BAB X
ARSITEKTUR SISTEM TERDISTRIBUSI

A. Sistem Terdistribusi

Hampir semua sistem berbasis computer yang besar saat ini mrupakan sistem terdistribusi (sistem tersebar). Sitem terdistribusi adalah sistem dimana pemrosesan informasi didistribusikan pada beberapa computer dan tidak terbatas hanya pada satu mesin saja. Jelas rekayasa terdistribusi memiliki banyak kesamaan dengan rekayasa perangkat luna lainnya tetapi ada isu-isu khusus yang harus diperhitungkan ketika merancang tipe sistem ini. Perekayasa perangkat lunak harus menyadari dan memperhitungkan karena Sitem terdistribusi ini banyak digunakan. Belum lama ini kebanyakan sistem besar masih menggunakan sistem sentral yang berjalan pada satu mainframe dengan terminal-terminal yang terhubung kepadanya. Sistem tersebut bayak kelemahannya dimana terminal-terminal hanya sedikit kemampuan pemrosesannya dan semua tergantung pada computer sentral.

Sampai saai ini ada tipe sistem yang utama yaitu:

- *Sistem Personal* yang tidak terditribusi dan dirancang untuk satu workstation saja.
- Sistem Embedded yang bejalan pada satu prosessor atau pada kelompok prosessor yang terintegrasi.
- *Sistem Terdistribusi* dimana perangkat lunak sistem berjalan pada kelompok prosessor yang bekerja sama dan terintegrasi secara longgar, dengan dihubungkan oleh jaringan. Contohnya sistem ATM bank, sistem groupware, dll

Menurut Coulouris et.al (1994) mengidentifikasi enam karakteristik yang penting untuk sistem terdistribusi yaitu:

- Pemakain bersama sumber daya
- *Keterbukaan*. Keterbukaan sistem adalah terbuka untuk banyak sistem operasi dan banyak vendor.
- *Konkurensi*. Sitem terdistribusi memungkinkan beberapa proses dapat beroperasi pada saat yang sama pada berbagai computer di jaringan. Proses ini dapat (tapi tidak perlu) berkomunikasi satu dengan lainnya pada saat operasi normlnya.

- *Skalabilitas.* Sitem terdistribusi dapat diskala dengan menguprade atau menambahkan sumber daya baru untuk memenuhi kebutuhan sistem.
- *Toleransi kkesalahan*. Sitem terdistribusi bersifat toleran terhadap beberapa kegagalan perangkat keras dan lunak dan layanan terdegradasi dapat diberikan ketika terjadi kegagalan.
- *Transparansi*. Sitem terdistribusi adalah bersifat terbuka bagi user.

Selain hal-hal tersebut ada juga kelemaham dari Sistem terdistribusi yaitu:

- Kompleksitas. Sistem terdistribusi bersigat lebih kompleks daripada sistem sentral.
- Keamanan. Sistem terdistribusi dapat diakses dari beberapa computer dan jalur jaringan mudah disadap, sehingga keamanan jarinagan sistem terdistribusi menjadi masalah yang besar.
- Kemampuan untuk dikendalikan. Komputer yang terdapat di sistem terdistribusi bis aterdiri dari berbagai tipe yang berbeda dan mungkin dijalankan pada sistem operasi yang berbeda pula. Kesalahan pada satu mesin bisa berakibat pada yang lainnya. Sehingga harus banyak usaha untuk mengendalikannya.
- Tidak dapat diramalkan. Sistem terdistribusi tidak dpat diramalkan tanggapannya. Tanggapan tergantung beban total sistem, pengorganisasian, dan beban jaringannya.

Ada dua tipe generic arsitektur sistem terdistribusi yaitu:

- **Arsitektur Client Server**.

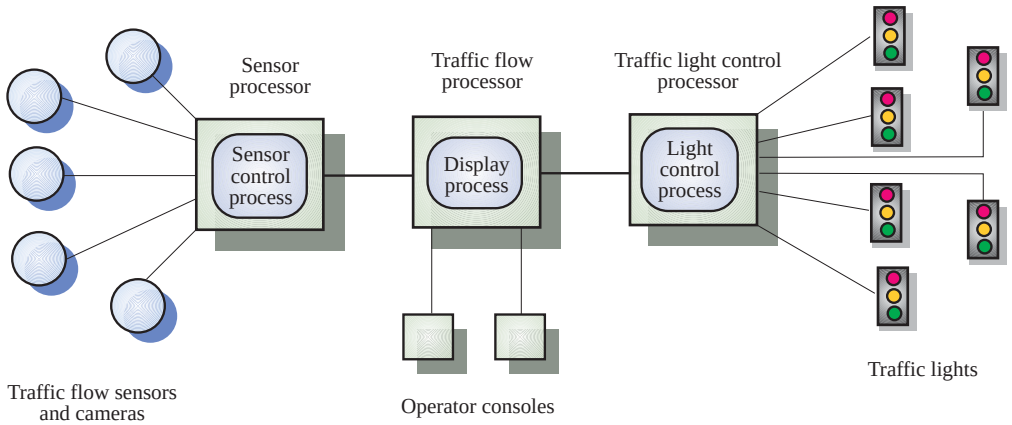
  Sistem dianggap sebagai satu set layanan yang disediakan untuk klien. Server dan Client diperlakukan berbeda .
- **Arsitektur Objek Terdistribusi**.

  Tidak ada perbedaan antara server dan client, sistem dapat sebagai satu set objek yang berinteraksi yang likasinya tidak relevan. Tidak ada perbedaan antara penyedia layanan dan user layanan.

B. Arsitektur Multiprosessor

Model sistem terdistribusi yang paling sederhana adalah sistem multiprosessor dimana sistem terdiri dari sejumlah proses yang dapat berjalan pada beberapa prosessor yang terpisah. Model ini umumnya digunakan pada sistem real time yang besar.

Penggunaan banyak prosessor ini berguna untuk memperbaiki kinerja dan fleksibilitas sistem. Distribusi proses ke prosessor dapat ditentukan sebelumnya atau bisa juga dikendalikan oleh *dispatcher* yang memutuskan proses mana yang akn dialokasikan ke masing-masing prosessor. Contoh tipe sistem ini dapat dilihat pada gambar 10.1 yang merupakan contoh model sistem control lau lintas. Satu set sensor terdistribusi mengumpulkan informasi dari aliran lalu lintas dan memproses informasi secara local sebelum mengirimnya ke ruangan control. oPerator mengambil keputusan dengan memakai informasi ini dan memberi instruksi ke proses control lampu liantas yang berbeda. Ada proses logika yang terpisah untuk menangani sensor ruangan control, dan lampu lalu lintas. Dan proses-proses ini berjalan pada prosesor yang terpisah.

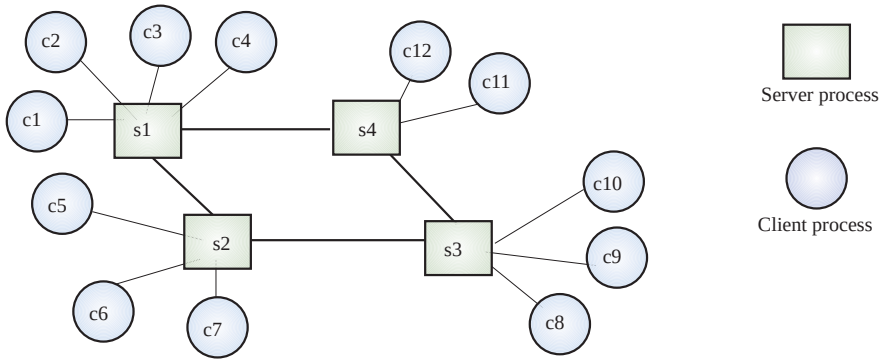


Gambar 10.1 Model Kontrol Lalu lintas

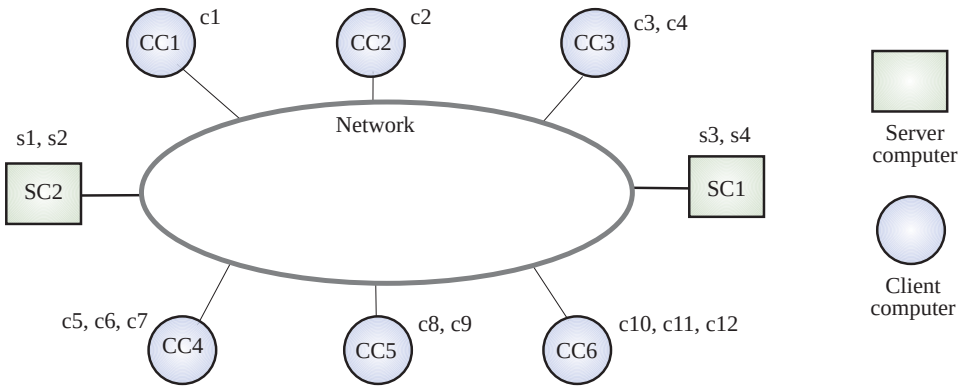
Sistem perangkat lunak yang terdiri dari banyka proses tidak harus merupakan sistem terdistribusi. Memang jika tersedia lebih dari satu prosessor maka distribusi dapat diimplementasikan.

C. Arsitektur Client Server

Seperti yang telah dijelaskan pada bab-bab sebelumnya model Arsitektur Cilent Server dimodelkan sebagai satu set layanan yang disediakan oleh server dan satu atau lebih client yang memakai layanan server. Client tidak perlu menyadari keberadaan server tetapi juga sebaliknya tidak mengetahui keberadaan klien yang lain. Tidak harus ada pemetaan 1:1 antara proses dan prosesor pada sistem. Hal tersebut bisa dilihat pada contoh gambar 10.2 dan dimana ada arsitektur fisik dengan enam kompuetr client dan dua server. Dan untuk proses logika dari gambar 10.2 adalah ditunjukkan pada gambar 10.3



GAmbar 10.2 Sistem Client Server



Gambar 10.3 Proses Logika Arsitektur C/S

Perancangan arsitektur C/S harus mempertimbangkan struktur logika aplikasi yang ditunjukkan pada gambar 10.4 yang menunjukkan aplikasi dibagi tiga lapisan yaitu:

- Lapisan Presentasi, yang berhubungan dengan penyajian informasi ke user dan dengan semua interaksi user.
- Lapisan Pemrosesan Aplikasi, yang berhubungan dengan implementasi logika aplikasi
- Lapisan Manajemen Data, yang berhubungan dengan operasi database.

Arsitektur C/S yang paling sederhana disebut arsitektur *C/S Two Tier*, diaman aplikasi diorganisir seperti server dan satu set klien seperti pada gambar 10.5 dimana arsitektur *C/S Two Tier* memiliki dua bentuk yaitu:

- ***Model Thin Client.***

  Pada model ini semua pemrosesan aplikasi dan manajaemen data dilakukan pada server. Klien bertanggung jawab untuk menjalankan perangkat lunak presentasi yang biasanya hanya berbentuk interface sistem atau GUI.

  Kelebihan:
  - Biaya lebih rendah
  - Lebih cocok untuk model jaringan yang sederhana

  Kekurangan:
  - Menempatkan beban berat pemrosesan pada server
  - Ada kekuatan pemrosesan yang besar yang tersedia pada PC modern dan tidak digunakan pada client

- **Model Fat Client**

  Pada model ini server hanya bertanggung jawab pada manajemen data. Perangkat client bertanggung jawab pada logika aplikasi dan interaksi denga user.

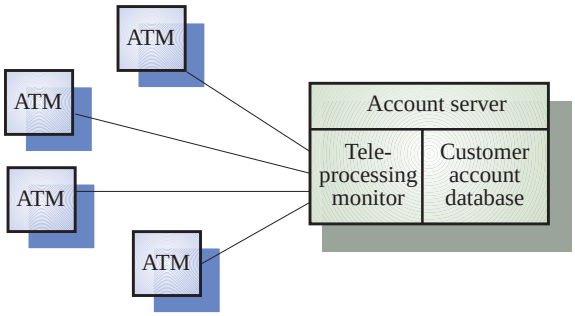
  Kelebihan:
  - Menggunakan kekuatan pemrosesan yang besar dan mendistribusikan pemrosesan logika aplikasi dan prsentasi pada klien
  - Server hanya menangani seluruh transaksi database

- Pendistribusian pemrosesan lebih efektif

Kekurangan :

- Manajemen sistemnya lebih komplek
- Biayanya lebih besar

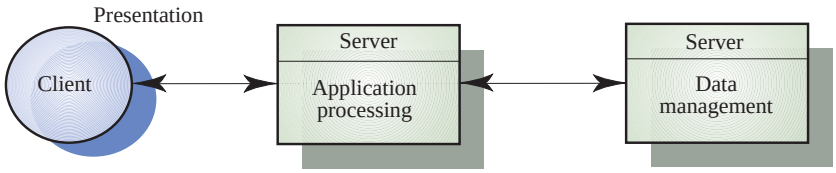
Contoh dari model arsitektur *C/S Two Tier* dapat dilihat pada gambar 10.4 tentang jaringan ATM. Pada gambar tersebut ATM tidak berhubungan langsung dengan database nasabah, tetapi terhubung ke *monitor teleprocessing*. *Monitor teleprocessing* (TP) merupalan *middleware* yang mengatur komunikasi dengan client jarak jauh (*remote)* dan menserikan transaksi client untuk diproses oleh database. Menggunakan transaksi *serial* berarti bahwa sistem dapat pulih dari kesalahan tanpa merusak data sistem.



Munculnya Java dan applet yang dapat di download secara gratis memungkinkan pengembangan sistem C/S antar model thin dan fat client. Beberapa pernagkat lunak pemrosesan aplikasi dapat didownload ke client seperti Java Applet sehingga mengurangi beban pada server. Interface User dibangun dengan web browser yang dapat menjalankan Java applet.

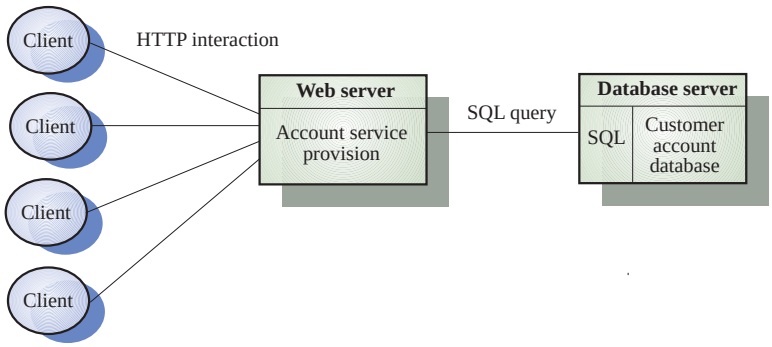
Masalah terpenting pada arsitektur *two tier C/S* adalah ketiga lapisan logika harus dipetakan ke dua sistem computer. Mungkin ada masalah skalabilitas dan kinerja jika dipilih model *thin client.* Mungkin pila ada masalah manajemen sistem jika dipilih *fat client.* Untuk mengatasi masalah ini pendekatan alternative menggunakan arsitektur *three tier client server* (ditunjukkan pada gambar 10.5). Pada arsitektur ini tidak mesti diartikan bahwa terdapat tiga computer ynag terhubung ke jaringan . Satu computer server dapat menjalankan pemrosesan aplikasi dan manajemen data aplikasi sebagai server logika yang terpisah. Tetapi jika permintaan bertambah, pemisahan pemrosesan aplikasi dan

manajamen data dapat dilakukan dengan langsung dan eksekusi dilakukan pada prosesor yang terpisah.



Gambar 10.5 Arsitektur C/S three tier

Sistem Internet banking merupakan salah satu contoh sistem arsitektur *three tier C/S.* Database nasabah bank menyediakan layanan manajemen data, web server menyediakan layanan aplikasi seperti fasilitas transfer uang tunai, membayar tagihan, dll. Komputer nasabah dengan browser internet merupakan client. Sistem ini mudah diskala untuk menambahkan web server baru dengan bertambahnya jumlah nasabah.



Gambar 10.6 Arsitektur terdistribusi sistem *internet banking*

Kelebihan sistem Three Tier C/S adalah diantaranya:

- Transfer informasi antara web server dan server database optimal
- Komunikasi aantara sistem-sistemtidak harus didasarkan pada standart internet, tetapi dapat menggunakan protocol komunikasi yang lebih cepat dan berada pada tingkat yang lebih rendah.
- Penggunaan middleware mendukung efesien query database dalam SQL dipakai untuk menangani pengambailan informasi dari database.

D. Middleware

Seperti yang sudah dijelaskan di bagian sebelumnya arsitektur sistem yang terdistribusi membutuhkan middleware (object request broker) untuk menangani komunikasi antar objek-objek. Pada prinsipnya, objek-objek pada sistem dapat diimplementasikan dengan bahasa pemrograman yang berbeda, dapat berjalan pada platform yang berbeda dan namanya tidak perlu diketahui semua objek lain pada sistem.

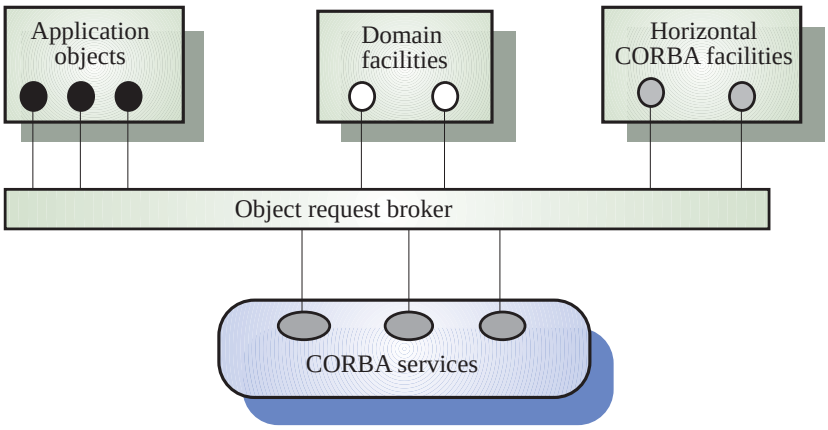
Pada saat ini ada dua standar utama middleware untuk mendukung komputasi objek terdistribusi yaitu:

- **CORBA (***Command Object Request Broker Architecture*)
  CORBA merupakan satu set standar middleware yang dikeluarkan oleh OMG (*Object Management Group*). Standar CORBA mendefinisikan pendekatan yang tidak dependen mesin dan generic terhadap komputasi objek terdistribusi. Sejumlah implementasi ini tersedia untuk aplikasi sistem operasi UNIX dan Microsoft.
- **DCOM** (*Distributed Component Object Mode*)
  DCOM dikembangkan oleh Microsoft . Model komputasi tersditribusinya kurang umum dari model CORBA dan DCOM memberikan dukungan yang terbatas pada interoperabilitas.
- **RMI** (Remote Method Invocation)
  Dikembangkan oleh Java.

CORBA dikembangkan oleh OMG yang berperan dalam mendefinisikan standar untuk pengembangan berorientasi objek tetapi tidak menyediakan implementasi untuk standar ini. Visi OMG tentang sistem terdistribusi ditunjukkan pada gambar 10.7 yang mengusulkan agar aplikasi terdistribusi terbuat dari sejumlah komponen yaitu:

- *Objek APlikasi*, yang dirancang dan diimlemantasi untuk aplikasi ini.
- *Objek Standar*, Yang didifinisikan oleh OMG untuk domain khusus misalnya untuk masalah keuangan, e-commerce, kesehatan dll.
- *LAyanan CORBA*, fundamental yang menyediakan layanan komputasi terdistribusi dasar seperti direktori, manajemen sekuritas, dll.

- *Fasilitas CORBA horizontal*, seperti fasilitas interface user, manajemen sistem, dll. Istilah horizontal berarti fasilitas bersifat umum untuk banyak domain aplikasi.



GAmbar 10.7 CORBA application structure

Standar CORBA mencakup semua aspek visi ini . ada empat elemen utama untuk standar ini yaitu:

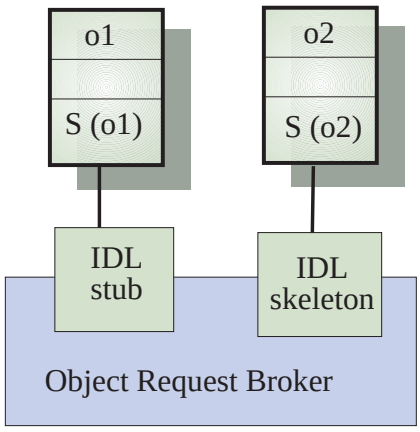
1. Model objek untuk aplikasi ini objek CORBA merupakan encapsulasi status dengan interface yang terdefinisi dan disdeskripsikan dalam IDL (*interface Definition Language*).
2. *Object Request Broker (ORB),* yang menangani permintaan layanan objek. ORB mencari objek yang menyediakan layanan tersebut, menyediakannya untuk permintaan, mengirimkan dan mengembalikan hasilnya kepada pemohon.
3. *Satu set layanan objek,* yang menyediakan layanan umum dan mungkin diperlukan oleh banyak aplikasi terdistribusi. Contoh layanan ini adalh layanan direktori, layanan transaksi,
4. *Satu set komponen umum,* yang dibangun di atas layanan=layanan dasar yang mungin dibutuhkan oleh aplikasi.

Objek CORBA memiliki identifier unik yang dinamakan Interoperable Object Reference (IOR) yang digunakan ketika satu objek meminta layanan dari yang lain.

Object Request Broker (ORB) tahu mengenai objek yang meminta layanan dan interface mereka. ORB menangani komunikasi diantara objek-objek tersebut. Objek-objek

yang berkomunikasi tidak perlu mengetahui lokasi objek lain dan implementasinya. Karena interface mengisolasi objek dari ORB, maka perubahan implementasi objek dilakukan dengan cara yang transparan. Lokasi objek dapat berubah antara invokasi dn hal ini transparan bagi objek lainnya pada sistem tersebut.

Hal ini dapat dilihat pada gambar 10.8 yang mengilustrasikan bagaimana dua objek *o1* dan *o2* berkomunikasi melakui ORB. Objek pemanggil (O1) memiliki IDL stub yang berhubungan dengan interface objek yang memberikan layanan yang dibutuhkan. Implementator o1 menggabungkan panggilan untuk stub ini pada implementasi objek ketika layanan dibutuhkan. IDL merupakan superset C++ segingga mudah untu mengaksesnya. Pemetaan ke IDL juga telah didefinikan untuk bahasa laian seperti ADA dan COBOL yang mendukung link ke IDL. Objek yang menyediakan layanan memiliki *IDL skeleton* yang berhubungan ke interface untuk aplikasi layanan. JAdi ketika layanan dipanggil melalui interface, *IDL skeleton* menerjemahkan panggilan ini menjadi panggilan layanan dalam bahasa implemenatsi apapun yang digunakan. Ketika metode atau prosedur telah dilaksanakan, *IDL skeleton* menerjemahkan hasilnya menjadi IDL sehingga dapat diakses oleh objek yang memanggil. Ketika objek menyediakan layanan ke objek lain dan menggunkan layanan yang disediakan ditempat lain, objek tersebut membutuhkan IDL skeleton dan IDL Stub. *IDL stub* dibutuhkan oleh setiap objek yang dipakai.



Gambar 10.8 Komunikasi Objek melalui ORB