2023 • 上海
SkyWalking Summit
演讲主题

高洪涛

Tetrate创始工程师 、 Apache SkyWalking核心贡献者

BanyanDB
Cloud Native Observability Database
01. Observability & TSDB

02. Data Access

03. Glance of BanyanDB

04. Next Plan
Observability & TSDB
Three Pillars

- **Metrics**
  - Aggregatable
  - Aggregatable events e.g. rollups

- **Tracing**
  - Request scoped

- **Logging**
  - Events
  - Request-scoped, aggregatable events

Low volume vs. High volume
More Models

Observability

Measure:
- Error/Exception
- Profiles
- Metrics

Stream:
- Traces
- Logs
- Events
Tags & Fields

metric_name (label: __name__)
Fields: single-valued or multiple values
Timestamp: epoch
LabelSet (Tags)

samples (metric_name String, labels String, timestamp Int64, value Float64)
ORDER BY (metric_name, labels, timestamp)
High Cardinality

How Quickly Does Cardinality Grow?

VIRTUAL-MACHINE BASED ENVIRONMENT

- 20 HTTP routes
- 5 HTTP status code types
- 5 services
- 300 Vms

CARDINALITY

150 thousand possible unique time series

CLOUD-NATIVE ENVIRONMENT

- 20 HTTP routes
- 5 HTTP status codes types
- 5 services
- 12,000 pods (40x VMs)
- 25 experiments / versions

CARDINALITY

150 million possible unique time series
Glance of BanyanDB
Logical Data Model

Pros:
- Reduce High Cardinality
- Validation
- Querying
- Integration

Cons:
- Complexity
- Overhead
- Maintenance
Physical Data Model

Storage Node
- Raft-based Metadata (ETCD)
- Group 1
- Group 2
- Group 3
- Group 4
- Group 5
- Group 6

1:N

Shard
- Series Metadata
- Segment 2023-05-29
- Segment 2023-05-30

Segment
- Global Index (e.g. Trace ID)
- Block 2023-05-29 21:00
- Block 2023-05-29 22:00

Block
- Data block
- Index block
Distributed
Data Access
Rum Conjecture

- Read Optimized Storage Systems
- Memory Optimized Storage Systems
- Update Optimized Storage Systems

Update

Memory

Read
BanyanDB Access Method

- Vertical write
- Horizontal read
- Insertions >> lookups
- Old data is less likely to be access
Compression based on Series

{component="printer",location="f2c16",level="error"} "Printing is not supported by this printer"

Label key/values hashed to form Stream ID: 3b2cea09797978fc

The log entry is added to a "chunk"

Additional log messages with the same labels are added to the same "chunk":

{component="printer",location="f2c16",level="error"} "Out of paper"
{component="printer",location="f2c16",level="error"} "Too much paper"

Chunks are filled then compressed and stored:

Printing is not supported by this printer
Out of paper
Too much paper

A separate and small index is kept to lookup chunks

Different label keys or values will hash to a different stream and different chunk:

{component="printer",location="f2c16",level="info"} "Consider the environment before printing this log message"

fd9a709ddf43a93a
Enhanced Gorilla

- **Facebook Gorilla:**
  - Timestamp: fixed interval => derived timestamp
  - Value: XOR
- Compress big chunk

<table>
<thead>
<tr>
<th>N-2 timestamp</th>
<th>02:00:00</th>
<th>-</th>
</tr>
</thead>
<tbody>
<tr>
<td>N-1 timestamp</td>
<td>02:01:02</td>
<td>Delta: 62</td>
</tr>
<tr>
<td>Timestamp</td>
<td>02:02:02</td>
<td>Delta: 60</td>
</tr>
</tbody>
</table>

- **Compressed Data:**
  - Header:
    - March 24, 2015 02:00:00
  - Bit length:
    - 64
    - 14
    - 64
  - Value:
    - 12
  - Value (XOR):
    - '0'
    - '0'
    - '0'
    - '11': 1, '1': 1

- **Previous Value:**
  - Value: 24.0
  - XOR: -

- **Delta of Deltas:**
  - -2

- **11 leading zeros, # of meaningful bits is 1**
Compression Ratio

- **Metric**: Low compression ratio
- **Log**: Moderate compression ratio
- **Trace**: High compression ratio
What’s Next
Column Storage

Stream Block Store

Primary Index

Tag Index

Meta

Minutes

Counts

Section Counts

Time Stamp

Tag Family 0

Tag Family 1
Balanced binary tree

Leaf nodes

Row IDs (FoR)

1-120

1-50

1-10

10-50

50-120

50-80

80-120

1024 records
Values (RLE)
Row IDs Idx (FoR)

3
43
56
...

3
43
56
85
89
154
189
...

RLE = Run length encoded
FoR = Frame of reference encoded
BanyanDB, as an observability database, aims to ingest, analyze and store Metrics, Tracing and Logging data. It's designed to handle observability data generated by observability platform and APM system, like Apache SkyWalking etc.

Introduction

BanyanDB, as an observability database, aims to ingest, analyze and store Metrics, Tracing, and Logging data. It's designed to handle observability data generated by Apache SkyWalking. Before BanyanDB emerges, the Databases that SkyWalking adopted are not ideal for the APM data model, especially for saving tracing and logging data. Consequently, there's room to improve the performance and resource usage based on the nature of SkyWalking data patterns.
欢迎提问交流
（仅限2位提问）
2023 · SkyWalking Summit

感谢您的观看