FHIQI MAULANA AS-SDDIQ

11221003

UTS Sister – Teori

1. Jelaskan karakteristik utama sistem terdistribusi dan trade-off yang umum pada desain Pub-Sub log aggregator.
   Jawab:
   Karakteristik utama sistem terdistribusi:
   a. Skalabilitas (Scalability): Kemampuan sistem untuk menangani peningkatan beban kerja atau pengguna tanpa penurunan kinerja, umumnya dicapai dengan menambahkan lebih banyak server.
   b. Keandalan (Reliability): Kemampuan sistem untuk terus berfungsi dengan benar meskipun satu atau lebih komponen mengalami kegagalan (fault-tolerant). Hal ini dicapai melalui redundansi komponen.
   c. Ketersediaan (Availability): Ukuran waktu sistem tetap operasional dan siap melayani permintaan. Ketersediaan tinggi memastikan pengguna dapat mengakses sistem kapan saja.
   d. Efisiensi (Efficiency): Kemampuan sistem untuk menggunakan sumber dayanya secara efektif. Diukur melalui waktu respons (latensi) dan throughput (jumlah item yang dikirim per satuan waktu) (Sakalli, 2023).

   Trade-off Umum pada Desain Pub-Sub Log Aggregator

   Trade-off pada log aggregator berbasis Pub-Sub (Publish-Subscribe) umumnya berkisar pada tiga hal:

   a. Konsistensi (Urutan Pesan) vs. Latensi: Menjamin urutan log yang ketat di seluruh broker akan meningkatkan latensi dan mengurangi throughput sistem secara keseluruhan.
   b. Keandalan vs. Biaya: Menyimpan log secara permanen (persistence) untuk keandalan data maksimum membutuhkan biaya penyimpanan dan overhead operasional yang lebih tinggi.
   c. Skalabilitas vs. Prediktabilitas: Sistem Pub-Sub terkelola (managed) memberikan skalabilitas otomatis namun pengguna memiliki kontrol yang lebih kecil terhadap partisi, yang dapat menyebabkan kinerja kurang dapat diprediksi (Khalilov, 2025).

2. Bandingkan arsitektur client-server vs publish-subscribe untuk aggregator. Kapan memilih Pub-Sub? Berikan alasan teknis.
   Jawab:

Fitur	Client-Server	Publish-Subscribe

Pola komunikasi	Sinkron: permintaan-respons	Asinkron: berbasis pesan/peristiwa
Keterkaitan	Keterkaitan kuat, klien (aplikasi penghasil log) harus mengetahui lokasi dan ketersediaan Server (Aggregator)	Keterkaitan longgar, publisher (aplikasi) tidak perlu mengetahui siapa atau berapa banyak Subscriber (sistem pemroses log) yang ada
Tujuan komunikasi	Point-to-point (satu pengirim ke satu penerima) atau pull-based (klien harus meminta data secara eksplisit)	One-to-many (satu publisher ke banyak subscriber), atau fan-out
Ketahanan pesan	Kurang tangguh; jika Server down, Client mungkin gagal mengirim log (blocking) dan datanya bisa hilang jika tidak ditangani dengan retry lokal.	Tangguh (Reliable); pesan diantrekan/disimpan sementara oleh broker (antrian pesan/topik). Publisher tidak terpengaruh jika Subscriber down

Secara teknis, arsitektur Publish-Subscribe adalah pilihan yang jauh lebih unggul dan menjadi praktik terbaik untuk sistem log aggregation modern. Terlebih lagi jika membutuhkan log aggregator skala besar atau high-throughput (PubNub, 2023).

3. Uraikan at-least-once vs exactly-once delivery semantics. Mengapa idempotent consumer krusial di presence of retries?
   Jawab:

   Delivery Semantics menetapkan jaminan pengiriman pesan:

   - At-Least-Once: Pesan dijamin dikirim satu kali atau lebih. Model ini adalah default pada banyak sistem Pub-Sub ketika retry diaktifkan untuk mengatasi omission failures (kegagalan pengiriman), namun berisiko duplikasi pesan.
   - Exactly-Once: Pesan dijamin dikirim dan diproses tepat satu kali. Ini adalah jaminan terkuat dan paling mahal, sering dicapai dengan menambahkan lapisan transactional logic atau deduplication di consumer.

   Idempotency adalah properti dari suatu operasi di mana operasi tersebut, ketika dieksekusi beberapa kali dengan input yang sama, akan menghasilkan efek status akhir yang sama dengan saat dieksekusi hanya sekali. Idempotent consumer sangat penting dalam sistem yang menggunakan semantik pengiriman at-least-once dan mekanisme retry karena dapat mencegah duplikasi status akibat retries (Confluent, 2020).

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

   Skema penamaan yang dirancang untuk Topic dan Event_ID sangat penting dalam log aggregation untuk organisasi dan ketahanan sistem, terutama dalam mendukung proses deduksi duplikasi (dedup).
   Event_ID harus unik secara global dan tahan terhadap tabrakan (collision-resistant) untuk mengidentifikasi pesan log tunggal. Ini dicapai melalui kombinasi data yang stabil dan ID unik yang dihasilkan secara terdistribusi.

   Dampak terhadap deduplication: Kehadiran Event_ID yang unik dan collision-resistant di dalam payload pesan adalah krusial untuk menerapkan deduksi duplikasi (dedup) yang efektif pada sistem at-least-once.

   - Memungkinkan Idempotency: Event_ID bertindak sebagai kunci unik (primary key) saat pesan log ditulis ke sistem penyimpanan hilir (misalnya, index Elasticsearch atau database).
   - Mekanisme Dedup: Ketika consumer menerima pesan dan mencoba menyimpannya, ia memeriksa apakah Event_ID tersebut sudah ada di penyimpanan. Jika ID sudah ada, itu menandakan duplikasi (retry), dan consumer dapat mengabaikan pesan tersebut (operasi idempotent). Jika ID belum ada, consumer menyisipkan log baru (Confluent, 2020).

5. Bahas ordering: kapan total ordering tidak diperlukan? Usulkan pendekatan praktis (mis. event timestamp + monotonic counter) dan batasannya.
   Jawab:

   Pemesanan (ordering) pesan dalam sistem terdistribusi dibagi menjadi dua konsep utama:

   a. Total Ordering: Semua consumer melihat semua pesan dalam urutan yang sama persis (global). Ini berarti jika pesan A dikirim sebelum pesan B, setiap consumer di seluruh sistem akan memproses A sebelum B.
   b. Parsial Ordering (atau Per-Partisi): Pesan hanya dijamin dalam urutan yang benar dalam konteks terbatas, seperti dalam satu partisi (partition) atau satu sumber (publisher). Pesan dari partisi yang berbeda tidak memiliki jaminan urutan satu sama lain.

   Total ordering hanya krusial ketika ada ketergantungan status (state dependency) antar pesan (misalnya, transaksi keuangan di mana debit harus terjadi setelah kredit, atau replikasi basis data). Untuk log aggregation yang fokus pada event stream, partial ordering per sumber sudah memadai.

Pendekatan paling praktis untuk mencapai ordering yang andal dan terukur dalam log aggregator adalah menggabungkan penempatan pesan yang cerdas dengan metadata yang akurat.

- Pemesanan Per Sumber (Partial Ordering): menggunakan kunci partisi yang stabil berdasarkan identitas sumber log. Hasilnya semua log dari satu sumber akan selalu dikirim ke partisi yang sama, menjamin pesan diterima consumer dalam urutan yang dikirim oleh sumber tersebut.
- Metadata Ordering: Menambahkan dua metadata kritis pada setiap payload log seperti event timestamp dan monotonic counter. Jika terjadi duplikasi pesan atau out-of-order yang disebabkan oleh kegagalan jaringan, consumer dapat menggunakan Monotonic Counter sebagai urutan sekunder untuk mengurutkan pesan dari sumber yang sama, dan Event Timestamp sebagai kunci pengurutan utama (Stack Overflow, 2025).

6. Identifikasi failure modes (duplikasi, out-of-order, crash). Jelaskan strategi mitigasi (retry, backoff, durable dedup store).
   Jawab:

   Tiga failure modes utama yang dihadapi oleh log aggregator berbasis Pub-Sub dalam lingkungan terdistribusi adalah:
   Failure Modes dan Strategi Mitigasi:
   a. Duplikasi (Duplication): Terjadi ketika publisher mengirim pesan yang sama lebih dari satu kali, atau ketika broker mengirimkan pesan yang sama berulang kali ke consumer karena gagal menerima acknowledgement.
   b. Tidak Berurutan (Out-of-Order): Terjadi ketika pesan diterima oleh consumer dalam urutan yang berbeda dari urutan pengirimannya.
   c. Kecelakaan (Crash): Terjadi ketika salah satu komponen, baik publisher (aplikasi sumber log), broker (antrean pesan), atau consumer (pemroses log) tiba-tiba berhenti berfungsi karena kegagalan hardware, bug software, atau resource exhaustion.

   Strategi Mitigasi

   a. Retry: Mekanisme otomatis bagi publisher atau consumer untuk mencoba kembali operasi yang gagal (misalnya, mengirim log atau memproses log). Ini memastikan keandalan pengiriman (delivery reliability).
   b. Backoff: Menerapkan penundaan yang meningkat (exponential backoff) antara upaya coba ulang. Ini mencegah sistem yang gagal menjadi overload dan memperburuk crash.
   c. Durable Dedup Store: Menerapkan idempotency dengan menyimpan Event_ID pesan yang telah berhasil diproses dalam basis data yang permanen (misalnya, Redis, DynamoDB, atau tabel database). Sebelum memproses pesan, consumer memeriksa

store ini; jika Event_ID sudah ada, pesan diabaikan, sehingga menjamin semantik exactly-once logis.

7. Definisikan eventual consistency pada aggregator; jelaskan bagaimana idempotency + dedup membantu mencapai konsistensi.
   Jawab:

   Eventual Consistency adalah model konsistensi data yang paling lemah dan paling umum dalam sistem terdistribusi skala besar, termasuk log aggregator. Dalam model ini, sistem menjamin bahwa jika tidak ada update baru yang dibuat pada suatu data, maka seiring berjalannya waktu, semua replika data tersebut pada akhirnya akan menyatu dan menjadi sama (eventually converge).

   Peran Idempotency dan Dedup dalam Mencapai Konsistensi

   Meskipun log aggregator beroperasi dengan eventual consistency, mekanisme idempotency yang didukung oleh deduplikasi (dedup) adalah elemen penting yang memastikan konsistensi tercapai dengan benar, terutama saat terjadi retry dan duplikasi pesan (semantik at-least-once).
   Idempotency (operasi berulang menghasilkan efek status akhir yang sama) dan Dedup (durable dedup store) bekerja sama untuk menahan duplikasi yang disebabkan oleh mekanisme retry pada jaringan atau broker (Confluent, 2020).

8. Rumuskan metrik evaluasi sistem (throughput, latency, duplicate rate) dan kaitkan ke keputusan desain.
   Jawab:

   Metrik evaluasi sistem sangat penting untuk mengukur kualitas dan efektivitas desain log aggregator dan untuk memvalidasi trade-off yang telah dipilih.

   - Throughput: Jumlah pesan log yang berhasil diproses (published atau consumed) per satuan waktu (misalnya, pesan per detik atau MiB per detik).
   - Latency: Waktu tunda total antara saat log dibuat (event creation) dan saat log tersebut tersedia untuk dikueri atau di-index di sistem penyimpanan hilir (Time to Index).
   - Duplicate rate: Persentase pesan log yang terdeteksi sebagai duplikat dan diabaikan selama proses deduplikasi (diukur pada sisi consumer).

   Keputusan desain arsitektur yang memilih Loose Coupling dan semantik At-Least-Once secara fundamental didorong oleh tujuan untuk memaksimalkan Throughput dan Ketersediaan (PubNub, 2023). Metrik ini kemudian digunakan untuk memantau apakah

trade-off (seperti mengorbankan Total Ordering) memberikan manfaat kinerja yang diharapkan.

Referensi:

Sakallı, D. (2023, Oktober 9). *Key characteristics of distributed systems.* Medium. https://medium.com/@sakalli.duran/key-characteristics-of-distributed-systems-0cc6e3ee08d3

Khalilov, M. (2025, Juni 20). *Kafka vs GCP Pub/Sub: A deep dive into architecture and scaling patterns*. Medium. https://medium.com/@mahammadkhalilov/kafka-vs-gcp-pub-sub-a-deep-dive-into-architecture-and-scaling-patterns-ce1b1451036f

AWS. (t.t.). *What is Pub/Sub Messaging?*. Diperoleh dari https://aws.amazon.com/what-is/pub-sub-messaging/

PubNub. (2023, September 21). *What is Publish-Subscribe (Pub/Sub Model)?*. Diperoleh dari https://www.pubnub.com/guides/pub-sub/

Confluent. (2020, Mei 14). *Exactly-Once Delivery is Possible*. Diperoleh dari https://www.confluent.io/blog/exactly-once-delivery-is-possible/

Stack Overflow. (2025, Februari 24). *How to get ordered logs using GCP Pub/Sub - python*. Diperoleh dari https://stackoverflow.com/questions/79464924/how-to-get-ordered-logs-using-gcp-pub-sub

Titievsky, K. (2020, Oktober 19). *Google Cloud Pub/Sub reliability user guide: Part 1 publishing.* Medium. https://medium.com/google-cloud/google-cloud-pub-sub-reliability-user-guide-part-1-publishing-12577b9069fd