Anggota Kelompok:

- Oktario Mufti Yudha (2320506044)
- Adetia Raymond S. (2320506036)
- Muhammad Ridwan A. (2320506049)

Studi kasus yang dipilih adalah Cloud Computing yaitu AWS (Amazon Web Service)

Deskripsi Sistem:

Amazon Web Services (AWS) menggabungkan arsitektur sistem terdistribusi di banyak layanan cloud-nya untuk meningkatkan kinerja, skalabilitas, dan keandalan. Prinsip desain ini mendasar dalam memenuhi tuntutan penanganan tugas pemrosesan dan manajemen data berskala besar secara efisien. Dengan mendistribusikan operasi di beberapa server dan lokasi, AWS memastikan bahwa layanannya tetap tersedia dan tangguh terhadap kegagalan.

Contoh layanan berikut menunjukkan bagaimana AWS menggunakan sistem terdistribusi untuk menyediakan solusi yang dapat diskalakan, andal, dan efisien:

- **Amazon S3 (Simple Storage Service):** S3 dibuat untuk menyimpan dan mengambil sejumlah data dari mana saja di web. Layanan ini menggunakan arsitektur terdistribusi untuk menangani sejumlah besar data dan konkurensi tinggi, memastikan ketersediaan dan ketahanan data yang kuat.

- **Amazon EC2 (Elastic Compute Cloud):** EC2 menyediakan kapasitas komputasi yang dapat diskalakan di cloud dengan memanfaatkan infrastruktur global AWS yang luas. Layanan ini memungkinkan pengguna untuk meluncurkan server virtual sesuai kebutuhan, mengoptimalkan pemanfaatan sumber daya dan fleksibilitas operasional. Amazon DynamoDB: DynamoDB adalah layanan basis data NoSQL yang cepat dan fleksibel yang dirancang untuk semua aplikasi yang memerlukan latensi milidetik tunggal yang konsisten dalam skala apa pun. Layanan ini mempertahankan arsitektur basis data yang terdistribusi secara otomatis, yang memfasilitasi akses data yang cepat dan skalabilitas.

- **Amazon RDS (Relational Database Service):** RDS memudahkan pengaturan, pengoperasian, dan penskalaan basis data relasional di cloud. Layanan ini menyediakan kapasitas yang hemat biaya dan dapat diubah ukurannya sekaligus mengotomatiskan tugas-tugas administrasi yang memakan waktu seperti penyediaan perangkat keras, pengaturan basis data, pembuatan patch, dan pencadangan.

Kekurangan Sistem Terdistribusi

Berikut adalah beberapa kekurangan dari sistem terdistribusi:

- **Tantangan Konsistensi Data**
  - **Sinkronisasi**: Memastikan semua node memiliki pandangan data yang konsisten bisa menjadi sulit, terutama jika terjadi pemisahan atau keterlambatan jaringan.
  - **Model Konsistensi**: Node yang berbeda mungkin memiliki tingkat konsistensi data yang bervariasi, yang dapat menyebabkan konflik atau informasi yang usang.
- **Masalah Keamanan**
  - **Perlindungan Data**: Mengamankan data yang dikirim melalui jaringan dan disimpan di beberapa node bisa menjadi kompleks serta memerlukan enkripsi dan kontrol akses yang kuat.
  - **Otentikasi dan Otorisasi**: Mengelola otentikasi dan otorisasi di antara komponen yang terdistribusi lebih menantang dibandingkan sistem terpusat.
- **Peningkatan Overhead**
  - **Overhead Komunikasi**: Komunikasi antar-node menambah latensi dan mengonsumsi bandwidth, yang dapat memengaruhi kinerja sistem secara keseluruhan.
  - **Manajemen Sumber Daya**: Mengelola status dan mengalokasikan sumber daya di antara node terdistribusi dapat menambah kompleksitas dan overhead.
- **Masalah Integritas Data**
  - **Keterlambatan Replikasi**: Replikasi data di antara node dapat menyebabkan inkonsistensi jika pembaruan tidak disebarkan secara cepat.
  - **Penyelesaian Konflik**: Menangani konflik data dan memastikan integritas data dalam kondisi pembaruan yang bersamaan bisa menjadi sangat kompleks.

Solusi yang bisa diterapkan

- **Menjamin Konsistensi Data**
  - **Algoritma Konsensus:** Gunakan algoritma seperti Paxos atau Raft untuk memastikan semua node memiliki data yang sinkron dan konsisten.
  - **Tunable Consistency:** Terapkan model konsistensi yang bisa diatur sesuai kebutuhan, sehingga trade-off antara konsistensi dan ketersediaan dapat dioptimalkan.
  - **Mekanisme Resolusi Konflik:** Rancang strategi untuk mendeteksi dan menyelesaikan konflik data yang terjadi akibat pembaruan simultan.
- **Memperkuat Keamanan**
  - **Enkripsi dan Proteksi Data:** Terapkan enkripsi end-to-end untuk data yang dikirim dan disimpan, serta gunakan protokol keamanan yang ketat.

    - **Manajemen Akses:** Gunakan otentikasi multi-faktor, kontrol akses berbasis peran (RBAC), dan audit log untuk memastikan hanya entitas yang berwenang yang dapat mengakses sistem.
- **Mengurangi Overhead**
    - **Optimasi Komunikasi:** Implementasikan kompresi data dan optimasi protokol komunikasi antar-node untuk mengurangi latensi dan penggunaan bandwidth.
    - **Manajemen Sumber Daya:** Gunakan teknologi container orchestration (misalnya Kubernetes) dan auto-scaling untuk mendistribusikan beban kerja secara efisien di antara node.
- **Memastikan Integritas Data**
    - **Replikasi Data Real-Time:** Gunakan sistem replikasi yang cepat dan konsisten untuk memastikan data selalu up-to-date di seluruh node.
    - **Monitoring dan Audit:** Implementasikan monitoring berkelanjutan untuk mendeteksi inkonsistensi data dan segera men-trigger mekanisme penyelesaian konflik, sehingga integritas data tetap terjaga.

Evaluasi Solusi

- **Menjamin Konsistensi Data**

  Kelebihan:

    - **Algoritma Konsensus (Paxos/Raft):**
        - **Kelebihan:** Memastikan bahwa semua node memiliki data yang sinkron dan konsisten, yang secara langsung meningkatkan integritas dan keandalan sistem.
    - **Tunable Consistency:**
        - **Kelebihan:** Memberikan fleksibilitas untuk mengatur tingkat konsistensi sesuai dengan kebutuhan aplikasi, sehingga memungkinkan trade-off yang optimal antara konsistensi dan ketersediaan.
    - **Mekanisme Resolusi Konflik:**
        - **Kelebihan:** Dapat mendeteksi dan menyelesaikan konflik data secara otomatis, sehingga meminimalkan gangguan operasional akibat pembaruan simultan.

  Kekurangan:

    - **Kompleksitas Implementasi:**
        - Algoritma konsensus seperti Paxos atau Raft umumnya kompleks dan membutuhkan overhead komunikasi yang tinggi, sehingga dapat mempengaruhi performa secara keseluruhan.
    - **Trade-off pada Tunable Consistency:**

    - Pengaturan tingkat konsistensi yang dapat disesuaikan memerlukan pemahaman mendalam mengenai kebutuhan aplikasi dan dapat menimbulkan kebingungan atau kesalahan konfigurasi.
- **Kompleksitas Resolusi Konflik:**
    - Mekanisme untuk menyelesaikan konflik data harus sangat cermat. Jika tidak, bisa terjadi kesalahan atau konflik yang tidak tertangani dengan baik pada situasi pembaruan simultan yang ekstrem.

- **Memperkuat Keamanan**

  Kelebihan:

  - **Enkripsi dan Proteksi Data:**
    - **Kelebihan:** Dengan menerapkan enkripsi end-to-end, data yang dikirim dan disimpan menjadi lebih aman dari akses tidak sah.
  - **Manajemen Akses:**
    - **Kelebihan:** Penggunaan otentikasi multi-faktor, kontrol akses berbasis peran (RBAC), dan audit log meningkatkan keamanan dengan memastikan bahwa hanya entitas yang berwenang yang dapat mengakses sistem.

  Kekurangan:

  - **Overhead Performa:**
    - Proses enkripsi dan dekripsi dapat menambah beban kerja pada sistem, yang berdampak pada kinerja terutama jika volume data besar.
  - **Kompleksitas Administrasi:**
    - Implementasi otentikasi multi-faktor dan kontrol akses yang ketat bisa menambah kompleksitas operasional dan terkadang mempengaruhi pengalaman pengguna (user experience).
- **Mengurangi Overhead**

  Kelebihan:

  - **Optimasi Komunikasi:**
    - **Kelebihan:** Teknik seperti kompresi data dan optimasi protokol komunikasi dapat mengurangi latensi dan penggunaan bandwidth, sehingga meningkatkan performa sistem secara keseluruhan.
  - **Manajemen Sumber Daya dengan Teknologi Orkestrasi (Kubernetes & Auto-Scaling):**
    - **Kelebihan:** Memungkinkan distribusi beban kerja secara dinamis dan efisien, sehingga membantu sistem menyesuaikan kapasitas dengan permintaan yang aktual.

  Kekurangan:

- **Kompleksitas Konfigurasi:**
    - Optimasi komunikasi memerlukan penyesuaian yang terus menerus terhadap kondisi jaringan yang dinamis, yang bisa menambah kompleksitas pemeliharaan.
- **Keahlian Teknis:**
    - Penggunaan platform orkestrasi seperti Kubernetes memerlukan keahlian khusus dalam manajemen dan pemeliharaan cluster, sehingga bisa meningkatkan kebutuhan sumber daya manusia dan pelatihan.

- **Memastikan Integritas Data**

  Kelebihan:

  - **Replikasi Data Real-Time:**
    - **Kelebihan:** Memastikan bahwa data di semua node selalu up-to-date, sehingga meningkatkan keandalan dan konsistensi informasi di seluruh sistem.
  - **Monitoring dan Audit:**
    - **Kelebihan:** Dengan adanya monitoring berkelanjutan, sistem dapat dengan cepat mendeteksi inkonsistensi data dan memicu mekanisme penyelesaian konflik, sehingga menjaga integritas data.

  Kekurangan:

  - **Beban Jaringan dan Proses:**
    - Replikasi data real-time dapat meningkatkan beban jaringan dan memerlukan sumber daya tambahan untuk pemrosesan, yang dapat menurunkan kinerja sistem jika tidak dikelola dengan baik.
  - **Investasi dan Kompleksitas Monitoring:**
    - Implementasi sistem monitoring dan audit yang efektif memerlukan investasi dalam perangkat lunak dan sumber daya manusia untuk analisis data secara terus menerus, yang dapat meningkatkan biaya operasional.

Kesimpulan

Secara keseluruhan, solusi-solusi yang diterapkan memberikan manfaat besar dalam meningkatkan kinerja, keandalan, dan keamanan sistem terdistribusi. Namun, keberhasilan implementasinya sangat bergantung pada kemampuan untuk mengelola kompleksitas, overhead, dan biaya operasional. Oleh karena itu, pemilihan serta penerapan solusi harus disesuaikan dengan kebutuhan spesifik aplikasi dan kondisi operasional untuk mencapai keseimbangan optimal antara keandalan, performa, dan efisiensi biaya.

Referensi

- GeeksforGeeks. (n.d.). *Does AWS use distributed systems?* Retrieved from https://www.geeksforgeeks.org/does-aws-use-distributed-systems/#aws-services-built-on-distributed-systems

- GeeksforGeeks. (n.d.). *Advantages and disadvantages of distributed systems*. Retrieved from https://www.geeksforgeeks.org/advantages-and-disadvantages-of-distributed-systems/