Nama : Arief Reno Fathurrahman
NIM : 11201014

Jawab ringkas-padat (sekitar 150–250 kata per poin) dan sertakan sitasi ke bab terkait di buku utama. Gunakan istilah teknis dalam Bahasa Inggris bila relevan.

T1 (Bab 1): Jelaskan karakteristik utama sistem terdistribusi dan trade-off yang umum pada desain Pub-Sub log aggregator.

= Sistem terdistribusi memiliki komponen yang tersebar di berbagai lokasi dan berinteraksi melalui jaringan. Karakteristik utamanya meliputi transparansi, skalabilitas, dan ketahanan terhadap kegagalan. Dalam konteks Pub-Sub log aggregator, trade-off yang sering muncul adalah antara konsistensi dan ketersediaan. Ketika sistem berusaha menjaga konsistensi data di seluruh node, hal ini dapat menyebabkan peningkatan latensi dan penurunan ketersediaan. Sebaliknya, meningkatkan ketersediaan dengan mengizinkan data yang tidak konsisten dapat menghasilkan informasi yang tidak dapat diandalkan.

Desain sistem Pub-Sub juga harus mempertimbangkan cara mengelola pengiriman pesan, pengelolaan langganan, dan penanganan duplikasi. Trade-off ini memengaruhi kecepatan agregasi log serta kapasitas sistem dalam menangani volume data yang besar (Van Steen & Tanenbaum, Bab 1).

T2 (Bab 2): Bandingkan arsitektur client-server vs publish-subscribe untuk aggregator. Kapan memilih Pub-Sub? Berikan alasan teknis.

= Arsitektur client-server bergantung pada komunikasi terpusat, di mana client mengirim permintaan kepada server yang kemudian memproses dan mengembalikan hasil. Sebaliknya, arsitektur publish-subscribe (Pub-Sub) memisahkan pengirim pesan (publisher) dari penerima (subscriber). Dalam sistem Pub-Sub, penerima berlangganan pada topik dan menerima pesan secara asinkron, yang memungkinkan fleksibilitas dan skalabilitas yang lebih besar.

Pemilihan Pub-Sub lebih tepat digunakan ketika ada banyak penerima yang memerlukan data yang sama secara bersamaan. Ini juga mengurangi beban pada server karena pesan didistribusikan ke banyak penerima tanpa memerlukan interaksi langsung. Pub-Sub sangat cocok untuk aplikasi seperti notifikasi real-time dan pengumpulan log, di mana latensi dan duplikasi dapat diterima (Van Steen & Tanenbaum, Bab 2).

T3 (Bab 3): Uraikan at-least-once vs exactly-once delivery semantics. Mengapa idempotent consumer krusial di presence of retries?

= At-least-once delivery semantics memastikan bahwa pesan dikirim setidaknya satu kali, tetapi dapat menyebabkan duplikasi. Sebaliknya, exactly-once delivery semantics menjamin bahwa pesan hanya diproses sekali. Dalam konteks sistem terdistribusi, idempotent consumer menjadi sangat penting saat terjadi pengulangan. Jika konsumen dapat memproses pesan yang sama tanpa efek samping, maka sistem dapat menghindari masalah duplikasi ketika pengiriman pesan diulang.

Idempotensi memastikan bahwa meskipun pesan yang sama diterima berkali-kali, hasil akhirnya tetap konsisten. Hal ini sangat penting dalam sistem yang menerapkan at-least-once semantics, di mana kegagalan komunikasi atau pengulangan pengiriman pesan dapat menyebabkan penerimaan pesan yang sama beberapa kali (Van Steen & Tanenbaum, Bab 3).

T4 (Bab 4): Rancang skema penamaan untuk topic dan event_id (unik, collision-resistant). Jelaskan dampaknya terhadap dedup.

= Skema penamaan yang efektif untuk topik dan event_id sangat penting untuk memastikan keunikan dan menghindari bentrok. Topik dapat diberi nama menggunakan pola hierarkis yang mencerminkan struktur data dan konteks, seperti logs/application_name/event_type. Event_id harus berupa UUID yang dihasilkan secara acak, memastikan bahwa setiap ID unik dan tidak mudah ditiru.

Dampaknya terhadap deduplikasi sangat signifikan; dengan menggunakan skema penamaan yang jelas dan event_id yang unik, sistem dapat lebih mudah mengidentifikasi dan menghapus duplikasi. Ini meningkatkan efisiensi agregasi dan memastikan bahwa setiap event hanya diproses sekali, menjaga konsistensi data dalam sistem (Van Steen & Tanenbaum, Bab 4).

T5 (Bab 5): Bahas ordering: kapan total ordering tidak diperlukan? Usulkan pendekatan praktis (mis. event timestamp + monotonic counter) dan batasannya.

= Total ordering tidak selalu diperlukan dalam sistem terdistribusi, terutama pada aplikasi di mana keterlambatan dapat diterima. Dalam situasi di mana pesan memiliki karakter independen, seperti log untuk analisis, urutan yang tepat dapat diabaikan. Namun, untuk transaksi keuangan atau sistem pemungutan suara, urutan total sangat penting untuk menjaga integritas.

Pendekatan praktis untuk menangani ordering adalah dengan menggunakan timestamp event yang digabungkan dengan monotonic counter. Setiap event akan diberi timestamp saat diterima, dan counter akan memastikan urutan penanganannya. Namun, pendekatan ini memiliki batasan, seperti kemungkinan konflik saat dua event dengan timestamp yang sama diterima (Van Steen & Tanenbaum, Bab 5).

T6 (Bab 6): Identifikasi failure modes (duplikasi, out-of-order, crash). Jelaskan strategi mitigasi (retry, backoff, durable dedup store).

= Failure modes dalam sistem terdistribusi mencakup duplikasi, out-of-order delivery, dan crash. Duplikasi dapat terjadi akibat pengulangan pengiriman pesan, sedangkan out-of-order delivery dapat disebabkan oleh rute yang berbeda. Strategi mitigasi meliputi penggunaan retry dengan backoff untuk mengurangi beban pada jaringan serta implementasi durable deduplication store yang menyimpan informasi tentang event yang telah diproses.

Dengan menggunakan strategi ini, sistem dapat mengurangi dampak dari kegagalan dan menjamin pengiriman pesan yang lebih andal. Mengimplementasikan penyimpanan deduplikasi yang tahan lama memungkinkan sistem untuk melacak event dan memastikan tidak ada yang terlewat, meningkatkan keandalan keseluruhan sistem (Van Steen & Tanenbaum, Bab 6).

T7 (Bab 7): Definisikan eventual consistency pada aggregator; jelaskan bagaimana idempotency + dedup membantu mencapai konsistensi.

= Eventual consistency dalam sistem aggregator berarti bahwa setelah periode waktu tertentu, semua salinan data akan konsisten. Ini dicapai dengan menggunakan idempotency dan mekanisme deduplikasi. Idempotency memastikan bahwa operasi yang sama dapat dilakukan berulang kali tanpa mengubah hasil akhir, sedangkan deduplikasi membantu menghindari pengolahan data yang sama.

Dengan menggabungkan kedua konsep ini, sistem dapat menjaga konsistensi meskipun ada latensi atau kegagalan jangka pendek. Penanganan secara efisien terhadap operasi yang berpotensi menduplikasi data memungkinkan sistem untuk mencapai tingkat konsistensi yang diinginkan tanpa mengorbankan ketersediaan (Van Steen & Tanenbaum, Bab 7).

T8 (Bab 1–7): Rumuskan metrik evaluasi sistem (throughput, latency, duplicate rate) dan kaitkan ke keputusan desain.

= Metrik evaluasi sistem terdistribusi meliputi throughput, latency, dan duplicate rate. Throughput mengukur jumlah pesan yang dapat diproses dalam waktu tertentu, sedangkan latency mengukur waktu yang dibutuhkan untuk mengirim pesan dari pengirim ke penerima. Duplicate rate mengukur frekuensi duplikasi dalam pengiriman pesan.

Metrik ini sangat penting dalam pengambilan keputusan desain. Misalnya, jika throughput rendah, sistem mungkin perlu dioptimalkan untuk meningkatkan efisiensi. Jika latency terlalu tinggi, pertimbangan untuk mengurangi jarak fisik antara server dan client atau menggunakan caching mungkin diperlukan. Mengelola duplicate rate juga penting untuk menjaga integritas data dan mencegah pemborosan sumber daya (Van Steen & Tanenbaum, Bab 1–7).