Messaging, storage, or both

The real-time story of Apache Pulsar and Apache DistributedLog

Matteo Merli — Sijie Guo
Apache BookKeeper

- A replicated log storage
- Low-latency durable writes
- Simple repeatable read consistency
- Highly available
- Store many logs per node
- I/O Isolation
Apache Pulsar

table

| fast, durable, flexible pub/sub messaging |
What is Apache Pulsar?

**Durability**
Data replicated and synced to disk

**Geo-replication**
Out of box support for geographically distributed applications

**Unified messaging model**
Support both Topic & Queue semantic in a single model

**Ordering**
Guaranteed ordering

**Multi-tenancy**
A single cluster can support many tenants and use cases

**High throughput**
Can reach 1.8 M messages/s in a single partition

**Delivery Guarantees**
At least once, at most once and effectively once

**Low Latency**
Low publish latency of 5ms at 99pct

**Highly scalable**
Can support millions of topics
Usage of Pulsar

- In production for 3+ years at Yahoo
- Powering critical products like:
  - Yahoo Mail, Yahoo Finance, Gemini Ads, Flickr and Sherpa (NoSQL database)
- 80+ tenants
- 2.3 Million topics
- 100 B messages / day
- Full-mesh replication in 8 data-centers
Architecture view

- Separate layers between brokers bookies
  - Broker and bookies can be added independently
  - Traffic can be shifted very quickly across brokers
  - New bookies will ramp up on traffic quickly
Flexible message model

Support both topic and queue semantics in a unified topic concept
Multi-Tenancy

- Authentication / Authorization / Namespaces / Admin APIs
- I/O Isolations between writes and reads
  - Provided by BookKeeper - Ensure readers draining backlog won’t affect publishers
- **Soft isolation**
  - Storage quotas – flow-control – back-pressure – rate limiting
- **Hardware isolation**
  - Constrain some tenants on a subset of brokers or bookies
Geo-Replication

- Scalable asynchronous replication
- Integrated in the broker message flow
- Simple configuration to add/remove regions
Pulsar client library

- Java – C++ – Python – WebSocket APIs
- Partitioned topics
- Transparent batching of messages
- Compression
- TLS encryption
- End-to-end encryption
- Individual and cumulative acknowledgment
Pulsar – Conclusion

- A fast durable, distributed pub/sub messaging
  - Flexible Traditional Messaging: Queuing and Pub/Sub
  - Focus on message dispatch and consumption
  - Remove data as soon as possible if they are not needed
- It is backed by a scalable log store
  - durable message store, zero data loss
  - allow rewinding / reprocessing messages for backfill, bootstrap systems and stream computing
  - It carries all the fantastic features from the log store.
Apache DistributedLog

a highly scalable log stream store
Problems

- Real-Time Infrastructure is fragmented
  - Fragmented Components
    - Kestrel, Kafka, BookKeeper, Scribe, Mysql, ...
  - Maintenance Overhead
    - Software Components - backend, clients, and interop with the rest of Twitter stack
    - Manageability and Supportability - deployment, upgrades, hardware maintenance and optimization
  - Technical know-how
A Shared Log Infrastructure
Benefits

• Log/Stream is a fundamental system abstraction. It is everywhere.

• Reuse infrastructure for different use cases. Save costs.

• Simplify developing replicated services.

• Make replicated services ‘stateless’, easier to run in containerized environments.

• Independent scalability
What is Apache DistributedLog?

- A high performance replicated log store
- It carries the fantastic features from Apache BookKeeper
  - Low latency replication with strong durability
  - Simple repeatable read consistency
  - High write/read availability
- I/O isolation, resource-aware data placement policies, fast replica repair
What is Apache DistributedLog?

- Extending segments to streams
  - Infinite Stream Abstraction
    - Infinite storage + Fast tailing facility
  - Support different data retention and segment rolling policies
  - Tunable read/write pipelines: batch, compression, caching, ...
  - Efficient fan-in and fan-out
- Geo-replication
Stream - Write & Read

Ledger 1

Ledger x

Catchup Reads
(rewind to any positions)

Tailing Reads
(close to head of stream)

New Records added here
Data distribution

Logical stream view

Segment 1  Segment 2  Segment 3  Segment 4  Segment X

Physical segments distribution

Segment 1  Segment 1  Segment 1  Segment 2  Segment 2  Segment 2  Segment X

Bookie 1  Bookie 2  Bookie 3  Bookie N
Fast Repair

Logical stream view

Segment 1  Segment 2  Segment 3  Segment 4  Segment X

Physical segments distribution

Segment 1  Segment 2  Segment X  Segment 1  Segment 2  Segment X  Segment 1

Bookie 1  Bookie 2  Bookie 3  Bookie N

streamlio
Low Latency Write-Read

Segment (Ledger)
Bookie 1
Bookie 2
Bookie 3
Bookie 4
Bookie 5

Writer
Add entries

Write quorum
Ack quorum

Other bookies in the cluster
Bookie x

Speculative Reads
Potential replacement of unhealthy bookie

Reader
Latency and Availability

- Parallel Replication => Mask machine failures
- Flexible Replication Settings
  - Ensemble => Increase write bandwidth for a single stream
  - Ack Quorum => Tradeoff Latency
- Ensemble Change => High write availability
- Speculative Reads => High read availability, Low read latency
Geo Replication

Region Aware Placement Policy

- Client
- Zk
- Zk
- Zk
- us-west
- us-central
- us-east
Datacenter Failure

Region Aware Placement Policy

us-west

us-central

us-east
DistributedLog – Conclusion

• A highly scalable log stream store
  • Infinite Stream Abstraction
  • A storage abstraction combining storing past/historic data and propagating/streaming future data
  • Focus on storing data (e.g. replication, durability) and streaming data (fast tailing).

• It can be used for
  • Streaming data that requires strong order
  • Building a message broker like Apache Pulsar
Message, storage, or both?
Messaging and storage

- Messaging
  - Propagating future data: waiting for new messages to arrive
  - Message dispatch and consumption

- Storage
  - Storing past (historic) data
  - Query and reprocessing
  - Make sure data is durably stored and zero data loss
Pulsar + BookKeeper

Producer (X) → Topic (T) → Subscription (A) → Consumer (A1)
Producer (Y) → Topic (T) → Subscription (B) → Consumer (B1) → Consumer (B2) → Consumer (B3)

Apache DistributedLog/BookKeeper

log streams
Pulsar + BookKeeper

Producer (X) -> Topic (T) -> Subscription (A) -> Consumer (A1)

Producer (Y) -> Topic (T) -> Subscription (B) -> Consumer (B1) -> Consumer (B2) -> Consumer (B3)

Apache DistributedLog/BookKeeper
Real-Time Solution

• The requirements for a unified real-time solution
  • The ability to process the past
  • The ability to process the future
  • The ability to keep intermediate states for future queries

• Messaging and Storage are two sides of a coin
  • Pulsar: fast durable messaging
  • DistributedLog/BookKeeper: highly scalable log store
Real-Time Solution
Curious to Learn More?

- Apache Pulsar: http://pulsar.incubator.apache.org
- Apache DistributedLog: http://bookkeeper.apache.org/distributedlog
- Apache BookKeeper: http://bookkeeper.apache.org
- Follow Us @apache_pulsar @asfbookkeeper @distributedlog
Curious to Learn More?

- **Messaging, Storage, or Both**: [https://streaml.io/blog/messaging-storage-or-both/](https://streaml.io/blog/messaging-storage-or-both/)


- **Introduction to Apache Pulsar**: [https://streaml.io/blog/intro-to-pulsar/](https://streaml.io/blog/intro-to-pulsar/)

Curious to learn more about Streamlio?

- Streamlio: https://streaml.io
- Sandbox Preview: https://streaml.io/docs/getting-started
- Learn slack channel: https://learn-streamlio.slack.com