Tema de casă #4 – Serviciu de planificare a aplicațiilor bazat pe CORBA

Responsabil de temă: Florin Pop (florin.pop@cs.pub.ro)

Data publicării: 09.12.2008
Data ultimei modificări a enunțului: 09.12.2008
Termenul de predare: 12.01.2009

Obiective

După realizarea acestei teme de casă studentul va fi capabil să:

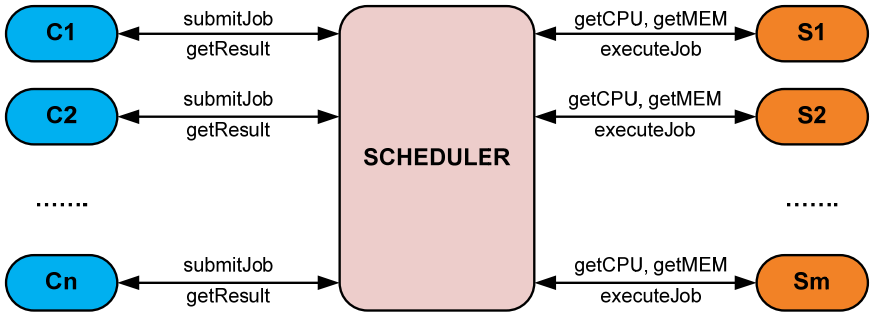
- realizeze o aplicație client-server bazată pe CORBA.
- gestioneze interacțiunea dintre mai mulți clienți şi mai multe servere.

Cunoştințele necesare rezolvării acestei teme de casă:

- Programare în JAVA.
- Noțiunea de socket TCP

Enunțul problemei

Folosind CORBA, să se scrie un planificator de aplicații conform cu arhitectura din figura:



Se va implementa un program client (pentru entitățile C), un program server (pentru entitățile S) şi un program planificator (pentru entitatea SCHEDULER) care să funcționeze astfel:

Un Client are un o aplicație de executat. El trimite aplicația Planificatorului care va alege Server-ul optim şi îl va da acestuia spre execuție. Server-ul va executa aplicația şi va întoarce rezultatul către Planificator. Clientul, după ce trimite o aplicație îşi continuă activitatea. El are posibilitatea de a interoga status-ul aplicației la orice moment de timp.

Un server va oferi următoarele servicii:

- `getCPU`: întoarce suma intre `CPU_sys`, `CPU_nice` si `CPU_usr` (obținute pentru server), valorare care reprezintă CPU_idle.
- `getMem`: întoarce Mem_free (pentru server)
- `executeJob`: execută a aplicație

Planificatorul va oferi următoarele servicii:

- `submitJob`: se trimite o aplicație unui server.
- `getResult`: se cere rezultatul aplicației (la un moment de timp ulterior); metoda poate întoarce rezultatul sau un mesaj de genul "`Not completed yet`"

La pornire/oprire un Server va anunța Planificatorul care va menține intern o listă de Servere active la un moment dat. Pentru selecția Server-ului optim, Planificatorul va calcula pentru fiecare server *i*:

$$F(i) = 3*CPU\_idle(i) + 2*Mem\_free(i)$$

şi va selecta serverul cu valoarea maximă.

O clasa Job (care reprezintă aplicația ce se va executa) implementează următoarea interfața:

```
interface Job {
     Object doWork() {
          ...
     }
}
```

De exemplu, metoda doWork va "dormi" un interval random de timp şi apoi întoarce numele Server-ului care execută aplicația.

5. Precizări legate de implementarea temei

- Pentru implementare se va folosi CORBA şi JAVA. Un exemplu de `makefile` pentru compilarea aplicației ar putea fi:

```
all:
    idlj -fall Scheduler.idl
    javac *.java
    cd Scheduler/ ; javac *.java ; cd ..

start ns:
    tnameserv -ORBInitialPort 15000

run_server:
    java Server -ORBInitialPort 15000

run_client:
   java Client

run_admin:
    java Scheduler -ORBInitialPort 15000 -ORBInitialHost localhost

clean:
    rm -rf *.class *~;rm Banca/*;rmdir Banca;
```

- Specificațiile IDL vor fi propuse de voi.