Experience of Using Hudi in ByteDance’s Recommendation System

Ziyue Guan
Translated by Y Ethan Guo
Agenda

01 Scenario Requirements
02 Design Decisions
03 Functionality Support
04 Performance Tuning
05 Future Work
01 Scenario Requirements

- BigTable CDC
- Feature Engineering
Scenario Requirements

- **Application**: Event Tracking, Impression, External Data Source
- **Recommendation Engine**: Trigger, Matching, Coarse Ranking, Fine Ranking, Re-Ranking
- **Data Service**: Forward Index, Inverted Index, Profile, Vector Database, Parameter Service
- **Near real-time Processing**: Incremental database building, State Change, Statistics Service, Feature Generation, Model Training
- **State Storage**: Tbase
- **Offline/Real-time Storage**: Sample Data, User Logs, Article Data, Offline Mining

User Logs/Other Logs

CDC
Scenario Requirements

**BigTable CDC**
- CDC for flat-wide table database
- Provide efficient OLAP query
- Provide online compatible data sync
- Irregular data (size, format)
- Requirements are diverse

**Feature Engineering**
- The splicing of instance and label
- Provide access to efficient IO for model pruning
- High-dimensional complex data (ten thousands of columns, nested types, sparse)
- High throughput and near real-time write (100 GB/s)
- EB-level storage
02 Design Decisions

- Multiple data lake engines
- MOR or COW
- Index type
- Compute engine
Design Decisions

01 Data lake selection
   Iceberg: Good data abstraction and excellent interface design
   Hudi: Flexible interface implementation, global index, MOR
   DeltaLake: Strong binding with spark

02 Real-time write
   COW or MOR

03 Index type
   Simple Bloom HbaseIndex

04 Compute engine
   Spark or Flink | RDD API or DataSource API
03 Functionality Support

- MVCC
- Schema registration system
MVCC

Payload  Custom data structure  Timestamp  View Access  Append
Schema Registry
- Atomic change
- Multi-Site high availability
- Versioning and revert
- Column property
- Heterogeneous data automatic sync
- Column sequence encoding

Application

Pull Sync

Local Cache
04 Performance Tuning

- Serialization
- Compaction HDFS
- SLA
- Process optimization
Serialization

State of the art
- 1000-10000+ columns
- Average column length of 20 characters
- Single row of 10MB+
- Resolver 4G+
- Serialization time 30%+

1. Rename columns as IDs
   Global singleton of schema object

2. Reduce the number of deserialization
   Tuning GC

3. PreCompile Implementation*
   Fix code length exceed

https://github.com/linkedin/avro-util *
Compaction

State of the art
- Inline
- Long time
- Inflexible resources

1. Independent resource scheduling
   Use cheap resources

2. Rules + heuristic scheduling *

3. Process and memory usage optimization
HDFS SLA

1. Hflush or Hsync

2. Aggressive retry strategy
   Slicing upon timeout/rollover

3. Independent cluster
Process Optimization

Some small process optimizations and bug fixes

- Avoid rewrite operations
- Plug-in record size evaluation
- Small file evaluation based on row count
- Simple adaptive execution to avoid write skew
- Custom partitioner to optimize shuffle
- Bulkinsert indexing bulkload
- Timeline cache inconsistent update
05 Future Work

- Productization
- Support for ecosystem
- Cost optimization
- Performance optimization
- Storage semantics
Future Work

1. Productization
   - User-friendly programming
   - Operability and maintainability
   - Simplified tuning

2. Support for ecosystem
   - Flink
   - Cross-language, cross-framework format
   - Universal access
   - Internal ecosystem improvement

3. Cost optimization
   - Tiered storage for cold and warm data
   - Serialization optimization
   - Mixed tidal compute
   - Optimize compaction method

4. Performance optimization
   - Fast machine Vectorization
   - New format index process
   - Workflow reconstruction

5. Storage semantics
   - Incremental trigger
   - Mutate
   - Check and scan
   - Data reorganization
Q&A Time
We are hiring!

字节跳动推荐架构团队
- 负责抖音、今日头条、西瓜视频等产品的推荐架构的设计和开发，保障系统稳定和高可用；
- 负责在线服务、离线数据流性能优化，解决系统瓶颈，降低成本开销；
- 抽象系统通用组件和服务，建设推荐中台、数据中台，支撑新产品快速孵化以及ToB赋能；
- 设计和实现高并发、高吞吐的服务框架、RPC框架，为业务提供快速构建服务以及高性能在线serving能力；
- 实现灵活可扩展的高性能存储系统和计算模型，打通在线数据流，构建统一的数据中台，支持推荐/搜索/广告；

团队目前招聘以下岗位：
- 大数据开发工程师
  深入了解大数据生态组件的原理
- 存储研发工程师
  熟悉rocksdb/Hbase，熟悉分布式存储
- 推荐/搜索/广告相关推荐架构工程师、后端开发工程师
- 深度学习框架研发
- devops/研发效能/编译优化
- 网络通信组件/rpc开发
- 运维工程师

工作地点：北京/上海/杭州/新加坡/山景城

欢迎自荐&推荐，岗位相关问题欢迎私戳微信
或将简历投递至邮箱
guanziyue.gzy@bytedance.com
THANKS.