MAKALAH

SISTEM TERDISTRUBUSI

Disusun Oleh :

Rizki Apriliyandi

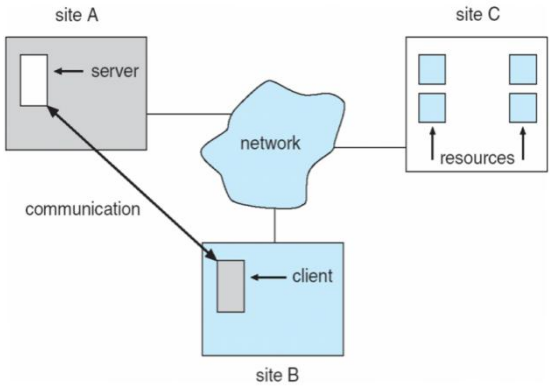
10108823

IF-15

1. Konsep Dasar Sistem Terdistribusi

Sistem terdistribusi berasal dari dua kata, Sistem dan Terdistribusi. Sistem merupakan sekumpulan elemen yang saling tergantung satu sama lain dan membentuk suatu kesatuan untuk menyelesaikan suatu tujuan secara spesifik menjalankan suatu fungsi. Terdistribusi berasal dari kata "sebar/distribusi" yang merupakan kebalikan dari kata "sentralisasi", yang artinya penyebaran, sirkulasi, penyerahan pembagian menjadi bagian-bagian yang lebih kecil.

Jadi sistem terdistribusi merupakan sekumpulan komputer otonom (elemen-elemen) yang saling berinteraksi secara sistematis dan teratur untuk mendistribusikan data, informasi, proses, objek dan layanan dari dan kepada pengguna yang terkait didalamnya sehingga menghasilkan satu fasilitas komputasi terpadu.



Beberapa alasan diterapkan Sistem Terdistribusi antara lain:

a. Berbagi pakai sumber daya, akses terhadap sumber daya jarak jauh sama dengan sumber daya lokal.
   - Berbagi pakai printer dan file dari jarak jauh (remote site)
   - Pemrosesan informasi dalam sistem basis data tersebar
   - Melakukan prose remote terhadap perangkat tertentu.

b. Mengatasi bottleneck, untuk mempercepat proses komputasi.

c. Reliability, melakukan proses penanganan dalam mendeteksi dan menangani kesalahan pada lokasi tertentu, proses pengiriman, dan dan kegagalan re-integrasi lokasi.

d. Komunikasi, dengan SisTer memungkinan mampu melakukan proses pengiriman pesan jauh lebih baik.

Sistem Tersebar memiliki dua tipe :

a. Network Operating System
b. Sistem Operasi Tersebar

Tujuan Sistem terdistribusi adalah :

- Untuk memberikan askes bagi pengguna unruk dapat mengembangkan sumber daya sistem
- Peningkatan kecepatan komputasi
- Meningkatkan ketersediaan dan reliabilitas data.

2. Konsep Proses dan Komunikasi Sistem Terdistribusi pada Linux

- **CORBA *(Common Object Request Broker Architecture)***

Dalam konteks sistem komputer terdistribusi, meskipun komponen-komponen aplikasi dibuat dengan bahasa pemrograman yang berbeda, menggunakan development tools yang berbeda, dan beroperasi di lingkungan yang beragam, mereka tetap harus dapat saling bekerjasama.

CORBA adalah sebuah arsitektur software yang berbasis pada teknologi berorientasi obyek atau *Object Oriented* (OO) dengan paradigma *client-server*. Dalam terminologi OO, sebuah obyek berkomunikasi dengan obyek lain dengan cara pengiriman pesan (message passing). Konteks komunikasi ini kemudian dipetakan ke dalam model *client-server*: satu obyek berperan sebagai *client* (si pengirim pesan) dan yang lain bertindak sebagai *server* (yang menerima pesan dan memroses pesan yang bersangkutan). Sebagai contoh, dalam ilustrasi di awal tulisan ini, jika si pasien memerlukan obat tertentu, maka obyek aplikasi di tempat praktek dokter berlaku sebagai *client* dan mengirim pesan ke obyek aplikasi di apotik guna mengetahui apakah obat yang diperlukan tersedia di sana.

Keunikan dari CORBA adalah kemampuannya dalam menangani heterogenitas antara *client* dan *server* (dalam terminologi CORBA, obyek *server* dinamakan **implementasi obyek** *(object implementation)*. Keduanya dapat saja diimplementasikan dalam hardware, sistem operasi, bahasa pemrograman, dan di lokasi yang berbeda, tetapi tetap bisa saling berkomunikasi.

Kuncinya ada pada sebuah lapisan software yang disebut dengan **ORB***(Object Request Broker)*.

Sistem terdistribusi merupakan sesuatu yang amat kompleks, apalagi jika ruang lingkupnya sangat luas (pada level *enterprise* misalnya). CORBA *(Common Object Request Broker Architecture)* membantu menyederhanakan persoalan dengan menyembunyikan berbagai detail pekerjaan pada level rendah dan heterogenitas sistem dan *platform*. Lebih lanjut, arsitektur OMA menyediakan sebuah *framework* pengembangan sistem terdistribusi yang konsisten, sehingga heterogenitas tetap dapat dikelola dengan baik. Dengan semakin banyaknya pemain teknologi informatika yang terjun ke dunia CORBA, agaknya contoh ilustrasi di awal tulisan ini dalam waktu yang relatif tidak terlalu lama akan dapat direalisasikan.

- **CORBA di Linux**

Dewasa ini cukup banyak perangkat pengembangan berbasis CORBA yang dapat dijalankan di sistem operasi Linux. Hampir semua paket hanya mendukung satu pemetaan bahasa saja, kecuali paket **Inter-Language Unification** (ILU) dari Xerox PARC yang mendukung beberapa bahasa sekaligus (ANSI C, C++, Python, Java, dsb). Tapi konsep ILU sendiri agak berbeda dengan CORBA, karena fokusnya adalah pada integrasi pada level bahasa pemrograman. Meskipun demikian, pendekatannya mirip, bahkan interface pada ILU dapat pula dispesifikasikan dengan menggunakan IDL.

Beberapa contoh perangkat pengembangan berbasis CORBA yang berjalan di Linux antara lain: **MICO** dari mico.org (bahasa yang didukung: C++), **Fnorb** dari DSTC, Australia (Python), **JacORB** oleh Gerard Brose dari Freie Universitat, Berlin (Java), **OmniORB2** dari AT&T (C++), serta tak ketinggalan pula **ORBit** keluaran laboratorium riset RedHat (mendukung bahasa C) yang dipakai dalam proyek Gnome.

TAO (The ACE ORB) dari Washington University adalah implementasi ORB yang dikembangkan dengan pendekatan yang berbeda. TAO tidak semata-mata merupakan sistem ORB sederhana, tetapi ia dirancang untuk bekerja pada lingkungan *real-time* dengan batasan-batasan *(constraints)* yang lebih ketat dibandingkan dengan sistem terdistribusi biasa. Konsekuensinya TAO lebih

memfokuskan diri pada dukungan terhadap aspek *real-time* dan koneksi berkecepatan tinggi, yang diimplementasikan ke dalam arsitektur inti ORB dan modul-modul pendukungnya.

Jika anda memiliki lingkungan komputasi terdistribusi di rumah, di kantor, atau di sekolah, anda bisa mencoba melakukan pemrograman sistem terdistribusi berbasis CORBA. Yang perlu anda lakukan adalah men-*download* salah satu perangkat pengembangan di atas, memasangnya di sistem anda, dan mengikuti petunjuk/ tutorial pemrograman yang diberikan. Jika petunjuk yang ada belum cukup memadai, anda bisa mencari tutorial pemrograman CORBA melalui Internet.

- **Proses pada Linux**

Locking yang paling umum digunakan dalam Linux adalah spin lock. Spin lock adalah lock yang hanya dapat dilakukan oleh satu thread. Ketika sebuah thread yang akan dijalankan meminta spin lock yang sedang digunakan, maka thread ini akan loops menunggu sampai spin lock tersebut selesai digunakan oleh thread yang sedang berjalan. Semafor dalam Linux adalah sleeping locks. Ketika sebuah thread meminta semafor yang sedang digunakan, maka semafor akan meletakkan thread tersebut dalam wait queue dan menyebabkan thread tersebut masuk status sleep.

Symmetrical multiprocessing (SMP) mendukung adanya pengeksekusian secara paralel dua atau lebih thread oleh dua atau lebih processor. Kernel Linux 2.0 adalah kernel Linux pertama yang memperkenalkan konsep SMP.