database Enhancement
  - QPS, TPS, Capacity, RO instance extension, Fast Backup

- PolarFS delivers extreme performance with high reliability for Cloud Database
  - Adopting Optane, NVMe SSD, RDMA, OS-bypass and zero-copy techniques, ParallelRaft
  - Allowing 3-replica write latency comparable to single replica on the local SSD

- Sharing Storage Architecture, POSIX-like Userspace interfaces
  - Easily port DB: Minor modifications to get whole database database improvement

- Future: New hardware and More DB optimizations
  - NVM, FPGA, etc.
  - Parallel Query, SQL Offloading
High Availability: Cross AZ-Cross Region

Gateway/Proxy

Backup AZ-A  |  Master AZ-B  |  Backup AZ-C

Parallel Raft/Paxos for Binlog

Master: Shanghai (Three AZs)

User Applications

mydb.mysql.rds.aliyuncs.com

Gateway/Proxy

Backup  |  Master

Backup: Beijing (One AZ)

Binlog Synchronization

DTS (Data Transmission Service)
Why Cloud Native?

A typical use-case: on-demand usage and elastic billing

Computation resource needs

Online resource

Offline resource

New products

Annual sales event

Time

- Yearly subscription---50% off
- On demand usage and elastic billing – minute-level
ParallelRaft vs. Raft

- FIO Test in Our Concurrent Environment
- Performance Gap appear under large I/O depth

The IOPS of Raft drops after I/O depth 8, ParallelRaft keep a steady high. ParallelRaft helps PolarFS to get high performance under heavy workloads.
Evaluation – PolarFS Latency

- PolarFS and CephFS both have 3 replicas; Local Ext4 one replica

The Average Write Latency Compared to Local Ext4

Random:  
PolarFS 1.6 to 4.7 times slower  
CephFS 6.5 to 75 times slower

Sequential:  
PolarFS 1.6 to 4.8 times slower  
CephFS 6.3 to 56 times slower
Evaluation – PolarFS Throughput

4KB Random Write

Write / Read throughput of Ext4 and PolarFS are 4/7.7, 4.4/5.1 higher than CephFS
iBTune - Preliminary Attempt

Buffer pool (BP) size is correlated to miss ratio

**Response Time**

![Response Time Graph]

**CPU usage**

![CPU Usage Graph]

**hit ratio**

![Hit Ratio Graph]

- **Challenge:** Heuristic method (such as shrinking 10% each time) does not work, since we have to try many times, which makes the system unstable and is unacceptable for mission-critical applications.

**Intuition:**

- Calculate BP based on hit ratio (miss ratio) to avoid restarting system multiple times
- Confirm whether the BP size meets the requirement of SLA
System halting avoidance

Based on X-Paxos: high availability protocol via binlog implementation at Alibaba

Switch master and backup after the backup node recovers

1. Adjust backup node's BP size
   It leads to 10~20 sec system halting to backup node, without affecting system service.

2. Switch

3. Monitor the new master for 24 hours

4. If the new master is abnormal, i.e., the number of slow SQLs increases, rollback will be triggered.

5. If the new master works fine during the following 24 hours, backup nodes' BP will be adjusted.

Rollback means switching between master and backup nodes: since the backup node's BP is still the old one, rollback restores the original status.
Evolvement of database systems

Structured Data
- RDBMS [SQL+OLTP]
- Data warehouse
- Data Cube [ETL+OLAP]

Structured Data

Heterogeneous Data
- Structured Data
- Graph
- Time Series
- Vector
- Spatial Data
- Text

RDBMS

NoSQL/NewSQL DB
[Multi-Model + HTAP]
Hardware-software co-design

RDMA

Open-Channel SSD

NVM 3D XPoint

GPU/FPGA
Multi-Model Database System

Query Interface
- SQL
- Put/Get
- SPARQL
- Doc QL
- GQL
- ...

DB Engine

Data format
- Structured data
- Graph
- Vector
- Doc
- Spatial-Temporal
- Non-structured data
- Time series
HTAP: Hybrid Transaction and Analytical Processing

Row Store
C1 | C2 | C3 | C4
---|---|---|---

Columnar Store
C1 | C2 | C3 | C4
---|---|---|---

Hybrid Store
C1 | C2 | C3 | C4
---|---|---|---

OLTP
- Real time update, point query
- Compression, analytics

OLAP
- Compression, read only, Complex queries and scan
- Updates

Hybrid: HTAP
Data Security

HTTPS

File System

Database

Database

Database

Raw Dev

Raw Dev

Raw Dev

BYOK

KMS

Transparent Data Encryption

Data at Rest Encryption

Hardware Security Module

SSL

TLS