LAPORAN PUB-SUB LOG AGGREGATOR

Sistem Paralel dan Terdistribusi – A

Disusun Oleh :

Muhammad Azka Yunastio 11231036

24 Oktober 2025

PENJELASAN

1. Jelaskan karakteristik utama sistem terdistribusi dan trade-off yang umum pada desain Pub-Sub log aggregator.

Jawaban:

Sistem terdistribusi memiliki ciri utama berupa *resource sharing*, *distribution transparency*, *openness*, *dependability*, dan *scalability* (Van Steen & Tanenbaum, 2023, hlm. 10–24). Tujuan utamanya ialah memudahkan akses sumber daya lintas jaringan sambil menyembunyikan kompleksitas distribusi. Namun, setiap karakteristik menghadirkan *trade-off* desain. Misalnya, peningkatan *transparency* (seperti *failure* atau *replication transparency*) dapat menurunkan *performance* karena adanya overhead komunikasi dan sinkronisasi antar node. Dalam konteks *Publish–Subscribe log aggregator*, muncul kompromi antara *consistency–availability (CAP)*, *ordering–throughput*, dan *latency–durability*. Sistem yang menargetkan throughput tinggi umumnya hanya menjamin *eventual consistency* dengan mengandalkan *idempotent consumer* untuk menjaga determinisme hasil tanpa koordinasi berlebih.

2. Bandingkan arsitektur client–server dan publish–subscribe untuk aggregator. Kapan memilih Pub-Sub?

Jawaban:

Arsitektur **client–server** bekerja secara *synchronous* antara peminta dan penyedia layanan, sementara **publish–subscribe (Pub-Sub)** menggunakan *event broker* yang memisahkan keduanya (Van Steen & Tanenbaum, 2023, hlm. 68–72). Dalam Pub-Sub, produsen (*publisher*) menerbitkan pesan bertopik tertentu, dan konsumen (*subscriber*) menerima event sesuai ketertarikan mereka. Pendekatan ini menyediakan *asynchronous decoupling* dalam waktu, ruang, dan sinkronisasi. Pub-Sub lebih tepat digunakan pada sistem agregasi log berskala besar karena mendukung *dynamic subscription*, *fault isolation*, dan *horizontal scalability*. Sebaliknya, *client–server* lebih cocok untuk komunikasi deterministik dengan *low latency*. Pub-Sub dipilih ketika sistem membutuhkan *event-driven pipeline* yang toleran terhadap latensi dan mendukung *fan-out* tinggi.

3. Uraikan at-least-once vs exactly-once delivery semantics. Mengapa idempotent consumer krusial?

Jawaban:

Dalam komunikasi pesan, dikenal tiga jaminan utama: **at-most-once**, **at-least-once**, dan **exactly-once delivery semantics** (Van Steen & Tanenbaum, 2023, hlm. 208–213). *At-least-once* menjamin setiap pesan dikirim minimal sekali, namun berisiko duplikasi akibat *retry*. Sementara itu, *exactly-once* memastikan tidak ada duplikasi, tetapi menuntut *transactional coordination* dan *logging* sinkron yang kompleks. Karena itu, *idempotent consumer* menjadi penting untuk menjaga konsistensi hasil meski event dikirim ulang. Dengan menyimpan *event_id* yang telah diproses dalam *deduplication store*, sistem dapat mencapai efek *semantic exactly-once* di atas transport layer yang hanya mendukung *at-least-once*.

4. Rancang skema penamaan untuk topic dan event_id (unik, collision-resistant). Jelaskan dampaknya terhadap dedup.

Jawaban:

Bab 6 dalam buku menjelaskan pentingnya *structured* dan *globally unique naming* agar entitas dapat diakses tanpa tabrakan (Van Steen & Tanenbaum, 2023, hlm. 385–387). Konsep ini diadaptasi dari *Named-Data Networking (NDN)* yang menggunakan hierarki nama seperti berikut:

topic = /domain/service/entity
event_id = SHA256(source_id + timestamp + seq)

Skema tersebut memungkinkan *collision resistance* serta meningkatkan efisiensi *filtering* pada sistem *Publish–Subscribe*. NDN juga mengasumsikan setiap versi data memiliki nama baru, sehingga pembaruan tidak menimpa versi lama. Hal ini memperkuat efektivitas *deduplication*, karena setiap event_id bersifat unik dan persisten, memungkinkan sistem mengenali duplikasi tanpa memeriksa keseluruhan payload.

5. Bahas ordering: kapan total ordering tidak diperlukan? Usulkan pendekatan praktis dan batasannya.
Jawaban:
Total ordering hanya diperlukan jika urutan operasi memengaruhi hasil akhir, misalnya pada transaksi keuangan (Van Steen & Tanenbaum, 2023, hlm. 260–266). Dalam sistem *log aggregator*, *causal ordering* sudah mencukupi. Pendekatan praktis menggunakan kombinasi **timestamp** dan **monotonic counter** per sumber data, sehingga urutan lokal tetap konsisten tanpa koordinasi global. Kelemahannya, *clock skew* antar node dapat menimbulkan *out-of-order event*. Alternatif seperti *Lamport clocks* atau *vector clocks* dapat memperbaiki urutan kausal, namun menambah overhead. Karena itu, sebagian besar sistem *Pub-Sub* memilih *approximate ordering* untuk menjaga performa.

6. Identifikasi failure modes dan strategi mitigasinya.
Jawaban:
Van Steen dan Tanenbaum (2023, hlm. 462–471) menyebut beberapa kegagalan umum pada sistem terdistribusi: **duplication**, **out-of-order delivery**, dan **crash failure**. Strategi mitigasi meliputi **retry dengan exponential backoff** guna mencegah *retry storm*, penggunaan **durable deduplication store** (misalnya Redis atau RocksDB) untuk mencatat *processed event_id*, serta **checkpoint dan log replay** untuk *recovery* status sistem. Selain itu, kombinasi *acknowledgment* dan *timeout control* membantu menyeimbangkan antara *reliability* dan *performance*. Strategi ini mengikuti prinsip *fault masking by redundancy*, memastikan sistem tetap beroperasi meskipun sebagian komponennya gagal.

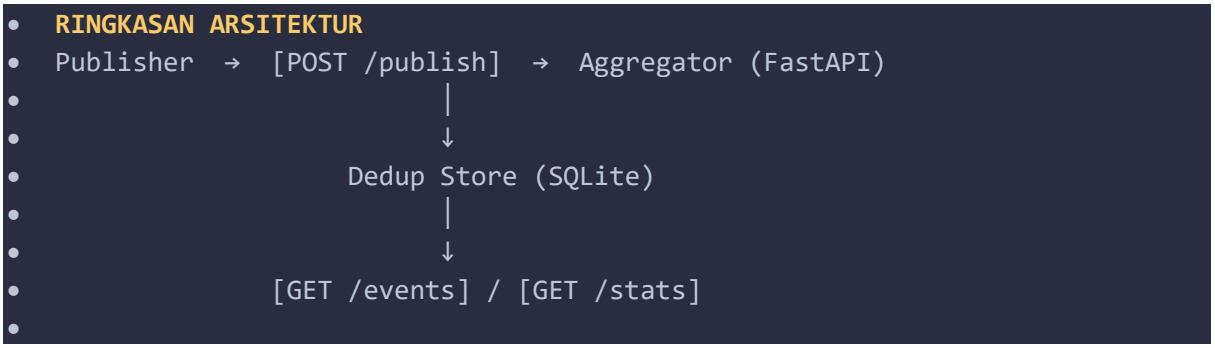
7. Definisikan eventual consistency pada aggregator; jelaskan bagaimana idempotency dan dedup membantu mencapainya.
Jawaban:
Menurut Van Steen dan Tanenbaum (2023, hlm. 406–410), **eventual consistency** berarti bahwa semua replika sistem akan mencapai keadaan yang sama setelah periode propagasi tertentu. Dalam *log aggregator*, *idempotent consumer* memastikan hasil pemrosesan tidak berubah walau event dikirim ulang, sementara *deduplication* mencegah divergensi antar node. Kombinasi kedua mekanisme tersebut menciptakan *convergence guarantee* tanpa koordinasi sinkron seperti *two-phase commit*. Pendekatan ini sesuai dengan prinsip *BASE (Basically Available, Soft state, Eventually consistent)*, yang menjaga ketersediaan tinggi sekaligus menjamin konsistensi akhir.

8. Rumuskan metrik evaluasi sistem dan kaitkan dengan keputusan desain.
Jawaban:

Evaluasi sistem terdistribusi melibatkan metrik seperti **throughput (events/s)**, **latency (ms/event)**, dan **duplicate rate (%)** (Van Steen & Tanenbaum, 2023, hlm. 208–213; 536–541). *Throughput* menunjukkan kapasitas paralel sistem, *latency* merepresentasikan efisiensi komunikasi, sementara *duplicate rate* mengukur efektivitas *deduplication*. Tambahan metrik seperti **durability** dan **recovery time** menilai ketahanan terhadap kegagalan. Peningkatan throughput sering menaikkan latency atau tingkat duplikasi. Oleh karena itu, desain optimal menggunakan *asynchronous I/O*, *batch acknowledgment*, dan *bounded retry* untuk menyeimbangkan performa, reliabilitas, dan konsistensi sesuai prinsip *scalable fault-tolerant distributed systems*.

Keputusan Desain Utama

- Idempotency: Setiap event memiliki (topic, event_id) unik yang disimpan di dedup store (SQLite).
- Dedup Store: Menggunakan tabel persisten dedup(topic, event_id, processed_at) agar data tidak hilang saat restart.
- Ordering: Menggunakan timestamp ISO 8601 sebagai urutan aproksimasi antar event.
- Retry Mechanism: Publisher dapat mengirim ulang event (at-least-once), tetapi aggregator menolak duplikat.
- Persistence: Volume Docker .data menjamin database tetap tersimpan di host.



Daftar Pustaka

Van Steen, M., & Tanenbaum, A. S. (2023). *Distributed systems: Principles and paradigms* (4th ed.). Vrije Universiteit Amsterdam.

DEMONSTRASI

https://youtu.be/QbtCG-hsy_4

LINK GITHUB

kyogree1/Pub-Log-Aggregator-11231036