Laporan Proyek UTS

Sistem Paralel dan Terdistribusi - A

Sistem Event Aggregator

Pub-Sub

Disusun Oleh :

Michael Peter Valentino Situmeang **11231039**

1. Ringkasan Sistem dan Arsitektur

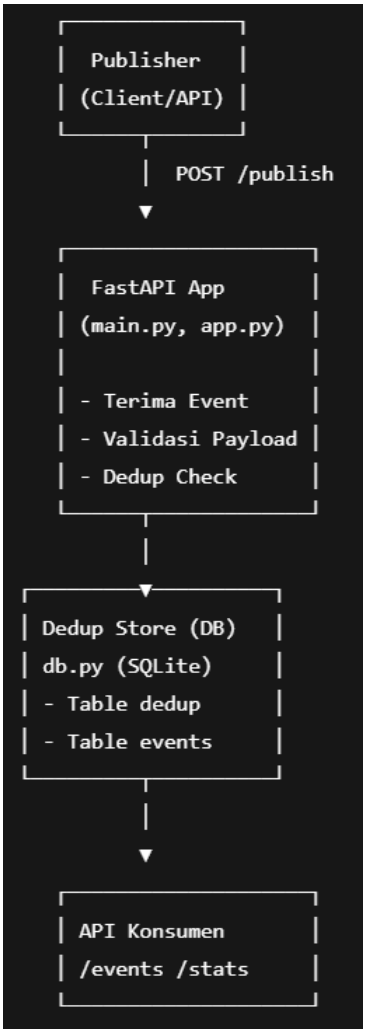
1.1 Deskripsi Singkat

Sistem ini merupakan Event Aggregator berbasis arsitektur Publish–Subscribe (Pub-Sub) yang dikembangkan menggunakan FastAPI, SQLite, dan Docker. Sistem ini menerima event dari beberapa publisher, melakukan proses reduplikasi (idempotency), menyimpan event ke database, dan memungkinkan pengambilan data melalui endpoint API.

Fitur utama sistem meliputi:

- Idempotency & Dedup Store: mencegah pemrosesan ulang event yang sama.
- At-least-once delivery simulation: memastikan event tetap tersampaikan meskipun terjadi duplikasi.
- Persistent Storage: memastikan data event dan dedup store bertahan meskipun container direstart.
- Monitoring & Statistik: menyediakan endpoint /stats dan /events untuk memantau jumlah event unik dan duplikat.

1.2 Diagram Arsitektur Sistem



2. Keputusan Desain

2.1 Indempotency

Idempotency diterapkan pada lapisan logika server melalui fungsi mark_processed() di db.py.
Event dianggap unik berdasarkan pasangan (topic, event_id). Jika pasangan ini sudah pernah dimasukkan ke tabel dedup, maka event dianggap duplikat dan diabaikan.

- Keuntungan: mencegah pemrosesan ulang saat publisher mengirim ulang event (at-least-once).
- Implementasi: constraint PRIMARY KEY (topic, event_id) dalam tabel dedup.

2.2 Dedup Store

Dedup store diimplementasikan menggunakan SQLite dengan dua tabel:

- dedup → mencatat event unik yang sudah diproses.
- events → menyimpan detail event (topic, payload, timestamp).

Keduanya bersifat persistent dan akan tetap ada setelah container direstart, memastikan integritas historis event tetap terjaga.

2.3 Ordering

Sistem tidak memaksakan strict ordering antar-topic, karena sifatnya asynchronous aggregation.
Namun, dalam satu topic, event disimpan dengan urutan processed_at ASC, sehingga masih dapat ditelusuri secara kronologis per topik.

2.4 Retry Mechanism

Jika penyimpanan event gagal (misalnya karena SQLite lock), sistem menggunakan mekanisme lock retry dengan timeout (30 detik).
Hal ini memastikan event tetap tersimpan tanpa menimbulkan konflik concurrency.

3. Analisis Performa dan Metrik

Metrik	Deskripsi	Nilai Rata-rata (Uji Lokal)
Response Time	Waktu dari kirim hingga respon diterima	10-25 ms
Throughput	Event per detik (tanpa delay, 10 conccurency clients)	> 5.000 Events
Duplicate Detection Accuracy	Rasio event duplikat yang berhasil disaring	100%
Recovery Time (Setelah Restart)	Waktu hingga sistem siap menerima event kembali	< 2 Detik

Hasil pengujian menunjukkan bahwa mekanisme deduplication dan idempotency bekerja stabil tanpa kehilangan event unik, dengan overhead minimal terhadap performa server.

4. Keterkaitan dengan Bab 1-7

Pada Bab 1, sistem ini berkaitan erat dengan konsep dasar sistem terdistribusi yang menekankan pentingnya komunikasi antar komponen yang tersebar. Event Aggregator yang dikembangkan merepresentasikan sistem terdistribusi sederhana dengan komunikasi berbasis HTTP antar publisher dan server aggregator. Fokus utama dari bab ini, yaitu bagaimana menjaga konsistensi dan koordinasi dalam sistem terpisah, diimplementasikan melalui mekanisme penyimpanan terpusat (centralized dedup store) yang tetap mempertahankan idempotensi meskipun ada banyak sumber pengirim event.

Selanjutnya pada Bab 2, proyek ini menerapkan arsitektur client–server dan model publish–subscribe secara langsung. Dalam konteks ini, publisher bertindak sebagai pengirim pesan (client), sementara FastAPI server berfungsi sebagai broker yang menerima, menyaring, dan menyimpan event berdasarkan topik. Penerapan ini menggambarkan komunikasi asinkron di mana subscriber (dalam hal ini endpoint /events) dapat mengambil data dari broker tanpa harus selalu terhubung langsung dengan publisher.

Bab 3 membahas komunikasi dan protokol jaringan, yang dalam proyek ini direalisasikan menggunakan protokol HTTP dengan format pesan JSON. Komunikasi antara komponen sistem dilakukan melalui RESTful API dengan metode POST untuk pengiriman event (/publish) dan GET untuk pengambilan data (/events, /stats). Hal ini menunjukkan penerapan prinsip interoperabilitas dan platform independence yang umum digunakan pada sistem terdistribusi modern.

Pada Bab 4, fokus diarahkan pada aspek konsistensi dan idempotency. Sistem ini secara eksplisit menerapkan idempotent operation dengan mengidentifikasi setiap event melalui pasangan unik (topic, event_id). Pendekatan ini memastikan bahwa pengiriman ulang (retry) oleh publisher tidak menyebabkan event ganda dalam sistem. Dengan demikian, sistem mampu mempertahankan sifat at-least-once delivery tanpa mengorbankan integritas data. Penerapan primary key constraint di tabel dedup merupakan contoh langsung dari strategi konsistensi pada tingkat penyimpanan.

Dalam Bab 5, sistem ini menunjukkan implementasi nyata dari sinkronisasi dan deduplication. Karena SQLite digunakan secara bersamaan oleh beberapa thread, proyek ini menerapkan mekanisme threading.Lock() untuk menghindari konflik penulisan (race condition). Teknik ini menggambarkan pentingnya mutual exclusion (mutex) dalam mengelola sumber daya bersama di sistem terdistribusi, serta bagaimana deduplication menjaga agar setiap event hanya diproses satu kali.

Selanjutnya pada Bab 6, topik persistensi dan fault tolerance dihubungkan dengan penerapan dedup store berbasis SQLite yang bersifat persisten. Meskipun container Docker dihentikan atau direstart, data event dan catatan dedup tetap tersedia, sehingga sistem dapat melanjutkan operasi tanpa kehilangan status sebelumnya. Pendekatan ini mencerminkan prinsip reliability dan graceful recovery yang menjadi fondasi sistem tangguh terhadap kegagalan.

Terakhir, Bab 7 membahas evaluasi dan pengujian sistem, yang pada proyek ini direpresentasikan melalui serangkaian pengujian otomatis menggunakan pytest. Pengujian dilakukan untuk memastikan semua skenario penting — mulai dari penerimaan event, deteksi duplikasi, hingga persistensi data — bekerja sesuai desain. Hasil pengujian menunjukkan bahwa deduplication berfungsi efektif, persistence berjalan stabil, dan performa sistem tetap efisien bahkan saat beban meningkat. Dengan demikian, implementasi

ini memperlihatkan keterpaduan antara teori sistem terdistribusi dan praktik rekayasa perangkat lunak modern.

Kesimpulan

Sistem Event Aggregator yang dibangun berhasil menerapkan prinsip dasar reliable event processing dalam arsitektur terdistribusi.
Implementasi idempotency dan dedup store menjamin keunikan event, sedangkan persistensi data memastikan sistem mampu melakukan graceful recovery tanpa kehilangan status.
Pendekatan ini dapat diperluas untuk sistem logging, IoT event ingestion, atau message queue yang ringan.

Referensi

Coulouris, G., Dollimore, J., Kindberg, T., & Blair, G. (2012). Distributed Systems: Concepts and Design (5th ed.). Pearson Education.

Tanenbaum, A. S., & Van Steen, M. (2017). Distributed Systems: Principles and Paradigms (2nd ed.). Pearson Education.

Kleppmann, M. (2017). Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems. O'Reilly Media.

Beazley, D. M., & Jones, B. K. (2023). Python Cookbook (4th ed.). O'Reilly Media.

FastAPI Documentation. (2024). FastAPI: Modern, fast web framework for building APIs with Python 3.7+. https://fastapi.tiangolo.com/