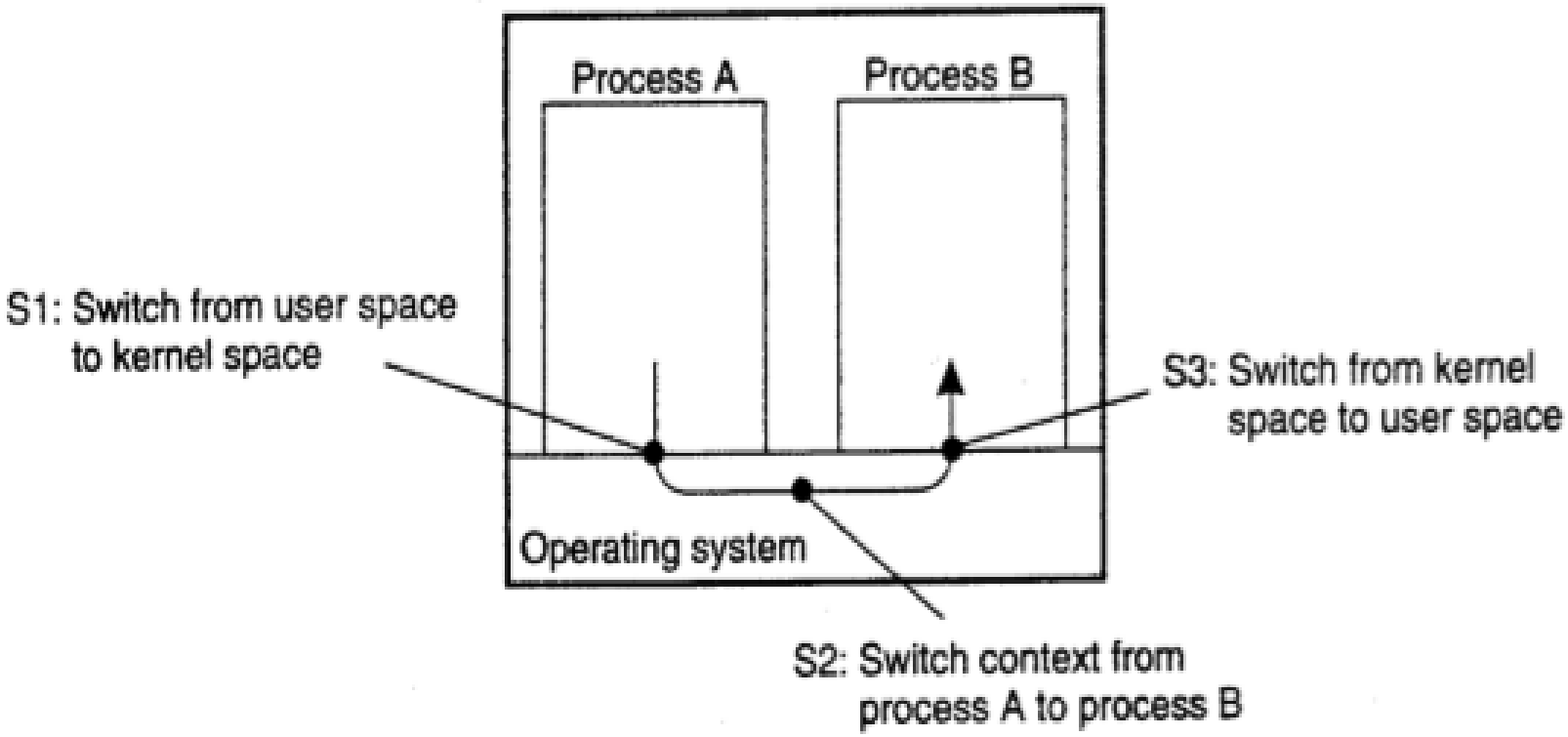
Pertemuan 3
Proses Sistem Terdistribusi

Pokok Bahasan

- Proses sitem terdistribusi
- Pengantar threads
- Peranan threads
- Implementasi threads
- Threads dalam sistem terdistribusi
- Klien multi threads
- Virtualization
- Client
- Server

Proses Sistem Terdistribusi

Dalam pertemuan ini, akan dibahas bagaimana berbagai jenis proses mempengaruhi peran penting sistem terdistribusi. Konsep proses berasal dari bidang sistem operasi di mana secara umum didefinisikan sebagai program dalam pelaksanaan. Dari perspektif sistem operasi, manajemen dan penjadwalan proses mungkin merupakan masalah yang paling penting untuk dihadapi. Namun, ketika datang ke sistem terdistribusi, masalah lain muncul menjadi sama atau lebih penting. Misalnya, untuk mengatur sistem klien-server secara efisien, sering kali nyaman menggunakan teknik multithreading. Thread adalah bahwa memungkinkan klien dan server untuk dibangun sedemikian rupa sehingga komunikasi dan pemrosesan lokal dapat tumpang tindih, menghasilkan kinerja yang tinggi.



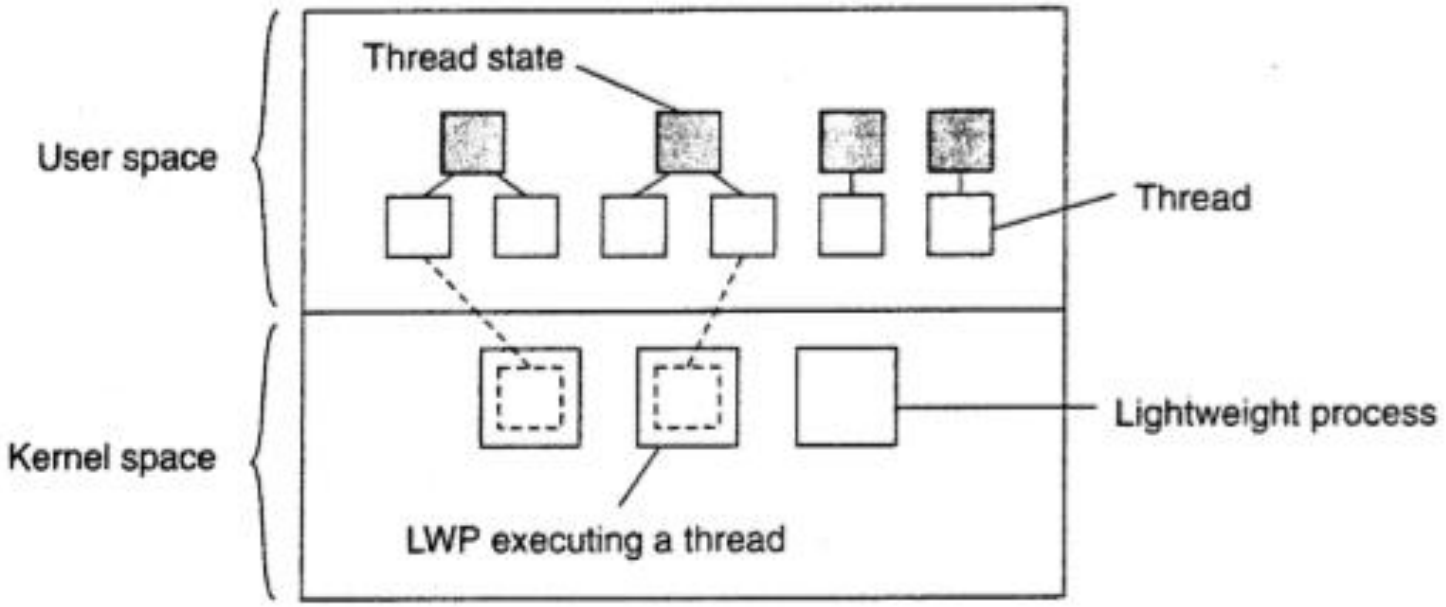
Gambar 1. Pergantian konteks sebagai hasil dari IPC

Pengantar Threads

Meskipun proses membentuk blok bangunan dalam sistem terdistribusi, praktik menunjukkan bahwa rincian proses yang ditentukan oleh sistem operasi tempat sistem terdistribusi dibangun tidak cukup. Sebaliknya, ternyata memiliki kesamaan granularity dalam bentuk beberapa thread kontrol per proses membuatnya lebih mudah untuk membangun aplikasi terdistribusi dan untuk mencapai kinerja yang lebih baik. Pada bagian ini, dapat dilihat lebih dekat pada peran utas dalam sistem terdistribusi dan menjelaskan mengapa mereka sangat penting. Lebih lanjut tentang utas dan bagaimana mereka dapat digunakan untuk membangun aplikasi

Peranan Threads

Untuk memahami peran thread dalam sistem terdistribusi, penting untuk memahami apa proses itu, dan bagaimana proses dan thread terkait. Untuk menjalankan program, sistem operasi menciptakan sejumlah prosesor virtual, masing-masing untuk menjalankan program yang berbeda. Untuk melacak prosesor virtual ini, sistem operasi memiliki tabel proses, yang berisi entri untuk menyimpan nilai register CPU, peta memori, file terbuka, informasi akuntansi. hak istimewa, dll. Suatu proses sering didefinisikan sebagai program dalam eksekusi, yaitu, program yang saat ini sedang dijalankan pada salah satu prosesor virtual sistem operasi.



Gambar 2. Menggabungkan proses ringan tingkat kernel dan utas tingkat pengguna.

Paket thread dapat dibagikan oleh banyak LWP, seperti yang ditunjukkan pada Gambar 2. Ini berarti bahwa setiap LWP dapat menjalankan utasnya sendiri (level pengguna). Aplikasi multithreaded dibangun dengan membuat utas, dan selanjutnya menetapkan setiap utas ke LWP. Menetapkan utas ke LWP biasanya tersirat dan tersembunyi dari programmer.

Implementasi Thread

Thread sering diberikan dalam bentuk paket thread. Paket seperti itu berisi operasi untuk membuat dan menghancurkan thread serta operasi pada variabel sinkronisasi seperti mutex dan variabel kondisi. Pada dasarnya ada dua pendekatan untuk mengimplementasikan paket thread. Pendekatan pertama adalah membangun pustaka thread yang dijalankan sepenuhnya dalam mode pengguna. Pendekatan kedua adalah membuat kernel sadar akan benang dan menjadwalkannya. Pustaka thread tingkat pengguna memiliki sejumlah keunggulan. Pertama, murah untuk membuat dan menghancurkan thread. Karena semua administrasi thread disimpan di ruang alamat pengguna, harga pembuatan thread terutama ditentukan oleh biaya untuk mengalokasikan memori untuk menyiapkan setumpuk thread.

Thread dalam Sistem Terdistribusi

Properti penting dari thread adalah mereka dapat memberikan cara yang nyaman untuk memungkinkan pemblokiran panggilan sistem tanpa memblokir proses pusat di mana thread berjalan. Properti ini membuat thread sangat menarik untuk digunakan dalam sistem terdistribusi karena membuatnya jauh lebih mudah untuk mengekspresikan komunikasi dalam bentuk mempertahankan beberapa koneksi logis pada saat yang sama. Dapat dilustrasikan hal ini dengan mengambil pandangan klien di multithreaded dan server masing-masing

Klien Multithreaded

Untuk membangun tingkat transparansi distribusi yang tinggi, sistem terdistribusi yang beroperasi di jaringan area luas mungkin perlu menyembunyikan waktu pengiriman pesan antarproses yang lama. Penundaan pulang-pergi dalam jaringan area luas dapat dengan mudah berada dalam urutan ratusan milidetik. atau bahkan kadang-kadang detik. Cara biasa untuk menyembunyikan latensi komunikasi adalah dengan memulai komunikasi dan segera melanjutkan dengan hal lain. Contoh khas di mana ini terjadi adalah di browser Web. Dalam banyak kasus, dokumen Web terdiri dari file HTML yang berisi teks biasa bersama dengan koleksi gambar, ikon, dll. Untuk mengambil setiap elemen dokumen Web, browser harus mengatur koneksi TCPIIP, membaca data yang masuk, dan meneruskannya ke komponen tampilan.

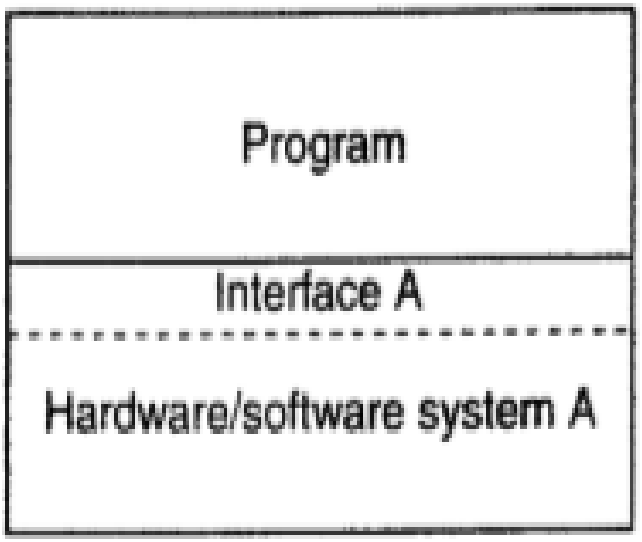
VIRTUALIZATION

Thread dan proses dapat dilihat sebagai cara untuk melakukan lebih banyak hal pada saat yang sama. Akibatnya, mereka memungkinkan membangun (potongan-potongan) program yang tampaknya dijalankan secara bersamaan. Pada komputer prosesor tunggal, eksekusi simultan ini, tentu saja, ilusi. Karena hanya ada satu CPU, hanya instruksi dari satu utas atau proses akan dieksekusi pada suatu waktu.

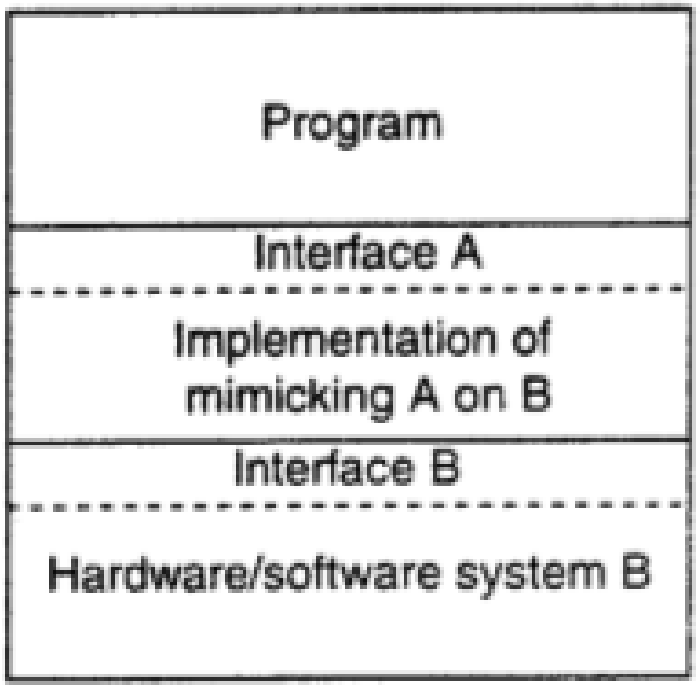
Virtualisasi ini telah diterapkan selama beberapa dekade, tetapi telah menerima minat baru karena sistem komputer (terdistribusi) menjadi lebih umum dan kompleks, yang mengarah pada situasi bahwa perangkat lunak aplikasi sebagian besar selalu hidup lebih lama dari perangkat lunak dan perangkat keras sistem yang mendasarinya.

Peran Virtualisasi dalam Sistem Terdistribusi

Dalam praktiknya, setiap sistem komputer (terdistribusi) menawarkan antarmuka pemrograman ke perangkat lunak tingkat yang lebih tinggi, seperti yang ditunjukkan pada Gambar 2. Ada banyak jenis antarmuka, mulai dari set instruksi dasar yang ditawarkan oleh CPU hingga koleksi besar antarmuka pemrograman aplikasi yang dikirimkan dengan banyak sistem middleware saat ini. Karena itu, virtualisasi berurusan dengan menambah atau mengganti antarmuka yang ada untuk meniru perilaku sistem lain, seperti yang ditunjukkan pada Gambar.3. Dalam hal ini akan dibahas rincian teknis tentang virtualisasi segera, tetapi berkonsentrasi pada mengapa virtualisasi penting untuk sistem terdistribusi.



Gambar 2



Gambar 3

Gambar 2. Organisasi umum antara program, antarmuka, dan sistem.

Gambar 3. Organisasi umum sistem virtualisasi A di atas sistem B.

Pendekatan alternatif menuju virtualisasi adalah menyediakan sistem yang pada dasarnya diimplementasikan sebagai lapisan yang sepenuhnya melindungi perangkat keras asli, tetapi menawarkan rangkaian instruksi lengkap yang sama (atau perangkat keras lain) sebagai antarmuka. Yang penting adalah kenyataan bahwa antarmuka ini dapat ditawarkan secara bersamaan ke berbagai program. sistem operasi berjalan secara independen dan bersamaan pada platform yang sama. Lapisan ini umumnya disebut sebagai monitor mesin virtual (VMM). Contoh khas dari pendekatan ini adalah VMware

CLIENTS

Sebelumnya telah dubahas mode klien-server peran klien dan server, dan cara mode tersebut berinteraksi. Lihat lebih dekat pada anatomi klien dan server. Dengan mulai di bagian ini mengenai klien. Server dibahas di bagian selanjutnya.

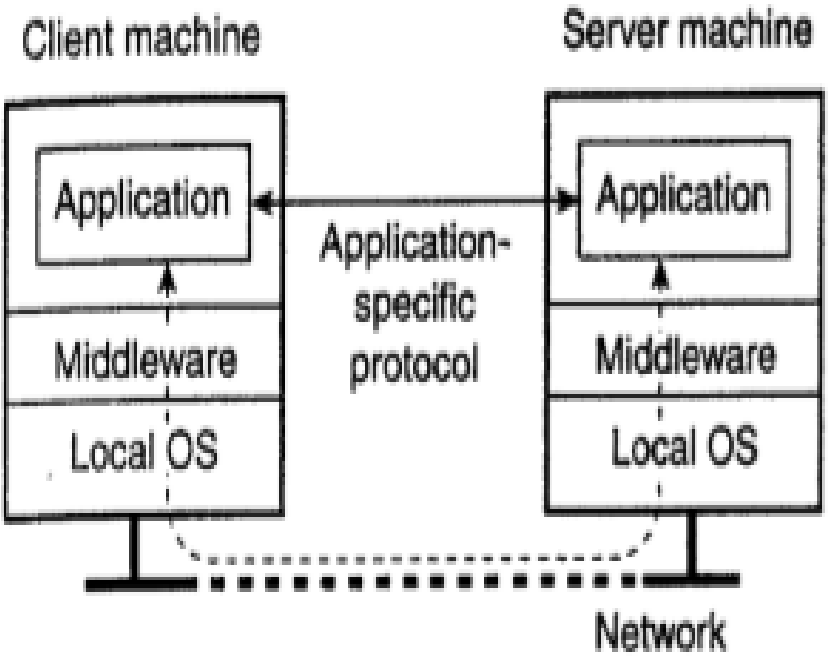
1. Networked User Interfaces

Tugas utama mesin klien adalah menyediakan sarana bagi pengguna untuk berinteraksi dengan server jarak jauh. Terdapat dua cara di mana interaksi ini dapat didukung. Pertama, untuk setiap layanan jarak jauh, mesin klien akan memiliki rekanan terpisah yang dapat menghubungi layanan melalui jaringan

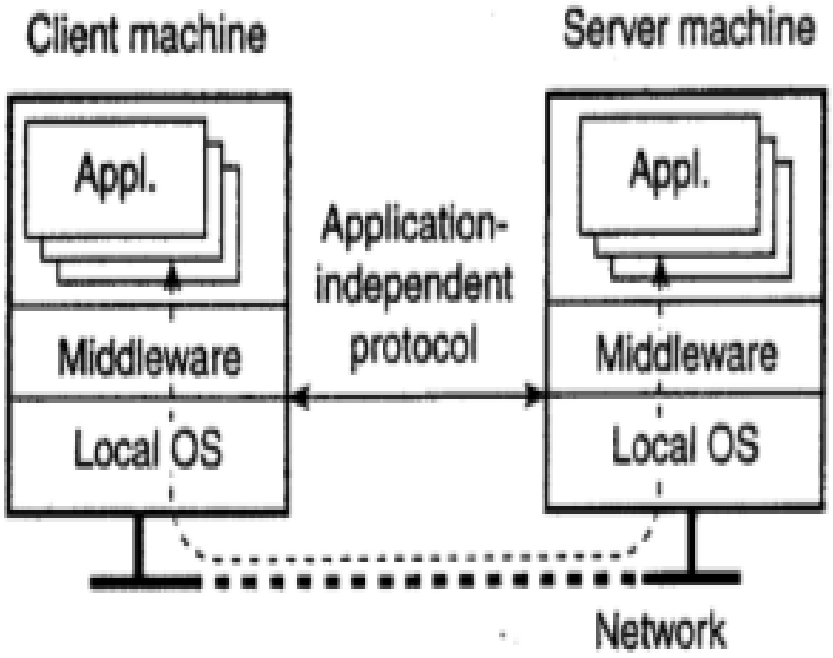
Contoh tipikal adalah agenda yang berjalan pada PDA pengguna yang perlu disinkronkan dengan agenda yang jauh dan mungkin dibagikan. Dalam hal ini, protokol tingkat aplikasi akan menangani sinkronisasi.

Solusi Pertama adalah menyediakan akses langsung ke layanan jarak jauh dengan hanya menawarkan antarmuka pengguna yang nyaman. Secara efektif, ini berarti bahwa mesin klien hanya digunakan sebagai terminal tanpa perlu penyimpanan lokal, yang mengarah ke solusi netral aplikasi seperti yang ditunjukkan pada Gambar 4a.

Dalam hal antarmuka pengguna jaringan, semuanya diproses dan disimpan di server. Pendekatan thin-client ini menerima lebih banyak perhatian saat konektivitas Internet meningkat, dan perangkat genggam menjadi lebih canggih. Seperti yang sudah dibahas sebelumnya, solusi thin-client juga populer karena meringankan tugas manajemen sistem.



Gambar 4a



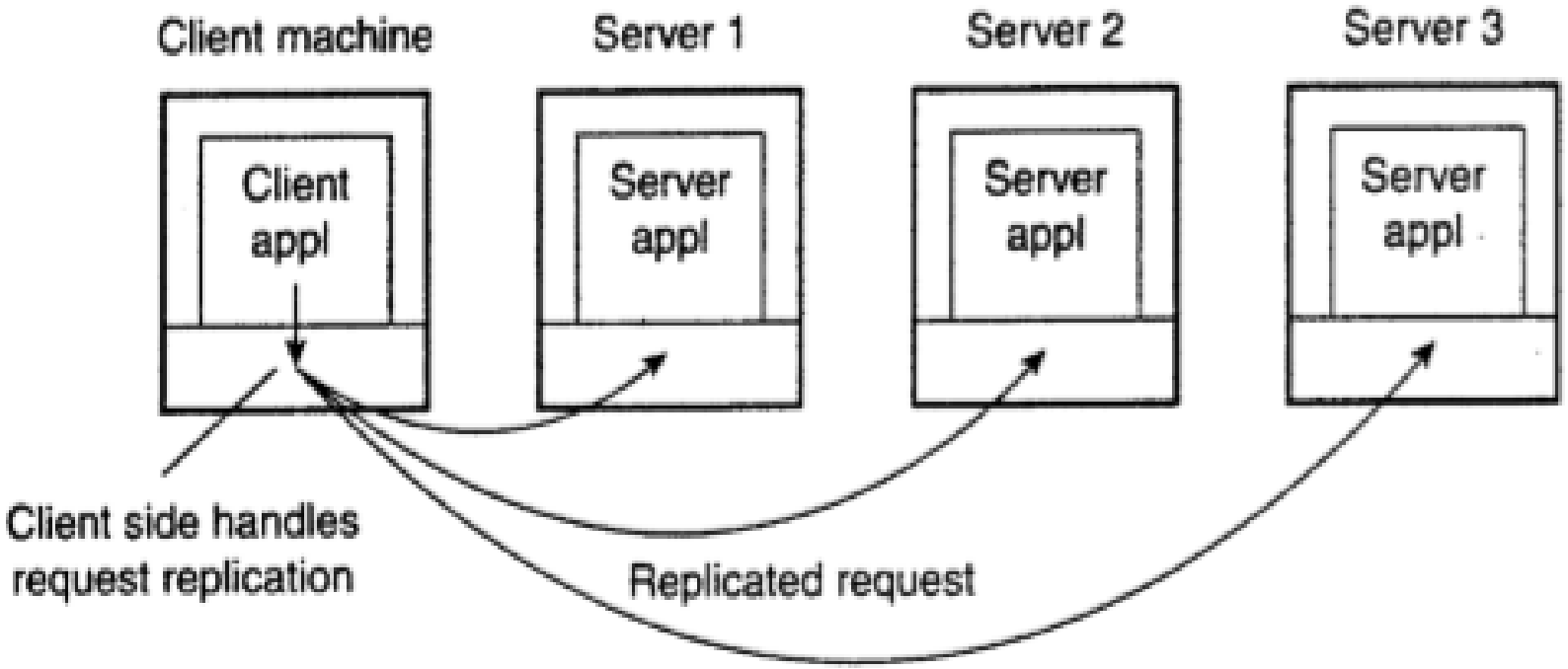
Gambar 4b

Solusi kedua adalah menyediakan akses langsung ke layanan jarak jauh dengan hanya menawarkan antarmuka pengguna yang nyaman. Secara efektif, ini berarti bahwa mesin klien hanya digunakan sebagai terminal tanpa perlu penyimpanan lokal, yang mengarah ke solusi netral aplikasi seperti yang ditunjukkan pada Gambar. 4b. Dalam hal antarmuka pengguna jaringan, semuanya diproses dan disimpan di server. Pendekatan thin-client ini menerima lebih banyak perhatian saat konektivitas Internet meningkat, dan perangkat genggam menjadi lebih canggih. Seperti yang sudah dibahas sebelumnya, solusi thin-client juga populer karena meringankan tugas manajemen sistem.

2. Client-Side Software for Distribution Transparency

Perangkat lunak klien terdiri dari lebih banyak antarmuka pengguna yang sempurna. Dalam banyak kasus, bagian dari tingkat pemrosesan dan data dalam aplikasi client-server dieksekusi di sisi klien juga. Kelas khusus dibentuk oleh perangkat lunak klien tertanam, seperti untuk mesin teller otomatis (ATM), mesin kasir, pembaca barcode, kotak TV, dll.
Idealnya, klien tidak perlu menyadari bahwa ia berkomunikasi dengan proses jarak jauh. Sebaliknya, distribusi seringkali kurang transparan ke server karena alasan kinerja dan kebenaran.

Dengan cara yang sama, banyak sistem terdistribusi menerapkan transparansi replikasi melalui solusi sisi klien. Sebagai contoh, bayangkan sebuah sistem terdistribusi dengan server replikasi, replikasi tersebut dapat dicapai dengan meneruskan permintaan ke setiap replika, seperti yang ditunjukkan pada Gambar 5. Perangkat lunak sisi klien dapat secara transparan mengumpulkan semua respons dan meneruskan tanggapan tunggal ke aplikasi klien.



Gambar 5. Replikasi transparan dari server menggunakan solusi sisi klien.

SERVERS

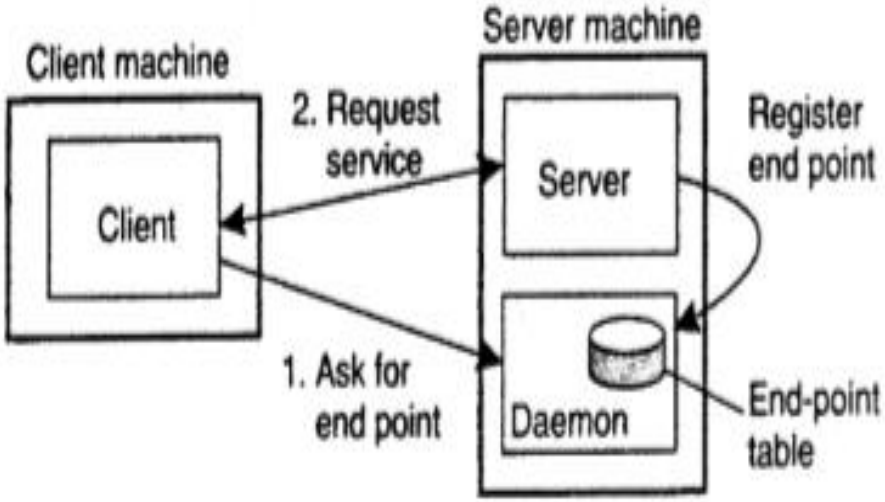
Dapat dilihat organisasi server lebih dekat. Pada halaman-halaman berikut, pertama-tama berkonsentrasi pada sejumlah masalah desain umum untuk server, yang akan diikuti oleh diskusi tentang cluster server.

1. Masalah Desain Umum

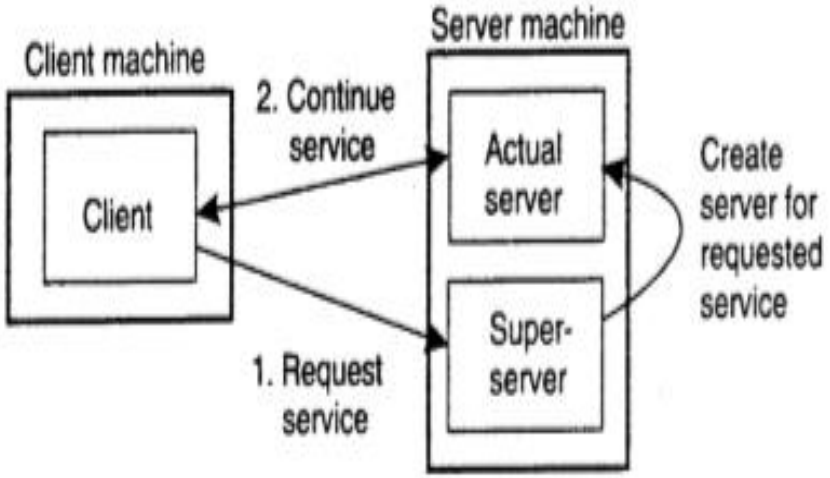
Server adalah sebuah proses yang mengimplementasikan layanan tertentu setelah kumpulan klien. Intinya, setiap server diatur dengan cara yang sama: ia menunggu permintaan masuk dari klien dan selanjutnya memastikan bahwa permintaan itu ditangani, setelah itu ia menunggu permintaan masuk berikutnya

Ada banyak layanan yang tidak memerlukan titik akhir yang ditentukan sebelumnya. Dalam hal ini, klien pertama-tama harus mencari titik akhir. Salah satu solusinya adalah memiliki daemon khusus yang berjalan di setiap mesin yang menjalankan server. Daemon melacak titik akhir saat ini dari setiap layanan yang dilaksanakan oleh server bersama. Daemon itu sendiri mendengarkan titik akhir yang terkenal. Klien pertama-tama akan menghubungi daemon, meminta titik akhir, dan kemudian menghubungi server tertentu, seperti yang ditunjukkan pada Gambar. 6a. Adalah umum untuk menghubungkan titik akhir dengan layanan tertentu. Namun, sebenarnya menerapkan setiap layanan melalui server yang terpisah mungkin merupakan pemborosan sumber daya.

Daripada harus melacak begitu banyak proses pasif, lebih efisien untuk memiliki satu saja. superserver mendengarkan setiap titik akhir yang terkait dengan layanan tertentu, seperti yang ditunjukkan pada Gambar 6b.



Gambar 6a



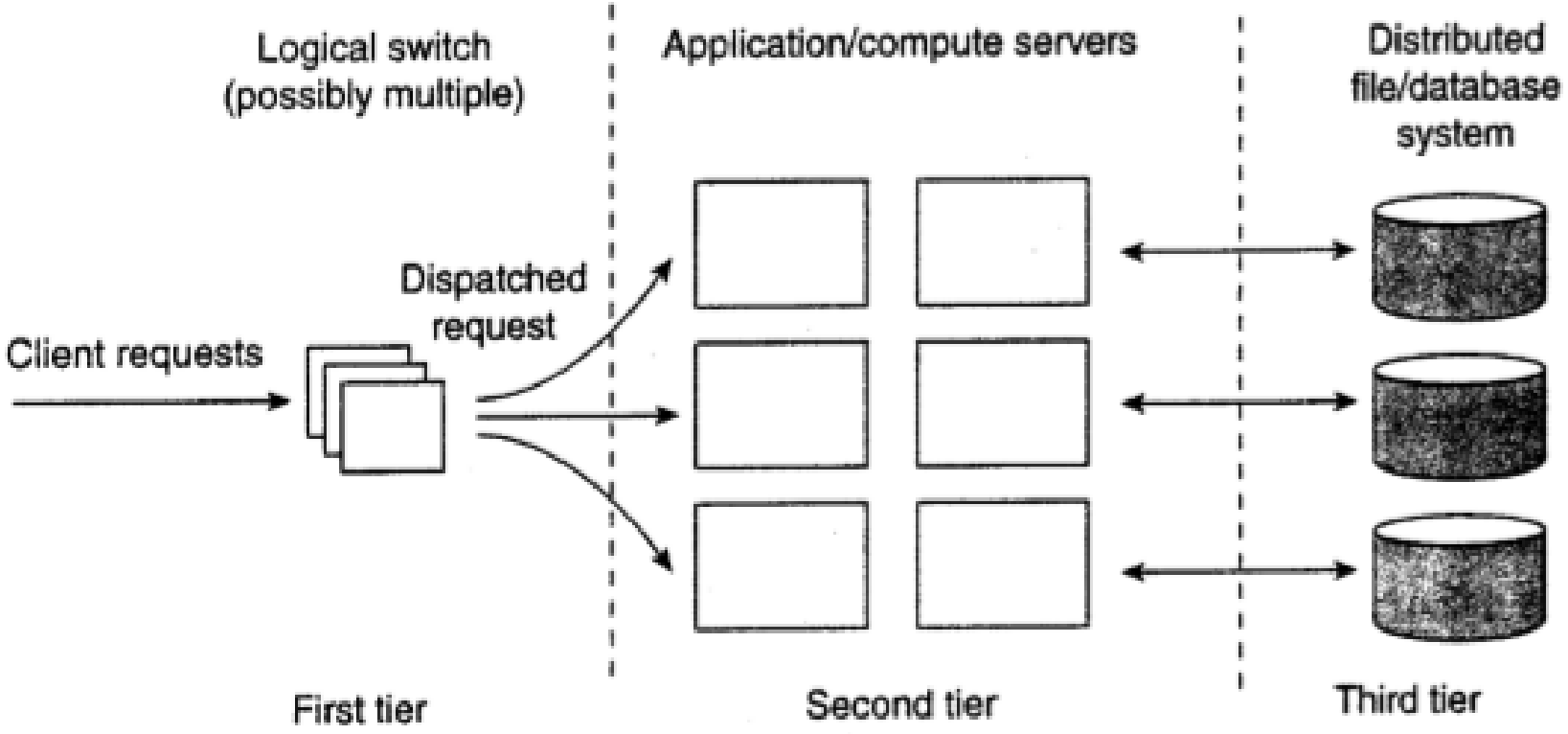
Gambar 6b

Gambar 6a. Client-to-server binding using a daemon

Gambar 6b. Client-to-server binding using a superserver.

2. Server Clusters

Sederhananya, sebuah cluster server tidak lain adalah kumpulan mesin yang terhubung melalui jaringan, di mana setiap mesin menjalankan satu atau lebih server. Cluster server yang dipertimbangkan di sini, adalah yang di mana mesin terhubung melalui jaringan area lokal, sering menawarkan bandwidth tinggi dan latensi rendah.



Gambar 7. Organisasi umum cluster server tiga tingkat.

Seperti dalam arsitektur klien-server multitier, banyak server cluster juga server yang didedikasikan untuk pemrosesan aplikasi. Dalam komputasi cluster, ini biasanya server yang berjalan pada perangkat keras berkinerja tinggi yang didedikasikan untuk memberikan daya komputasi. Namun, dalam kasus cluster server perusahaan, mungkin itu kasus bahwa aplikasi hanya perlu berjalan pada mesin yang relatif rendah, karena daya komputasi yang diperlukan bukan hambatan, tetapi akses ke penyimpanan adalah. Ini membawa ke tingkat ketiga, yang terdiri dari server pemrosesan data, terutama server file dan database. Sekali lagi, tergantung pada penggunaan cluster server, server ini mungkin menjalankan mesin khusus, dikonfigurasi untuk akses disk berkecepatan tinggi dan memiliki cache data sisi server yang besar.