AnalyticDB: Real-time OLAP Database System at Alibaba Cloud

Chaoqun Zhan, Maomeng Su, Chuangxian Wei, Xiaoqiang Peng, Liang Lin, Sheng Wang, Zhe Chen, Feifei Li, Yue Pan, Fang Zheng, Chengliang Chai
Alibaba Inc.

Presenter: Liang Lin, Alibaba Database BU
1.1 Background: OLAP system evolution

- ~2008
  - Oracle RAC
- 2009
  - Greenplum
- 2011
  - HBase
  - MySQL Sharding
  - Hadoop
- 2012
  - AnalyticDB 1.0
- 2016
  - AnalyticDB 2.0

![Diagram](image)

- High concurrency
- Consistency
- Agility
- Accuracy

- Volume (~PB)
- High concurrency
- Low Latency
- Accuracy

- Volume (~PB)
- High concurrency
- Agility
- Low Latency
- Accuracy
- High availability

- Volume (~100PB)
- High concurrency
- Agility
- Low Latency
- High availability
- Accuracy
- Realtime RW

- High concurrency
- Volume
- High availability (leader node)
- Realtime Write

- Consistency (offline/online)
- Agility (Cube)
- ACID

- Consistency (offline/online)
- Realtime Write
- ACID

- ACID

High concurrency: ~1000 QPS (Complex Query)  Volume: 100TB+  Realtime Write: 10M Records/s
1.2 Background: Design Challenge

**Agility**
- Arbitrarily Join
- Arbitrarily Filter

**Real-time**
- Read Committed
- 10M Records/s Insert

**Concurrency**
- 100K QPS
- 10K Clients

**Accuracy**
- 100%

**High Availability**
- 99.999%

**Low Latency**
- 95% 50ms

**AnalyticDB**

755M+ Active Users

5+ PB Max instance
2. AnalyticDB: Architecture

![AnalyticDB Architecture Diagram]
3.1 Storage System Overview

- **Read/Write decoupling**
  - High-throughput write
  - High-throughput query

- **High Scalability**
  - Scale transparently
  - up to 1024 nodes/DB

- **High Availability**
  - Fault-tolerant
  - Self-healing
  - All replicas are active

- **Strong Consistency**
  - Real-time read

- **Async Data Builder**
  - All-Column Index Builder
  - Re-partition Builder
  - Re-Clustered

---

**Coordinator**

- R/W Manager
- GTM
- Builder
- Metadata
- ...

**Write Node**

- Master Thread
- Reader Cache
- Online/Offline
- Writer Process
- Baseline Thread
- GroupCommit Pool

**Read Node**

- Table Engine
  - Realtime Engine
  - Full data Engine
  - Detail
  - Index
  - Version

- Cache System

**Tiered File System**

- Existing data Vol
- Inc Data Vol
- Updated Data Vol

- Fuxi RM & Pangu DFS

---

- Read/Write decoupling
- High-throughput write
- High-throughput query

- High Scalability
- Scale transparently
- up to 1024 nodes/DB

- High Availability
- Fault-tolerant
- Self-healing
- All replicas are active

- Strong Consistency
- Real-time read

- Async Data Builder
- All-Column Index Builder
- Re-partition Builder
- Re-Clustered
3.1 Storage System: Lambda, Multi-Version

- **Lambda architecture**
  - Support fast insert
  - Block index for Incremental data
  - Column index for baseline data

- **Multi-Version**
  - Mark for delete with bitsets
  - Copy-on-write for dedup
  - Support snapshot read
  - Support delete and update

- **Merge**
  - Incremental index build
  - Time/size based merge
  - Merge in background
  - Data vacuum

### Query Execution (Version = N)

- **MINUS**
  - Deleted Bitset
    - \( V_n \) 010001
    - \( V_2 \) 010000
    - \( V_1 \) 000000

- **UNION**
  - Baseline Data
  - Incremental Data

- **Insert/Delete/Update**
3.2 Index Computing

- **All-columns Indexing**
  - Indices built for all columns (automatic/optional)
  - Runtime index selection
- **High performance ad-hoc**
  - Index Computing
  - K-way merge for indexing results.
- **Various Data Type**
  - int/varchar/time/date/…
  - Full text and JSON

**Example:**

```sql
SELECT ... From t WHERE (name = "Bob"
  AND gender != "female"
  AND (CITY = "Hangzhou" OR CITY = "Shanghai")
  OR JSON_EXTRACT(ATTR,"time") > 0
  OR ANN(VEC, [1,1,1,1], 2))
```

![Diagram showing index computing and query example](image)
3.3 Hybrid Row-column Storage

**Multi-dimensional Analysis**
- Any column join
- Complex long computing tasks, ETL

**Complex Query**
- 1000+ columns
- Extremely wide table
- Semi-structured, large fields

**Real-time Read/Write**
- Live updates
- 10 million TPS
- 10K+ QPS
3.4 Execution

- Pipelining
- Codegen
- Mixed workload
- Vectorized execution
- Memory pool/cache
3.5 Optimization

Efficient Real-time Sampling
- More data meta info, better execution plan
- Index-based join and aggregation
- Less data read, less computation

Optimizer
- Data Sources Capabilities
- Dynamic Programming
- Source-specific Planner
- Connector Manager

Capability Registration
Executable API Calls
Data Location

Storage Engine

Join push-down
Predicate push-down
Join Reordering

SELECT ...
FROM a
JOIN b
  ON a.id = b.id
JOIN c
  ON b.id = c.id
JOIN d
  ON c.id = d.id
JOIN e
  ON d.id = e.id
WHERE ...

Bushy Tree Join Order

Optimized Join Order
### 4. Experiment: Setup

<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>
<tr>
<td>Point Lookup (Q2)</td>
<td><code>SELECT * FROM orders WHERE o trade time BETWEEN '2018-11-13 15:15:21' AND '2018-11-13 16:15:21' AND o trade prize BETWEEN 50 AND 60 AND o seller id=9999 LIMIT 1000</code></td>
</tr>
<tr>
<td>Multi-table Join (Q3)</td>
<td><code>SELECT o seller id, SUM(o trade prize) AS c FROM orders JOIN user ON orders.o user id = user.u id WHERE u age=10 AND o trade time BETWEEN '2018-11-13 15:15:21' AND '2018-11-13 16:15:21' GROUP BY o seller id ORDER BY c DESC LIMIT 10;</code></td>
</tr>
</tbody>
</table>

- Eight Physical Machines
  - Intel Xeon Platinum 8163 CPU, @ 2.5 GHz
  - 300GB main memory and 3TB SSD
  - 10Gbps Ethernet network

- Deployment of AnalyticDB
  - 4 coordinators
  - 4 write nodes, and 32 read nodes
  - Workloads
  - 1TB and 10 TB
4. Experiment: Latency Analysis

- All three types of queries are completed within seconds.
- Query performance of AnalyticDB is slightly impacted by the table size.
4. Experiment: Latency Comparison

![Graphs showing latency comparison for different concurrency numbers for PDB, ADB, Druid, and Spark.](image)
4. Experiment: TPCH Evaluation

Advantages:
- 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
5. Future Work

- Separation storage/computation
- HTAP
- Multi-Master
- Cross AZ HA
- Serverless
- Separation storage/computation/memory
- ML Optimizer
Q & A