Nama : Alpian Roymundus Siringo-ringo
NIM : 11211009
Sistem Terdistribusi A

Kasus 1:

Sinkronisasi Basis Data Terdistribusi Sebuah bank internasional menggunakan basis data terdistribusi di beberapa wilayah geografis. Sistem ini harus menjamin bahwa pembaruan di satu wilayah akan terlihat di wilayah lain tanpa mengorbankan konsistensi.

a. Metode Sinkronisasi untuk Konsistensi Eventual Tanpa Mengorbankan Ketersediaan

   Untuk konsistensi eventual tanpa mengorbankan ketersediaan, pendekatan seperti algoritma Lamport sangat relevan. Lamport's Logical Clocks digunakan untuk mengurutkan kejadian tanpa harus menyinkronkan clock fisik, yang penting dalam sistem yang tersebar. Dalam kasus bank, ini bisa diterapkan dengan menggunakan clock logika yang memastikan semua pembaruan akhirnya akan mencapai konsistensi di seluruh wilayah meskipun pada waktu yang berbeda

b. Penanganan Konflik di Saat Dua Wilayah Memperbarui Data yang Sama Secara Bersamaan

   Untuk basis data terdistribusi bank, dapat digunakan algoritma seperti Lamport’s Happened-Before Relation untuk menentukan urutan kejadian. Ketika dua wilayah memperbarui data yang sama, algoritma ini bisa menentukan urutan waktu sehingga konflik bisa dicegah atau diselesaikan dengan menetapkan kejadian yang terjadi lebih dulu sebagai prioritas.

   Selain itu, algoritma distributed mutual exclusion dapat digunakan, di mana setiap pembaruan dikirim ke semua node, dan setiap node harus setuju untuk memberikan akses ke critical region. Ini bisa diterapkan untuk menghindari konflik di sistem bank, meskipun ada latensi komunikasi antarwilayah

c. Rekomendasi Sistem Penamaan (Naming System) untuk Basis Data Terdistribusi

   Hierarchical naming seperti dalam DNS cocok untuk sistem ini, dengan setiap wilayah mendapatkan prefiks unik, sehingga data dapat diambil lebih efisien. Misalnya, menggunakan Universal Resource Names (URNs) atau Global Unique Identifiers (GUIDs) memungkinkan data untuk diakses tanpa ambigu dan konflik. Dengan sistem penamaan yang efisien, seperti dalam DHT (Distributed Hash Table), pencarian data dapat dilakukan dengan cepat karena setiap node bertanggung jawab atas subset tertentu dari data berdasarkan hash.

   Secara keseluruhan, metode sinkronisasi yang efektif dalam materi ini menunjukkan bagaimana sinkronisasi clock logika dan algoritma mutual exclusion tersebar dapat diaplikasikan untuk memastikan konsistensi dan menangani konflik dalam basis data terdistribusi pada bank internasional.

Kasus 2:

Platform Streaming Video Terdistribusi Sebuah perusahaan membangun platform streaming video terdistribusi yang memungkinkan pengguna untuk menonton konten dari server yang berlokasi di berbagai wilayah di seluruh dunia. Mereka ingin memastikan bahwa pengalaman pengguna lancar tanpa masalah latensi.

a. Sistem Penamaan untuk Akses Konten dari Server Terdekat
   i. DNS Geolocation Routing:

      Sistem ini memungkinkan pengguna dialihkan ke server terdekat secara otomatis. Ketika pengguna melakukan request untuk sebuah video, DNS secara dinamis mengarahkan mereka ke server yang paling dekat secara geografis berdasarkan alamat IP pengguna. Nama domain bisa diatur dalam hirarki wilayah, misalnya video-us.server.com untuk pengguna di AS, atau video-eu.server.com untuk Eropa.
   ii. Anycast Routing:

      Dengan Anycast, alamat IP tunggal dipetakan ke beberapa lokasi server. Pengguna secara otomatis diarahkan ke server terdekat yang menyediakan alamat IP yang sama, sehingga latensi dapat diminimalkan.

b. Tantangan dalam Pengelolaan Proses dan Thread pada Server di Lokasi Geografis Berbeda

   Load Balancing:

   Tantangan utama adalah memastikan beban distribusi server merata, terlepas dari lokasi geografis. Ketidakseimbangan beban terjadi jika satu wilayah memiliki lebih banyak pengguna aktif daripada yang lain, menyebabkan beberapa server overload sementara yang lain tidak digunakan sepenuhnya.

   Solusi: Penggunaan Global Load Balancers yang mendistribusikan lalu lintas ke berbagai server berdasarkan ketersediaan sumber daya (CPU, RAM), latensi jaringan, dan waktu respons. Sistem juga dapat menggunakan dynamic load balancers untuk memantau lalu lintas secara real-time dan menyesuaikan beban.

c. Solusi untuk Sinkronisasi Real-time Konten Video Antar Server
   i. Real-time Data Replication (Eventual Consistency):

      Data replication di antara server dapat dilakukan menggunakan protokol seperti Apache Kafka atau Pulsar yang memungkinkan sinkronisasi event-driven antar server. Dengan pendekatan eventual consistency, setiap server di wilayah geografis yang berbeda akan menerima perubahan terbaru secara asinkron tetapi dalam waktu yang sangat singkat.
   ii. Clock Synchronization with NTP (Network Time Protocol):

      Sinkronisasi konten antar server bisa dilakukan dengan sinkronisasi clock menggunakan

NTP. Dengan memastikan waktu di setiap server terdistribusi tetap sinkron, setiap perubahan atau pembaruan video dapat diterapkan dengan tepat waktu dan seragam di seluruh wilayah.

Kasus 3:

Penerapan Docker untuk Sistem Microservices Sebuah perusahaan perangkat lunak sedang beralih ke arsitektur microservices dan berencana menggunakan Docker untuk menyebarkan aplikasi mereka. Mereka belum familiar dengan Docker dan meminta bantuan Anda.

a. Keuntungan Menggunakan Docker dalam Sistem Terdistribusi untuk Microservices
   i. Isolasi Layanan: Docker memungkinkan setiap microservice dijalankan dalam kontainer terpisah. Isolasi ini memastikan bahwa satu layanan tidak mempengaruhi layanan lainnya, yang penting dalam sistem terdistribusi.
   ii. Portabilitas: Docker membuat aplikasi portabel, sehingga dapat dijalankan di berbagai platform (development, testing, dan production) dengan konsistensi yang sama. Ini penting dalam sistem terdistribusi, di mana aplikasi harus dijalankan di berbagai lingkungan dan server.
   iii. Skalabilitas: Dengan Docker, setiap microservice dapat di-scale secara independen sesuai kebutuhan. Ini memudahkan untuk menambah instans baru dari microservice tanpa mempengaruhi layanan lainnya, yang penting dalam sistem terdistribusi berbasis microservices.
b. Perbedaan Pengelolaan Proses dan Thread dalam Docker Dibandingkan dengan Mesin Virtual Tradisional
   i. Ringan dan Efisien: Tidak seperti mesin virtual (VM) yang menjalankan seluruh OS di atas hypervisor, Docker berjalan langsung di atas kernel host menggunakan containerization. Setiap kontainer hanya mencakup aplikasi dan dependensinya, yang membuat Docker jauh lebih ringan dibandingkan VM. VM mengisolasi seluruh sistem operasi, sementara Docker mengisolasi hanya proses yang berjalan di dalamnya, sehingga Docker lebih hemat sumber daya.
   ii. Startup Time: Docker memulai kontainer jauh lebih cepat daripada VM. Karena VM harus boot seluruh OS, proses ini bisa memakan waktu beberapa menit. Sebaliknya, kontainer Docker biasanya hanya membutuhkan beberapa detik karena hanya memulai aplikasi dan dependensinya.
   iii. Overhead Minimal: VM membawa overhead yang lebih besar karena memerlukan emulasi hardware, sementara Docker tidak memerlukan ini, karena berbagi kernel yang sama dengan host. Ini mengurangi overhead CPU dan memori dalam Docker.
c. Pengelolaan Komunikasi Antar Sistem Antar Kontainer Docker

i. Docker Networks: Docker memiliki fitur jaringan bawaan yang memungkinkan komunikasi antar kontainer secara efisien. Anda bisa membuat jaringan khusus untuk aplikasi Anda, di mana setiap kontainer yang berada dalam jaringan yang sama dapat saling berkomunikasi dengan menggunakan nama kontainer mereka.

ii. Service Discovery dengan Docker Swarm/Kubernetes: Dalam arsitektur microservices, layanan harus bisa menemukan satu sama lain secara dinamis. Docker Swarm atau Kubernetes menawarkan service discovery, di mana layanan dapat diidentifikasi dan diakses menggunakan nama layanan. Ini mempermudah komunikasi antar kontainer yang berjalan di berbagai host.

iii. Message Brokers (Kafka, RabbitMQ): Untuk komunikasi antar kontainer secara asinkron dalam arsitektur microservices, message brokers seperti Kafka atau RabbitMQ sering digunakan. Kontainer bisa berkomunikasi secara tidak langsung dengan cara mengirim pesan melalui broker, memastikan bahwa komunikasi tetap efisien meskipun layanan terpisah secara geografis atau tidak berjalan pada waktu yang sama.