MEMBANGUN SISTEM KOMPUTASI TERDISTRIBUSI DENGAN PEMROGRAMAN C++

Oleh: Maman Somantri
Jurusan Teknik Elektro – Fakultas Teknik – Universitas Diponegoro, Semarang
Jl. Prof. Sudharto, SH Tembalang, Semarang
Email: mmsomantri@yahoo.com

Abstrak

Pemrograman C++ untuk membangun aplikasi client server tunggal salah satunya bisa dikembangkan dengan memanfaatkan aplikasi socket. Tetapi lain halnya untuk membangun aplikasi komputasi terdistribusi yang memiliki server aplikasi yang terpisah/terdistribusi secara fisik maupun secara logik. Lebih berdaya lagi jika C++ bisa dimanfaatkan untuk aplikasi terdistribusi ini. Pada tulisan ini akan coba dirancang dan dikembangkan pemrograman C++ untuk sistem terdistribusi yang lebih kompleks. Agar C++ bisa diberdayakan untuk membangun aplikasi terdistribusi maka terlebih dahulu harus memilih sebuah framework yang mendukung sistem terdistribusi. Diantara beberapa framework yang bisa digunakan adalah diantaranya framework terdistribusi berbasis CORBA (Common Object Request Broker Architectur). Dalam Arsitektur CORBA, ada bahasa-bahasa pemrograman yang bisa digunakan untuk mengembangkan aplikasi terdistribusi adalah diantaranya: Java, C++, Delphi, dan Phyton. Untuk bahasa C++ programmer bisa memanfaatkan compiler yang memiliki dukungan untuk CORBA. Compiler yang akan digunakan adalah MICO-C++ dari Mico.org. Compiler ini free dan akan dicoba untuk dimanfaatkan untuk memberdayakan C++ untuk aplikasi terdistribusi.

Kata kunci : CORBA, C++, Pemrograman Terdistribusi, Mico

Sistem Komputasi Terdistribusi

Pada sistem client-server dengan server tunggal, server akan memiliki beban yang semakin berat jika semakin banyak aplikasi yang ada di server dan semakin banyak client yang me-request aplikasi-aplikasi tersebut. Salah satu solusi untuk bisa mengatasi masalah tersebut adalah dengan memanfaatkan sistem komputasi terdistribusi. Dalam sistem ini, aplikasi-aplikasi akan didistribusikan secara fisik maupun logik. Secara fisik, aplikasi akan didistribusikan ke beberapa mesin, sehingga server akan merupakan sebuah kesatuan yang terdiri dari beberapa mesin. Salah satu deskripsi sederhana untuk menjelaskan sistem terdistribusi ini adalah sistem layanan nasabah di sebuah bank. Teler ibarat sebuah server yang digunakan untuk melayani berbagai macam transaksi, seperti stor tabungan, transfer, dan pengambilan tabungan. Transaksi-transaksi itu bisa dianggap sebagai aplikasi-aplikasi yang bisa dilakukan oleh sebuah server. Jika nasabah yang antri untuk dilayani semakin banyak sementara teler hanya satu orang, maka beban teler akan berat, antrian akan sangat lama untuk bisa dilayani semua. Solusinya adalah dengan menambah beberapa teler, sehingga antrian bisa didistribusikan ke beberapa teler itu. Dalam konteks sistem terdistribusi secara logik, sistem akan dibagi-bagi berdasarkan aplikasi logik, sistem model ini tidak memandang apakah setiap aplikasi itu berada di mesin yang sama atau berbeda.

Untuk membangun aplikasi komputasi terdistribusi, maka memerlukan framework yang bisa mendukung integrasi dari beberapa aplikasi. Salah satu framework yang digunakan (dan yang akan dipakai) dalam tulisan ini adalah arsitektur CORBA (Common Object Request Broker Architecture).

Arsitektur CORBA

CORBA pertama kali dikembangkan oleh OMG (Object Management Group), yaitu sebuah konsorsium dari beberapa perusahaan software besar, seperti SUN Microsystem, IBM dan termasuk Microsoft, walaupun kemudian Microsoft membuat sebuah arsitektur sendiri yang diberi nama DCOM (Dynamic Common Object Model).

Awalnya ada belasan perusahaan, tetapi sekarang ada sekitar lebih dari 800 vendor yang mendukung arsitektur CORBA ini.

Salah satu isu dikembangkannya arsitektur CORBA adalah sistem terdistribusi yang bisa didukung oleh banyak platform pemrograman, dan oleh banyak vendor perusahaan software. CORBA ingin mewujudkan bahwa beberapa vendor software maupun platform yang berbeda tetapi antara satu dengan lainnya diharapkan bisa salah melakukan interoperasi. Beberapa isu yang lain yang dialamati CORBA adalah :

- orientasi objek, aplikasi CORBA dibangun oleh objek-objek
- independensi hardware, sistem operasi
- transparansi distribusi

CORBA terdiri dari empat komponen utama, yaitu ORB (Object Request Broker) yang merupakan jalan raya objek-objek untuk bisa saling berkomunikasi, COS (Common Object Services) yang merupakan layanan-layanan yang bisa dipakai untuk komunikasi antar objek tersebut, CORBA Facilities yang merupakan fasilitas-fasilitas yang bisa digunakan oleh objek-objek yang dibangun, dan Object Application yang merupakan objek-objek yang instan untuk dapat langsung digunakan oleh programmer.

Empat komponen tersebut ada ditataran konsep, sedangkan untuk membangun aplikasi real berbasis CORBA yang dibutuhkan adalah sebuah komponen yang disebut IDL (Interface Definition Language).

IDL (Interface Definition Language)

IDL adalah bukan bahasa pemrograman. IDL hanya bahasa untuk mendeklarasikan atau mendefinisikan interface-interface yang akan digunakan. IDL adalah bagian paling utama yang harus dikembangkan ketika pertama kali akan mengembankan sistem komputasi terdistribusi berbasis CORBA.

Dalam IDL hanya menyebutkan interface yang akan dijadikan layanan (what), tidak menjelaskan bagaimana detail layanan itu (How). Script IDL memang mirip script bahasa C, tetapi bukan bahasa C. IDL ini akan dikompilasi dengan IDL Compiler yang dimiliki setiap software ORB. Jika yang akan digunakan untuk membangun sistem terdistribusi adalah bahasa Java maka IDL Compiler akan men-generate class-class bahasa Java. Begitu pula jika untuk bahasa C++. IDL Compiler yang mendukung bahasa C++ adalah MICO-C++ dari Mico.org. MICO ini akan men-generate class C++ yaitu *.h dan *.cc. Maka untuk bisa membangun aplikasi berbasis framework CORBA, seorang programmer harus mengetahui terlebih dahulu konsep pemrograman IDL.

Teknik Pemrograman C++ Terdistribusi

Langkah-langkah teknis pemrograman C++ dengan menggunakan Compiler MICO-C++ adalah sebagai berikut:

- membuat file IDL dan mengkompilasi file tersebut.
- Membangun aplikasi server.
- Membangun aplikasi client
- Mengkompilasi file-file aplikasi server dan client yang sudah dibuat itu.

Selain langkah-langkah diatas beberapa konsep pemrograman yang penting untuk membangun aplikasi CORBA adalah pemrograman IDL (interface definition language). IDL ini sangat penting dan pertama kali yang harus dipersiapkan pada saat mulai pengembangan program.

Hal sangat penting dan perlu diperhatikan dalam konsep pemrograman distribusi berbasis CORBA adalah CORBA memisahkan interface dan implementasi dalam modul terpisah. Berbeda dengan konsep Remote Method Invocation (RMI) yang dimiliki Java Teknologi. RMI menggabungkan interface dengan implementasi.

Implementasi Pemrograman C++ Untuk Komputasi Terdistribusi

a. Analisa Kebutuhan sistem

Sebelum melangkah ke implementasi sistem beberapa perangkat lunak yang perlu dipersiapkan adalah :

- Compiler C++, karena sistem akan dikembangkan dibawah sistem operasi Linux, salah satu yang ada di paket linux untuk compiler C++ adalah g++.
- Perangkat lunak ORB C++ yang akan digunakan yaitu MICO-C++ yang bisa didownload secara gratis dari mico.org. Kemudian diinstalasi di sistem operasi linux. Untuk pengujian client server dengan 2 komputer atau lebih maka MICO-C++ ini harus diinstalasi pada semua komputer yang akan digunakan. MICO-C++ yang digunakan adalah versi 2.3.11 yang sangat selaras dan mendukung CORBA spesifikasi 2.3.
- Implementasi yang akan dipakai untuk uji coba program adalah aplikasi perbankan. Dalam aplikasi ini ada beberapa layanan diantaranya withdraw, balance dan deposit.

b. Pemrograman MICO-C++

Sesuai dengan analisa kebutuhan sistem diatas maka aplikasi yang akan dipakai adalah bank account. Bank account menawarkan 3 layanan, withdraw yaitu pengambilan sejumlah uang, deposit yaitu menyimpan sejumlah uang, dan balance yaitu aplikasi bank untuk menghitung jumlah uang yang ada.

Langkah pertama adalah membuat file IDL. Untuk aplikasi bank account ini maka file IDLnya adalah sebagai berikut:

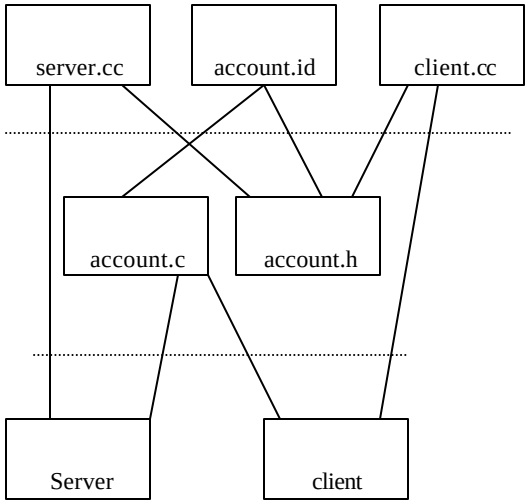
```
interface Account {
      void deposit (in unsigned
      long amount);
      void withdraw(in unsigned
      long amount);
      long balance();
      };
```

kita dapat melihat bahwa aplikasi bank account ini memiliki 3 layanan deposit dan withdraw dengan nilai input amount, dan layanan balance. File IDL ini disimpan dengan nama file account.idl dan kemudian dikompilasi dengan compiler IDL yang ada didalam MICO-C++. Kompilasi yang dilakukan adalah sebagai berikut:

```
c/>idl --boa –nopoa account.idl
```

hasil kompilasi file tersebut akan men-generate file account.h dan account.c. Dalam file-file itu memuat deklarasi class sebagai dasar class untuk implementasi objek account. Untuk setiap layanan/interface yang dideklarasikan di file idl maka IDL Compiler akan men-generate sebuah C++ class.

Untuk lebih jelas bagaimana aplikasi MICO ini dibangun, maka dapat diperhatikan gambar berikut:



Gambar pembuatan implementasi MICO

Langkah berikutnya adalah membangun aplikasi client dan server. Agar client bisa memanggil aplikasi server maka client harus tahu identitas server yang akan dipanggil. Kalau dalam jaringan komputer kita mengenal IP Address untuk mengalamati sebuah host dalam sebuah jaringan, dalam CORBA pengalamatan ini bisa menggunakan sebuah layanan yang disediakan oleh CORBA Common Object Services. Layanan untuk pengalamatan ini adalah Naming Service (layanan penamaan). Naming Service ini akan memberi sebuah ID (identitas) sesuai dengan format CORBA. Salah satu metode agar programmer tahu seperti apa pengalamatan yang dilakukan oleh CORBA adalah Stringification. Stringification ini mengubah alamat objek (yang akan dipanggil) ke bentuk string, kemudian disisi client string ini akan di konversi lagi oleh CORBA ke format aslinya. Dalam implementasi pemrograman method ini dilakukan dengan cara memanggil string_to_object dan object_to_string.

Untuk aplikasi server maka program bank account ini bisa ditulis sebagai berikut:

```
// file account_server.cc

#include <iostream.h>
#include <fstream.h>
#include "account.h"

class Account_app : virtual public
Account_skel
{
//Definisi implementasi Account
Account::Account()
{
      _current_balance = 0;
}
void Account::deposit(unsigned long
amount)
{
      _current_balance += amount;
}
void Account::withdraw(unsigned long
amount)
{
      _current_balance -= amount;
}
long Account::balance ()
{
      return _current_balance;
}

int main(int argc, char *argv[])
{
// inisialisasi ORB
CORBA::ORB_var orb = CORBA::init (argc,
argv, "mico-local-orb");
//inisialisasi ORB
CORBA::BOA_var boa = orb->BOA_init
(argc, argv, "mico-local-boa");
//Inisialisasi Object adapter
Account_app* server = new
Account_app;
//objek server
CORBA::String_var ref=orb-
>object_to_string (server);
//panggil method stringification
//untuk memberi ID ke objek server
ofstream out("/home/maman/tmp/
account.objid");
//simpan di direktori
out << ref << endl;
//tampilkan refensi objek ID
out.close();

boa->impl_is_ready
(CORBA::ImplementationDef::nil());
//menginisialisasi object adapter

orb->run;
//running ORB untuk memposisikan
//server menjadi posisi
//mendengarkan panggilan
//dari client

CORBA::release(server);
//melepaskan panggilan jika sudah
// selesai
return 0;
}
```

sedangkan untuk aplikasi client harus dibuat sebuah program yang memanggil ID server dan memanggil aplikasi yang ada di server .

```
//file account_client.cc
#include <iostream.h>
#include <fstream.h>
#include "account.h"

int main(int argc, char *argv[])
{ //main method

CORBA::ORB_var orb = CORBA::init (argc,
argv, "mico-local-orb");
//inisialisasi ORB

CORBA::BOA_var boa = orb->BOA_init
(argc, argv, "mico-local-boa");
//inisialisasi objek adapter

ifstream in
```

```
("/home/maman/tmp/account.objid");
char_ref[1000];
in >> ref;
in.close();
//membaca file objek ID yang masih
//dalam bentuk string

CORBA::Object_var obj =
orb->string_to_object (ref);
//menkonversi string ke objek ID

Account_var client =
Account::_narrow(obj);
//objek client

client->deposit (1000);
client->withdraw (500);
cout << "Balance is " <<
client->balance() << endl;
//memanggil aplikasi sambil memberi
//nilai untuk diproses

return 0;
}
```

Langkah berikutnya adalah mengkompilasi program server dan client yang sudah dibuat menjadi objek (.o). Untuk mengkompilasi dengan menggunakan MICO-Compiler maka digunakan perintah berikut:

```
mico-c++ -I. –c account_server.cc –o
account_server.o

mico-c++ -I. –c account_client.cc –o
account_client.o

mico-c++ -I. –c account.cc –o account.o
```

objek-objek yang sudah dibuat kemudian dikompilasi lagi untuk membuat sebuah program yang bisa dieksekusi. Perintahnnya adalah:

```
mico-ld –o server account_server.o
account.o –lmico2.3.11

mico-ld –o client account_client.o
account.o –lmico2.3.11
```

Pengujian

Setelah semua class yang di-generate oleh Compiler IDL di kompilasi, aplikasi server dan client juga didevelop dengan acuan kerangka dari class-class yang sudah ada, langkah berikutnya adalah running program dan pengujian program. Program yang harus pertama kali dijalankan adalah class Server, berikutnya baru dari sisi client dijalankan aplikasi client. Untuk pengujian ini hal yang pertama kali harus di perhatikan adalah koneksi dari client ke server, dalam hal ini kemampuan untuk sebuah objek koneksi ke objek lain dan mampu melakukan interoperasi, atau dalam istilah lain disebut interoperabilitas. Interoperabilitas adalah kemampuan antara 2 sistem atau lebih untuk bisa saling bertukar informasi dan menggunakan informasi tersebut.

Menjalankan Server

Server bisa dijalankan dengan memanggil class aplikasi server sesuai dengan perintah berikut:

server

Menjalankan Client

Client bisa dijalankan dengan memanggil class aplikasi client dengan perintah sebagai berikut:

```
client
```

Aplikasi client bisa dilakukan di sebuah komputer yang sama dengan server atau di mesin komputer yang berbeda.

Respon dari hasil setelah client dijalankan adalah sebagai berikut:

```
Balance is 500
```

Sedangkan jika server dimatikan respon yang muncul adalah sebagai berikut:

Analisa dan Pembahasan

Dengan menggunakan IDL Compiler milik MICO maka secara otomatis file idl, yang memuat deklarasi interface-interface yang akan digunakan, akan membangkitkan class-class dasar yang akan digunakan untuk hubungan client/server. Programmer tidak perlu sulit untuk memikirkan class-class apa yang harus dibuat untuk antarmuka dan bagaimana menghubungkan client/server dengan memanfaatkan antarmuka yang ada. Yang perlu dilakukan hanya mendeklarasikan interface ke dalam file idl, kemudian di kompilasi.

File-file yang digenerate oleh compiler MICO IDL secara otomatis akan memisahkan client dan server. Bagian client ini kemudian disebut client stub, dan bagian untuk server disebut server skeleton, kedua bagian ini merupakan kerangka yang akan digukan arsitektur CORBA untuk bisa mengintegrasikan semua objek yang dikembangkan.

Pemrograman C++ memang cukup powerful untuk aplikasi client/server terdistribusi, namun disisi client interfacenya sederhana. Untuk bisa membangun aplikasi berbasis web maka disisi client harus menggunakan bahasa pemrograman yang mendukung untuk aplikasi web. Misalnya Java dengan teknologi JSP dan Servletnya bisa digunakan untuk membangun aplikasi berbasis web. Dan disisi server tetap bisa menggunakan C++ untuk aplikasinya. Hal ini bisa dilakukan dengan menggunakan CORBA.

Daftar Pustaka

---, Mico Documentaion, http://mico.org

D. Allen, 1996, *CORBA Technology for Cross-domain Interoperability in Embedded Military Systems, and Issues in Its Use,* Proceeding of WORDS '96 Second Workshop, pp. 173-178.

S. Baker, Addison Wesley, November 1997, *CORBA Distributed Objects: Using Orbix.*

Cetus Links, CORBA Links, <URL: http://www.cetus-links.org/oo_corba.html>.

Chris Exton, Damien Watkins and Dean Thompson, Department of Software Development, Faculty of Computing & Information Technology,

Monash University Australia, 1997, *Comparisons between CORBA IDL & COM/DCOM MIDL Interfaces for Distributed Computing*

Ivor Horton, Wrox Press Ltd., 2000, *Beginning Java 2 – JDK 1.3 Edition.*

T. Mowbray and T. Brando, 1993, Object Magazine, *Interoperability and CORBA – Based Open System,* pp. 50-54.

OMG Links, OMA Links <URL: http://www.omg.org/omaov>

OMG Links, <URL: http://www.omg.org>

J. Siegel, John Wiley and Sons, April 1996, *CORBA Fundamental and Programming.*

Suhail M. Ahmed, Sams Publishing, 1998, *CORBA Programming Unleashed*.

Andrew S. Tanembaum, Maarten Van Steen, Prentice Hall, 2002, *Distributed System, Principles and Paradigms.*