Powering Real-Time AI at Pinterest: Feature Management and Serving at Scale

Andrey Gubenko, Software Engineer, Pinterest

Li Tang, Sr. Software Engineer, Pinterest

Organized by HOPSWORKS

Presented by

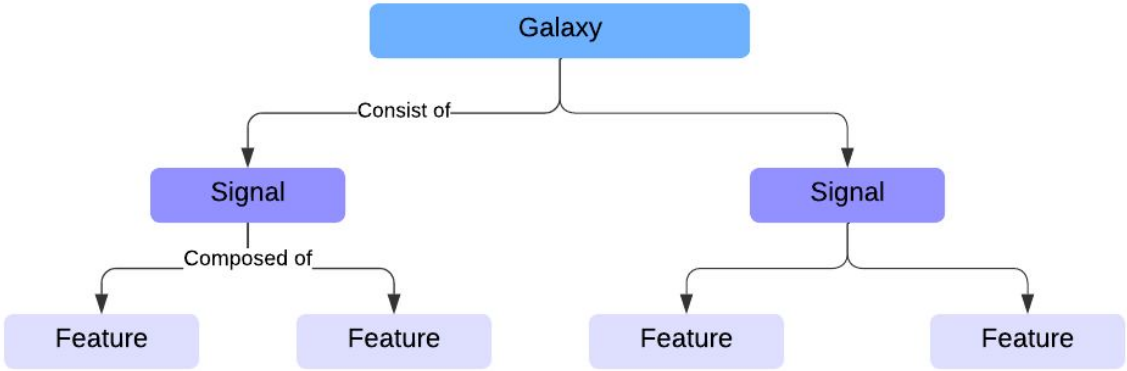
Galaxy Signal Platform

Overview

- End-to-End Signal Management
- Signal Registry and Metadata
- Storage and Access
- Integration with Data Processing Frameworks
- Online Signal Serving

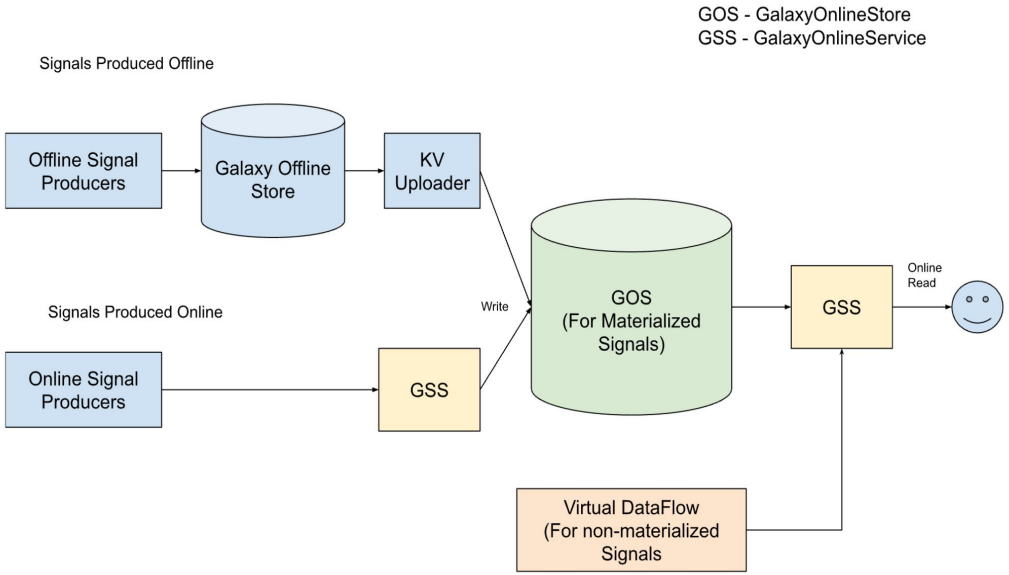
Galaxy Signal Platform

Concepts



Galaxy Signal Platform

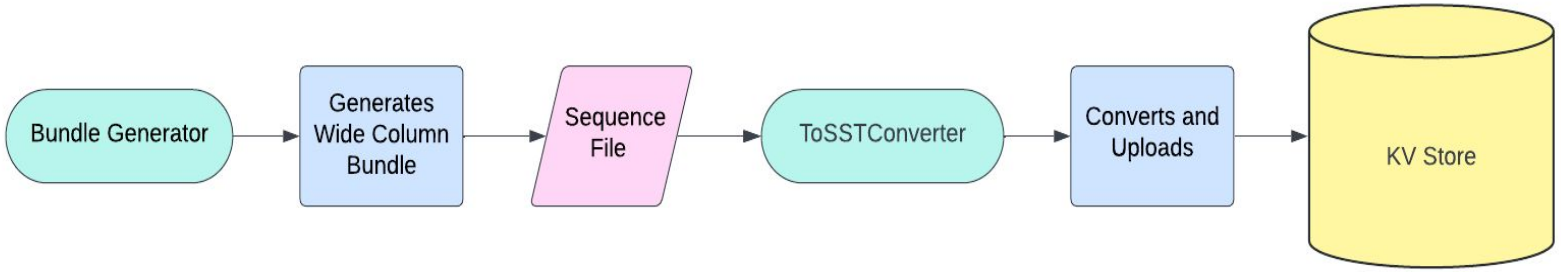
Architecture



Galaxy Online Store

Feature Ingestion: Batch/Streaming

- Flink
  - Realtime upload through Galaxy sink API
- Galaxy KV Uploader



Galaxy Signal Platform

Signal types

- Materialized Signals- directly from Galaxy Online Store
- Virtual Signals - lazily computed per request

Galaxy Signal Platform

Virtual Signal concept

- Lazily computed at real time
- Accessing many real time APIs
- Expressed as a set of transforms
- Can produce cross join features

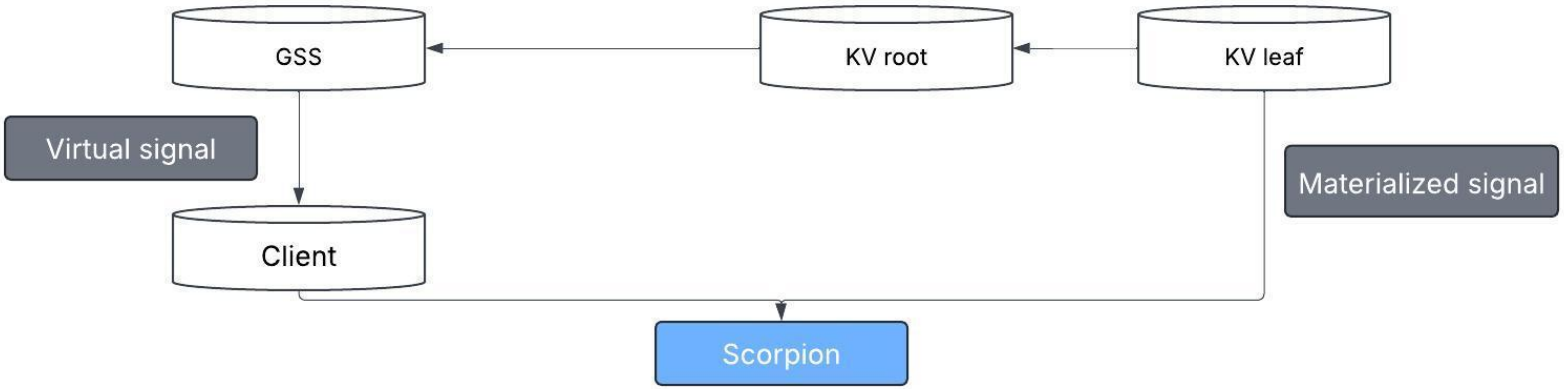
Galaxy Signal Platform

User Sequence serving

- User Sequence serving via Virtual Signals
- Time-based Stitching
- Request-Time Aggregation and Transformation
- Unified User Sequence Serving

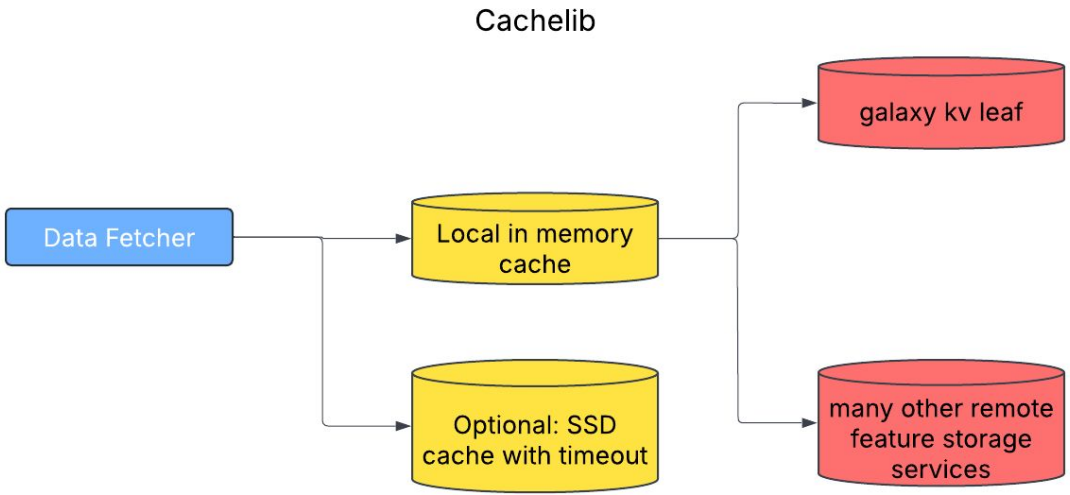
Scorpion: ML Recommender serving service

- Feature fetch, training data logging, dynamic batching and gpu serving
- Strong performance requirement - *sub-100-millisecond with hundreds millions QPS*



Scorpion Data fetcher

Cache layer → remote storage layer

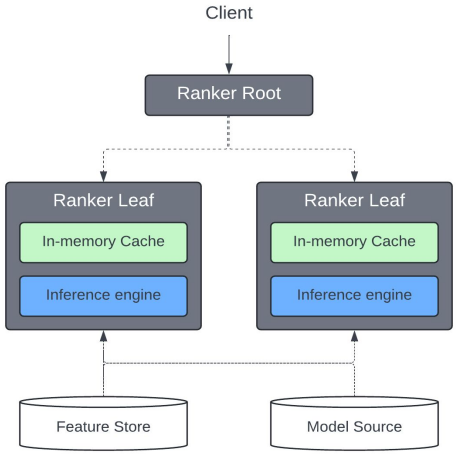


Data fetcher cache layer: Cachelib (open sourced by meta)

1. **C++ compatibility**
2. **Memory management**
   a. Minimize memory fragmentation
   b. Eviction Algorithms: LRU, TinyLFU….
   c. TTL garbage collection
   d. Cache pool management
3. **Zero-copy reads**
4. **Persistent cache**:
5. **SSD support**:
6. **Robust Operations in O(300M) QPS with no downtime**

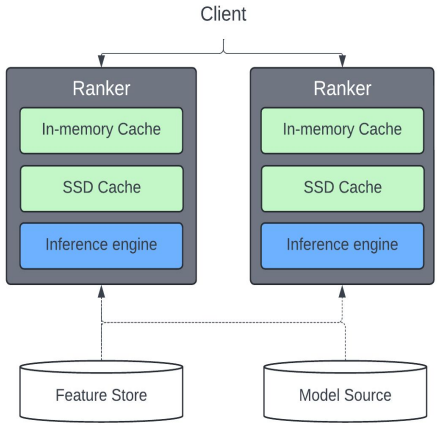
Cachelib Adoption patterns in Scorpion

Sharded Cache



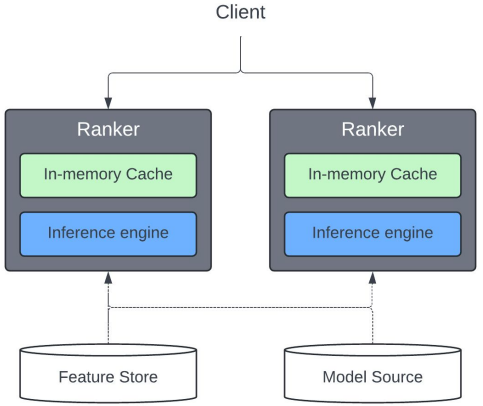
Good for cpu serving with extremely high cache hit rate

Hybrid cache



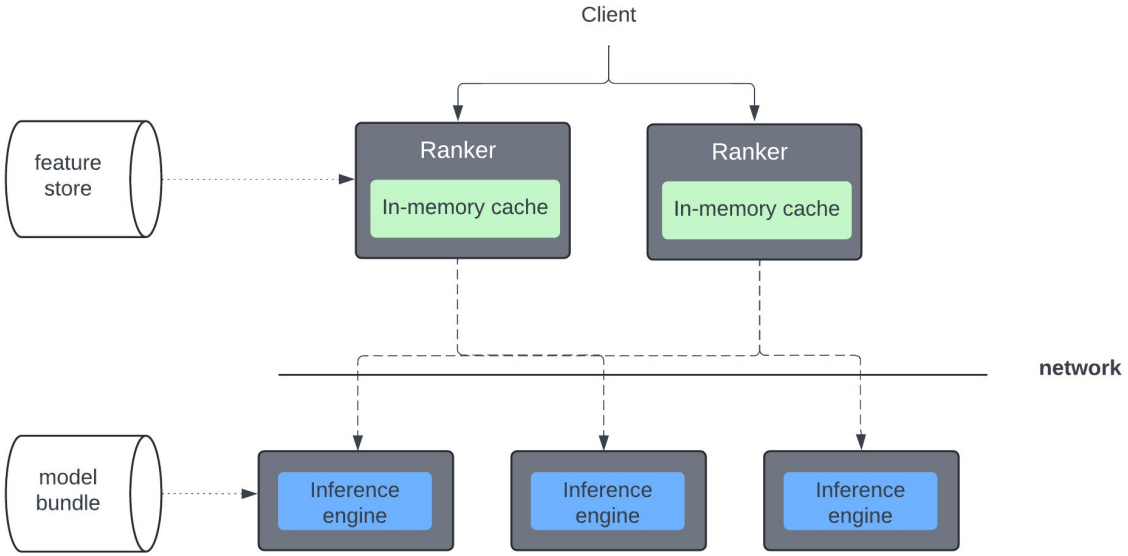
Good for gpu serving with non latency sensitive use-cases.

Local large in-memory cache



Good for gpu serving with latency sensitive use-cases.

Current setup in most of use case: Model farm



Feature hybradtion layer

- Feature hydration
- Logging
- Traffic routing

Inference layer

- model inference.
- composed of several partitions, each containing a small subset of models.

Reference

- https://medium.com/pinterest-engineering/feature-caching-for-recommender-systems-w-cachelib-8fb7bacc2762

Thank you!