Nama : Muamnar Ihsan
NIM : 11221053

T1 (Bab 1): Karakteristik Utama dan Trade-off Desain Pub-Sub Log Aggregator

Karakteristik utama yang memandu desain sistem terdistribusi, termasuk log aggregator, meliputi Resource Sharing, Distribution Transparency (seperti transparansi lokasi dan kegagalan), Openness, Dependability (keandalan dan ketersediaan), Security, dan terutama, Scalability (Van Steen & Tanenbaum, 2023). Scalability telah menjadi salah satu tujuan desain yang paling penting untuk pengembang sistem terdistribusi, didorong oleh peningkatan perangkat jaringan dan layanan cloud (Van Steen & Tanenbaum, 2023).

Dalam desain log aggregator Pub-Sub, trade-off yang umum terjadi adalah antara Consistency dan Scalability (Van Steen & Tanenbaum, 2023). Trade-off ini muncul karena replikasi data sering digunakan untuk meningkatkan kinerja, tetapi replikasi memerlukan mekanisme sinkronisasi global untuk menjaga semua salinan tetap konsisten saat pembaruan terjadi (Van Steen & Tanenbaum, 2023). Sayangnya, mekanisme sinkronisasi global ini sangat sulit diimplementasikan secara scalable, terutama dalam sistem berskala besar, karena adanya batas bawah alami pada latensi jaringan (network latencies) (Van Steen & Tanenbaum, 2023). Oleh karena itu, log aggregator biasanya memilih model konsistensi yang lebih lemah, seperti Eventual Consistency, untuk memungkinkan throughput dan high availability yang lebih tinggi, mengorbankan jaminan konsistensi seketika (instantaneous consistency) (Coulouris et al., 2012).

T2 (Bab 2): Perbandingan Arsitektur Client-Server vs Publish-Subscribe Kapan memilih Pub-Sub?

Arsitektur client-server tradisional adalah model berlapis (layered) yang mengikuti pola komunikasi request-reply (Van Steen & Tanenbaum, 2023). Interaksi dalam model ini bersifat tightly coupled karena klien secara eksplisit merujuk ke server (space-coupled) dan biasanya menunggu respons secara sinkron (time-coupled) (Coulouris et al., 2012).

Sebaliknya, Publish-Subscribe (Pub-Sub) adalah architectural style yang fundamentalnya dirancang untuk information dissemination melalui layanan perantara (intermediary service) (Coulouris et al., 2012). Fitur utamanya adalah publisher dan subscriber tidak memiliki referensi eksplisit satu sama lain (Van Steen & Tanenbaum, 2023). Ini menawarkan space uncoupling dan time uncoupling (Coulouris et al., 2012).

Log aggregator harus memilih Pub-Sub karena alasan teknis skalabilitas dan decoupling. Pub-Sub adalah paradigma komunikasi one-to-many yang sangat cocok untuk sistem diseminasi data besar, di mana satu publisher (sumber log) perlu mengirim event ke banyak consumer (berbagai proses agregasi, analitik, dan penyimpanan) (Coulouris et al., 2012). Pemisahan ini memungkinkan sistem memiliki potensi untuk skala ke sistem berskala sangat besar (very large scale systems) dengan overhead yang rendah, karena publisher tidak perlu mengetahui detail consumer dan dapat terus memublikasikan event bahkan jika consumer gagal atau sedang sibuk (Coulouris et al., 2012).

T3 (Bab 3): At-Least-Once vs Exactly-Once Delivery Semantics dan Idempotent Consumer

Delivery semantics menjadi perhatian utama dalam menghadapi kegagalan komunikasi. At-least-once semantics menjamin bahwa suatu operasi (seperti pemrosesan log) akan dieksekusi setidaknya satu kali, tetapi mungkin lebih dari sekali (possibly more) (Van Steen & Tanenbaum, 2023). Semantik ini biasanya merupakan hasil dari mekanisme retry yang digunakan untuk menutupi omission failures (pesan hilang) (Coulouris et al., 2012). Exactly-once semantics adalah semantik yang ideal, menjamin eksekusi tepat satu kali, tetapi secara umum tidak ada cara untuk merealisasikannya (no way to arrange this in general) (Van Steen & Tanenbaum, 2023).

Idempotent consumer sangat penting dalam konteks log aggregator yang beroperasi di bawah at-least-once delivery. Suatu operasi disebut idempotent jika dapat diulang berkali-kali tanpa menghasilkan perubahan status yang berbahaya atau side effect yang tidak diinginkan (Van Steen & Tanenbaum, 2023). Karena at-least-once delivery menghasilkan pesan duplikat akibat retransmissions (Van Steen & Tanenbaum, 2023), consumer harus dirancang agar idempotent. Dengan ini, meskipun consumer menerima log yang sama berulang kali, ia dapat menerapkan duplicate filtering (Coulouris et al., 2012) atau memastikan bahwa operasi tulis agregat di penyimpanan tujuan hanya berdampak sekali secara logis, sehingga menjaga Consistency data agregasi.

T4 (Bab 4): Skema Penamaan Topic dan Event_ID yang Collision-Resistant

Skema penamaan untuk topic dalam sistem Pub-Sub harus memfasilitasi routing dan filtering (Van Steen & Tanenbaum, 2023). Untuk topic-based Pub-Sub, topic dapat dinamai menggunakan kata kunci bertingkat (compound keyword) seperti region.service.type (Van Steen & Tanenbaum, 2023). Skema ini memungkinkan distributed event matching di mana server dapat secara deterministik membagi pekerjaan berdasarkan hashing pada nama topic (Van Steen & Tanenbaum, 2023).

Untuk event_id, diperlukan pengenal (identifier) yang dirancang untuk mengacu pada paling banyak satu entitas, menjadikannya unik. Skema yang umum dan collision-resistant adalah menghasilkan event_id menggunakan Hash Function yang kuat (secure hashing), seperti SHA-1, yang diterapkan pada gabungan payload event, timestamp orisinal, dan ID publisher (Coulouris et al., 2012). Hasil hash ini menghasilkan kunci yang unik dan tamper-proof.

Dampak skema penamaan event_id ini terhadap deduplication (dedup) sangat langsung. Karena log aggregator sering menggunakan at-least-once semantics, duplikasi dapat terjadi. Dengan event_id unik, consumer dapat menggunakan ID tersebut sebagai kunci untuk mencatat execution history di dalam durable dedup store. Setiap event yang masuk diperiksa terhadap store ini; jika event_id sudah ada, pesan tersebut adalah duplikat dan diabaikan, sehingga mencapai exactly-once processing secara logis. (Van Steen & Tanenbaum, 2023).

T5 (Bab 5): Ordering: Kapan Total Ordering Tidak Diperlukan dan Pendekatan Praktis

Total Ordering menjamin bahwa jika suatu pesan dikirimkan sebelum pesan lain di satu proses, urutan yang sama persis akan dipertahankan di semua proses (Coulouris et al., 2012). Urutan ini diperlukan untuk sistem yang harus berperilaku sebagai state machine replication, di

mana semua replika harus mengeksekusi operasi yang sama dalam urutan yang sama untuk menjaga konsistensi yang kuat (Van Steen & Tanenbaum, 2023).
Total ordering tidak diperlukan jika aplikasi hanya membutuhkan FIFO Ordering (urutan dari satu sumber/proses) atau Causal Ordering (urutan happens-before) (Coulouris et al., 2012). Dalam log aggregation, urutan log global antar topic yang berbeda mungkin tidak penting; yang penting adalah bahwa urutan log yang dihasilkan oleh satu aplikasi diproses secara berurutan (FIFO ordering) atau bahwa log yang saling bergantung secara kausal diproses dalam urutan yang benar (Van Steen & Tanenbaum, 2023).
Pendekatan praktis adalah menggunakan Lamport's Logical Clocks (Van Steen & Tanenbaum, 2023). Ini melibatkan counter yang diperbarui pada setiap event dan disinkronkan berdasarkan timestamp pesan yang diterima. Untuk menghasilkan Total Ordering yang unik, timestamp logis ini dapat diperluas dengan menambahkan pengenal proses unik (unique process identifier) untuk memecahkan ikatan (break ties).
Batasan utama pendekatan ini adalah bahwa urutan total yang dihasilkan adalah urutan logis yang konsisten dengan kausalitas, tetapi tidak menjamin urutan yang sesuai dengan waktu fisik nyata (real-time order), yang merupakan persyaratan untuk Linearizability (Coulouris et al., 2012). Untuk linearizability, dibutuhkan mekanisme yang mengandalkan sinkronisasi physical clocks yang sangat akurat, seperti Google TrueTime, atau mekanisme konsensus yang kompleks (Van Steen & Tanenbaum, 2023).

T6 (Bab 6): Failure Modes dan Strategi Mitigasi

Failure modes yang umum dalam sistem terdistribusi meliputi Crash Failure (proses berhenti, yang diasumsikan fail-stop), Omission Failure (gagal merespons atau mengirim/menerima pesan), dan Timing Failure (respons di luar interval waktu yang ditentukan) (Van Steen & Tanenbaum, 2023). Dalam konteks jaringan, omission failures sering diatasi dengan retry, yang dapat menyebabkan Duplikasi Pesan.
Strategi mitigasi utama berfokus pada redundancy dan keandalan komunikasi (Van Steen & Tanenbaum, 2023):
1. Retry: Digunakan untuk menutupi omission failures (pesan hilang) (Coulouris et al., 2012). Klien akan mengirim ulang permintaan jika tidak ada balasan yang datang dalam batas timeout.
2. Backoff (Implisit): Penggunaan retry harus menyertakan timeout untuk deteksi kegagalan. Meskipun sumber tidak mendefinisikan backoff secara eksplisit, timeout adalah komponen penting dalam mendeteksi dan mengatasi masalah latensi dan crash tanpa membanjiri server dengan retransmissions (Van Steen & Tanenbaum, 2023).
3. Durable Dedup Store: Untuk mengatasi masalah duplikasi (yang dihasilkan oleh retry di bawah at-least-once semantics), mekanisme duplicate filtering harus diimplementasikan (Coulouris et al., 2012). Consumer harus mencatat request ID unik (event_id) dalam penyimpanan yang tahan lama (durable log). Penyimpanan ini harus tahan terhadap crash failure sehingga memungkinkan sistem mendeteksi dan mengabaikan event yang telah diproses sebelumnya saat consumer pulih (recovery) (Van Steen & Tanenbaum, 2023).

T7 (Bab 7): Eventual Consistency, Idempotency, dan Dedup dalam Aggregator

Eventual Consistency (EC) adalah model konsistensi yang relatif mudah dipahami dan diimplementasikan (Van Steen & Tanenbaum, 2023). EC menjamin bahwa jika tidak ada pembaruan lebih lanjut, semua replika data akan akhirnya konvergen ke nilai yang sama, meskipun mungkin ada periode inkonsistensi sementara (Coulouris et al., 2012). EC sering dipilih dalam log aggregator karena mendukung ketersediaan tinggi (high availability) dan kinerja yang scalable (Van Steen & Tanenbaum, 2023).

Idempotency dan Deduplication (Dedup) adalah mekanisme penting untuk mencapai konsistensi di bawah model EC, terutama dalam sistem dengan at-least-once delivery (Van Steen & Tanenbaum, 2023).

1. Idempotency: Memungkinkan proses consumer untuk mengulang operasi logis apa pun tanpa mengakibatkan side effect tambahan.
2. Dedup: Menggunakan unique event ID untuk mencatat setiap operasi unik dalam durable store (Van Steen & Tanenbaum, 2023).

Dalam log aggregator, duplikasi pesan adalah side effect dari retry untuk mencapai fault tolerance. Dengan memastikan bahwa consumer idempotent dan menggunakan dedup berbasis ID unik, sistem secara efektif memastikan bahwa pemrosesan event hanya terjadi satu kali secara logis, bahkan jika salinan fisik event diterima beberapa kali. Hal ini menjamin bahwa, seiring waktu, semua replika dari hasil agregasi (jika direplikasi) akan konvergen ke hasil yang benar dan konsisten, yang merupakan esensi dari strong eventual consistency (Van Steen & Tanenbaum, 2023).

T8 (Bab 1–7): Rumusan Metrik Evaluasi Sistem dan Kaitannya dengan Keputusan Desain

Metrik evaluasi kunci untuk log aggregator Pub-Sub meliputi:

1. Throughput: Jumlah event log yang diproses per detik.
2. Latency: Waktu yang diperlukan untuk menyelesaikan pemrosesan event.
3. Duplicate Rate: Tingkat pesan yang diterima yang merupakan duplikat logis, yang mengukur overhead keandalan.
4. Availability: Persentase waktu layanan tersedia.

Metrik ini terkait erat dengan keputusan desain inti sistem terdistribusi (Van Steen & Tanenbaum, 2023):

• Throughput / Scalability: Keputusan desain untuk menggunakan arsitektur Publish-Subscribe (Indirect Communication) secara langsung mengoptimalkan throughput dan scalability dibandingkan dengan client-server. Memilih Eventual Consistency daripada Total Ordering adalah trade-off yang disengaja untuk meningkatkan throughput dengan menghindari bottleneck sinkronisasi global (Van Steen & Tanenbaum, 2023).

• Latency / Consistency: Peningkatan latency terjadi jika sistem mencoba menjamin konsistensi yang lebih ketat (seperti Linearizability) karena adanya latensi minimum pada jaringan. Sebaliknya, desain yang memprioritaskan latensi rendah (misalnya, dengan pemrosesan pada replika terdekat) seringkali menerima eventual consistency sebagai gantinya.

• Duplicate Rate / Dependability: Keputusan untuk menggunakan at-least-once semantics (untuk meningkatkan dependability dan fault tolerance terhadap omission failures) secara inheren meningkatkan risiko duplicate rate. Untuk memitigasinya, keputusan desain harus mencakup implementasi idempotency dan deduplication di sisi consumer untuk menjamin integrity data agregasi (Coulouris et al., 2012).