Manajemen Sumber Daya dalam Lingkungan Terdistribusi adalah sistem pengelolaan sumber daya seperti file data lainnya melalui

Sistem terdistribusi yang tujuan utamanya adalah memastikan bahwa pengguna/klien dapat mengakses sumber daya jarak jauh dengan kemudahan yang dapat diaksesnya sumber daya lokal.

Penyeimbangan beban adalah aspek penting dari sistem terdistribusi yang bertujuan untuk mengoptimalkan pemanfaatan sumber daya, meminimalkan waktu respons, dan menghindari kelebihan beban server.

Namun, beberapa tantangan muncul dalam merancang dan memilih mekanisme penyeimbangan beban yang tepat untuk sistem terdistribusi berdasarkan jaringan area lokal atau komputasi awan. Pendekatan realistis terhadap strategi penyeimbangan beban diperlukan dalam mengatasi distribusi beban kerja yang tidak seimbang, peningkatan latensi dan overhead, masalah skalabilitas, dan kompleksitas dalam merancang algoritma penyeimbangan beban.

Selain itu, penerapan praktik terbaik seperti metrik pemantauan yang tepat, penyeimbang beban berbasis cloud, redundansi, mekanisme failover, tinjauan rutin, dan optimalisasi strategi dapat memastikan penyeimbangan beban yang efektif. Penting untuk selalu mengetahui perkembangan terkini dalam algoritma penyeimbangan beban untuk mencapai ketersediaan tinggi dan kinerja sistem yang optimal.

Sistem terdistribusi adalah proses pengaturan dan pengelolaan semua sumber daya yang tersedia dalam lingkungan yang terdiri dari beberapa komputer atau server yang saling terhubung.

Tujuan utama manajemen sumber daya adalah untuk memastikan penggunaan sumber daya yang efisien dan optimal, sehingga sistem dapat berjalan dengan baik.

Terdapat beberapa manajemen sumber daya, yaitu :

1. Manajemen Sumber Daya Proses

2. Manajemen Sumber Daya Jaringan

3. Manajemen Beban dan Manajemen Failover

4. Manajemen Sumber Daya Penyimpanan

5. Manajemen Sumber Daya Terkait Keamanan

Sistem terdistribusi memiliki beberapa kelebihan, seperti bebas untuk menambah file atau sumber daya tanpa diketahui oleh user (dalam rangkan meningkatkan kinerja), concurrency transparency, dan failure transparency.

Beberapa tantangan dalam sistem terdistribusi meliputi latensi, penskalaan, pemahaman atas API, kegagalan dalam sistem terdistribusi dapat terjadi dalam banyak cara, dan kesulitan dalam membangun perangkat lunak.

Kelemahan pada sistem terdistribusi adalah kesulitan dalam membangun perangkat lunak.

1. **Pengertian Terdistribusi**

Sistem Terdistribusi adalah Sekumpulan komputer otonom yang terhubung ke suatu jaringan, dimana bagi pengguna sistem terlihat sebagai satu komputer. Maksud komputer otonomi adalah walaupun komputer tidak terhubung ke jaringan, komputer tersebut tetap data berjalan. Dengan menjalankan sistem terdistribusi, komputer dapat melakukan :

1. **Keuntungan Sistem Terdistribusi**
2. Performance

Kumpulan dari beberapa prosesor akan memberikan kinerja yang lebih baik dari pada komputer yang terpusat. Begitu juga kalau dilihat dari sisi biaya.

1. Distribution

Reliability (Fault tolerance) apabila salah satu komponen terjadi kerusakan, system tetap dapat berjalan

1. Incremental Growth

Mudah dalam melakukan penambahan komputer/komponen

1. Permasalahan dalam Sistem Terdistribusi
2. **Karakteristik Sistem Terdistribusi**

Ada beberapa hal yang harus diperhatikan dalam membangun sistem terdistribusi, yaitu :

1. Transparency (Kejelasan)
2. Communication (Komunikasi)
3. Performance & Scalability (Kinerja dan Ruang Lingkup)Heterogeneity (Keanekaragaman)
4. Opennes (Keterbukaan)
5. Reliability & Fault Tolerancy (Kehandalan dan Toleransi Kegagalan)
6. **RPC**

Remote Procedure Call (RPC) adalah teknologi komunikasi yang digunakan oleh suatu program untuk membuat permintaan ke program lain untuk memanfaatkan layanannya di suatu jaringan tanpa mengetahui detail jaringan tersebut. Panggilan fungsi atau panggilan subrutin adalah istilah lain untuk panggilan prosedur.

Hal ini didasarkan pada konsep client-server. Klien adalah program yang membuat permintaan, dan server adalah program yang memberikan layanan. RPC, seperti panggilan prosedur lokal, didasarkan pada operasi sinkron yang mengharuskan aplikasi yang meminta dihentikan hingga proses jarak jauh mengembalikan hasilnya. Beberapa RPC dapat dieksekusi secara bersamaan dengan memanfaatkan proses ringan atau thread yang berbagi ruang alamat yang sama. Program Panggilan Prosedur Jarak Jauh sesering mungkin menggunakan Interface Definition Language (IDL), suatu bahasa determinasi untuk mendeskripsikan Application Programming Interface (API) komponen program komputer. Dalam keadaan ini, IDL bertindak sebagai antarmuka antara mesin di kedua ujung koneksi, yang mungkin menjalankan sistem operasi dan bahasa pemrograman berbeda.

Prosedur Kerja Model RPC:

Argumen proses ditempatkan di lokasi yang tepat oleh pemanggil ketika prosedur perlu dipanggil.

Kontrol kemudian diteruskan ke bagian utama metode, yang memiliki serangkaian instruksi.

Badan prosedur dijalankan di lingkungan eksekusi yang baru dibuat yang memiliki duplikat argumen instruksi pemanggil.

Pada akhirnya, setelah operasi selesai, titik panggilan mendapatkan kembali kontrol, yang mengembalikan hasilnya.

Pemanggilan suatu prosedur hanya dimungkinkan untuk prosedur-prosedur yang tidak berada dalam ruang alamat pemanggil karena kedua proses (penelepon dan yang dipanggil) mempunyai ruang alamat yang berbeda dan aksesnya dibatasi pada data lingkungan pemanggil dan variabel-variabel dari prosedur jarak jauh.

Proses pemanggil dan penerima panggilan di RPC berkomunikasi untuk bertukar informasi melalui skema penyampaian pesan.

Tugas pertama dari sisi server adalah mengekstrak parameter prosedur ketika pesan permintaan masuk, lalu hasilnya, mengirim pesan balasan, dan terakhir menunggu pesan panggilan berikutnya.

Hanya satu proses yang diaktifkan pada titik waktu tertentu.

Penelepon tidak selalu harus diblokir.

Mekanisme asinkron dapat digunakan di RPC yang memungkinkan klien untuk bekerja meskipun server belum merespons.

Untuk menangani permintaan masuk, server mungkin membuat thread yang membebaskan server untuk menangani permintaan konsekuen.