Nama : Reno Fahrezi Purnomo
NIM : 11221051
Kelas : Sistem Paralel dan Terdistribusi A

Soal Teori

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

   Beberapa Karakteristik dari sistem Terdistribusi adalah sebagai berikut:

a. Resource Sharing

Tujuan utama sistem terdistribusi adalah memudahkan akses dan berbagi sumber daya jarak jauh seperti data, file, layanan, dan penyimpanan, demi efisiensi biaya melalui penggunaan bersama fasilitas tunggal. (Steen & Tanenbaum, 2023, p. 10).

b. Distribution Transparency

Salah satu tujuan penting dari sistem terdistribusi adalah untuk menyembunyikan kebenaran bahwa proses dan sumber dayanya secara fisik tesebar pada banyak komputer yang mungkin terpisah oleh jarak yang jauh. (Steen & Tanenbaum, 2023, p. 11).

c. Openness

Tujuan penting lain dari sistem terdistribusi adalah *Openness* (Keterbukaan). Sistem terdistribusi yang terbuka pada dasarnya adalah sistem yang menawarkan komponen-komponen yang dapat dengan mudah digunakan oleh, atau diintegrasikan ke dalam sistem lain. Pada saat yang sama, sistem terdistribusi yang terbuka itu sendiri sering kali terdiri dari komponen-komponen yang berasal dari tempat lain. (Steen & Tanenbaum, 2023, p. 15).

d. Dependability

Dependability mengacu pada sejauh mana sistem komputer dapat diandalkan untuk beroperasi seperti yang diharapkan. Berbeda dengan sistem komputer tunggal, keandalan dalam sistem terdistribusi bisa jadi cukup rumit karena adanya kegagalan parsial, yakni saat ada komponen di suatu tempat yang gagal, sementara sistem secara keseluruhan masih tampak memenuhi ekspektasi (hingga titik atau momen tertentu). (Steen & Tanenbaum, 2023, p. 18).

e. Security

Sistem yang tidak aman sama saja dengan tidak andal. Perhatian khusus diperlukan untuk memastikan integritas dan kerahasiaan sebuah sistem. (Steen & Tanenbaum, 2023, p. 21).

f. Scalability

Scalability menjadi salah satu tujuan desain yang paling penting bagi para pengembang sistem terdistribusi. *Scalability* merupakan kemampuan sistem terdistribusi untuk tumbuh. *Scalability* sebuah sistem dapat diukur dalam setidaknya

tiga hal, *Scalability* Ukuran, *Scalability* Geografis dan *Scalability* Administratif. (Steen & Tanenbaum, 2023, p. 24).

Salah satu tradeoff pada pub-sub agrregator adalah pada aspek ordering. Karena sistem bersifat terdistribusi dengan banyak publisher, penentuan urutan kedatangan data menjadi tantangan. Hal ini dipengaruhi oleh berbagai faktor, termasuk variasi *latency* jaringan serta ketiadaan jam global yang sinkron.

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

   Perbedaan yang terdapat pada arsitektur *Client-Server* dan *Publish-Subcribe* adalah pada relasinya. Terdapat relasi yang jelas antara client dan server pada arsitektur *client-server*, sedangkan pada arsitektur *publish-subscribe*, *publisher* dan *subscriber* terhubung secara tidak langsung melalui *event bus* (van Steen & Tanenbaum, 2023, hlm. 66–70). Contoh dimana *pub-sub* lebih baik dipilih adalah ketika ingin membuat dashboard sensor, karena mendukung komunikasi asynchronous dan skalabilitas horizontal yang memudahkan penambahan sensor dan konsumen baru.

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

   At-least-once delivery menjamin setiap pesan akan dikirim minimal satu kali. Kelebihannya adalah reliabilitas tinggi, tetapi konsekuensinya pesan bisa diterima lebih dari sekali (*duplicates*). Sedangkan Exactly-once delivery berusaha menjamin bahwa setiap pesan diproses tepat satu kali, tanpa kehilangan maupun duplikasi. Exactly-once delivery jauh lebih kompleks dan mahal secara performa karena memerlukan protokol tambahan yang digunakan untuk *deduplication*, *acknowledgment tracking,* dan *kadang distributed transactions*. (van Steen & Tanenbaum, 2017, Bab 4.2, hlm. 154–160). Idempotent consumer krusial di *presence of retries* karena pada skenario *at-least-once delivery* pesan dapat dikirim ulang akibat kegagalan jaringan atau crash, sehingga konsumen berpotensi menerima duplikasi. Dengan sifat idempotensi, konsumen mampu memproses pesan berulang tanpa mengubah hasil akhir, menjaga konsistensi sistem sekaligus menghindari kompleksitas penuh dari *exactly-once delivery*.

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

   topic = {topic}/{location}/{type}-{4-digit-numbers}
   contoh:
   - sales/crm/invoice-3456
   - infra/monitoring/log-4004

   event_id = {topic}/{timestamp_utc}/{uuid}
   contoh
   - sales/crm/invoice-3456/2026-03-01T10:30:00Z/a1b2c3d4-e5f6-7g8h-9i0j-k1l2m3n4o5p6
   - hr/onboarding/employee-0101/2026-03-01T10:30:00Z/b2c3d4e5-f6g7-8h9i-0j1k-l2m3n4o5p6q7

   Hal ini penting untuk dedup karena dengan adanya ID yang unik dan tahan benturan (terutama dari komponen UUID), kemungkinan data baru ditolak karena sudah ada topic dan event_id yang sama dalam dedup store (kesalahan false Positive) menjadi sangat kecil bahkan mungkin tidak ada. Karena Jika topic dan event_id terlalu umum, akan ada kemungkinan data baru yang sebenarnya unik ditolah karena memiliki topic dan event_id yang sama dengan yang sudah ada pada dedup store

5. 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 ketika event yang terjadi tidak saling bergantung atau tidak menimbulkan konflik antar proses (Steen & Tanenbaum, 2023, p. 274). Misalnya, pada sistem logging aplikasi *microservices*, event dari service A (misalnya request autentikasi) dan event dari service B (misalnya query produk) dapat dicatat tanpa harus memiliki urutan global, karena masing-masing hanya relevan di dalam konteks servicenya sendiri. Dalam kasus ini, cukup digunakan *ordering per-partition* atau *per-topic*.

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

Type of Failure	Description	Mitigation
Crash	Halts, but is working correctly until it halts (Steen & Tanenbaum, 2023, p. 467).	Mengimplementasi Checkpoint agar state sebelum crash tersimpan
Timing Failure	Response lies outside a specified time interval (Steen & Tanenbaum, 2023, p. 467).	Menjaga batasan latensi dengan menerapkan *Quality of Service*
Omission Failure	Fails to respond to incoming requests Fails to receive incoming messages Fails to send messages (Steen & Tanenbaum, 2023, p. 467).	Gunakan timeout untuk menghentikan pengiriman pesan dan mekanisme pengiriman ulang untuk mengirim ulang pesan yang hilang.
Response Failure	Response is incorrect The value of the response is wrong Deviates from the correct flow of control (Steen & Tanenbaum, 2023, p. 467).	Gunakan *sanity check* dan *data validation* sebelum memproses hasil

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

   Eventual Consistency merupakan bentuk konsistensi dimana jika tidak ada perubahan dalam waktu yang lama, semua komponen pada suatu waktu akan menjadi konsisten (Steen & Tanenbaum, 2023, p. 407).
   Idempotency dan deduplication berperan dalam mencapai eventual cinsistency karena keduanya memastikan bahwa meskipun terjadi pengiriman berulang atau duplikasi pesan, hasil akhir tetap consisten.

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

Metrics	Deinition	Design
Dupilcate Rate	Persentase event duplikat yang dapat diiterima sistem dibandingkan dengan total semua event yang dikirimkan.	Penggunaan kombinasi (topic, event_id) sebgaai composite key untuk mencegah duplikasi Penggunaan SQLite sebagai dedup store untuk memastikan idempotensi
Latency	Waktu yang dibutuhkan untuk memproses event	Penggunaan database lokal berupa SQLite untuk mengurangi overhead jaringan
Throughput	Jumlah event yang dapat dikelola sistem dalam satu waktu	Pengguinaan asyncio dan Flask agar proses asinkron bisa dilakukan.

Daftar Pustaka

Steen, M. van, & Tanenbaum, A. S. (2023). Distributed Systems (Fourth edition, version 4.01 (January 2023)). Maarten van Steen.