O arhitectură client-server two-tier portabilă pentru gestiunea sarcinilor la distanță. eRIMP

Ciprian Pungilă
Universitatea de Vest
Facultatea de Matematică-Informatică
Timişoara, 2007
ciprianpungila@yahoo.com

Abstract

Arhitectura client-server este o paradigmă des întâlnită în era informațională actuală, deosebit de utilă atunci când vine vorba de modelarea unor procese sau activități distribuite, sau de controlul workflow-ului într-o rețea de calculatoare. O altă utilizare importantă a aceste arhitecturi o reprezintă controlul încărcării în sistemele distribuite, respectiv gestiunea centralizată a informației. Articolul de față îşi propune să prezinte o arhitectură client-server portabilă pentru gestiunea la distanță a sarcinilor, atât pe sistemele actuale Windows, cât şi pe sistemele UNIX, împreună cu o aplicație client-server care o modelează.

1 Introducere

Arhitectura client-server este o arhitectură computațională care presupune unul sau mai multe procese client, aflate pe calculatoare client, invocând servicii de la un proces server, aflat pe un calculator conectat în rețea cu calculatoarele client. Arhitectura derivă din programarea modulară, model care presupune divizarea codului sursă în mai multe părți (identificate ca fiind module), pentru o mai uşoară gestiune ulterioară şi o mai facilă dezvoltare de cod ulterioară. Un exemplu este redat în figura 1.

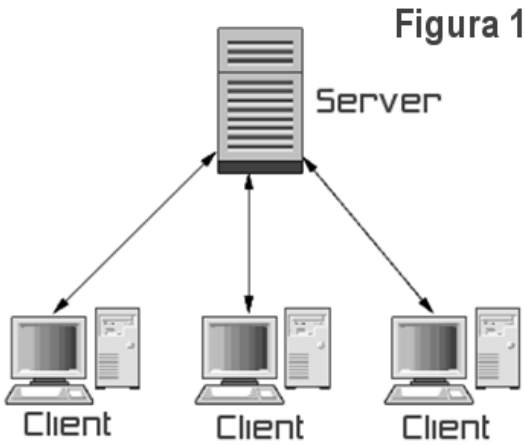


Figura 1

Un domeniu important al tehologiei informaționale este cel legat de grafică şi de procesarea grafică. Domeniile de maxim interes în acest scop includ procesarea digitală a imaginilor şi animația computerizată. În acest scop există numeroare stații dedicate prelucrărilor grafice, cele mai cunoscute fiind cele de la Sillicon Graphics, stații care folosesc o arhitectură multi-procesor împreună cu plăci grafice corelate cu acceleratoare bi şi tri-dimensionale.

Un factor determinant pentru orice sistem şi pentru orice prelucrare, de orice fel, este costul. Ca şi alți factori la fel de semnificativi, el este unul dintre elementele determinante atunci când vine vorba de a alege un sistem dedicat prelucrărilor digitale sau de alt tip. În general însă, stațiile grafice dedicate exclusiv

algoritmilor intens computaționali folosesc cel puțin patru procesoare, împreună cu un sistem de operare stabil şi eficient din punct de vedere al optimizării la nivel elementar (inline) al codului.

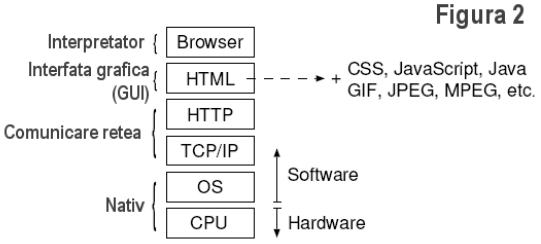
O arhitectură two-tier este o arhitectură în care clientul comunică direct cu serverul, fără intervenția unui alt server. Este o arhitectură folosită în general în mediile cu cel mult 50 de utilizatori, pentru a preveni supraîncărcarea serverelor. Dacă se doreşte o mărire a numărului de clienți, este necesar să se folosească o arhitectură three-tier.

Plecând de la aceste considerente, propunem o arhitectură client-server two-tier portabilă (multi-platformă), dedicată prelucrărilor intens computaționale, pe care o aplicăm în particular procesării în format digital a imaginilor, dezvoltată în limbajul Delphi şi respectiv C++. Mediile de dezvoltare folosite la implementare vor fi Delphi 7 Enteprise (pentru partea de client, respectiv server Windows a aplicației) şi Kylix C++ 3 Enterprise (pentru partea de server a aplicației, pe sistemele de operare UNIX).

2 Activități similare

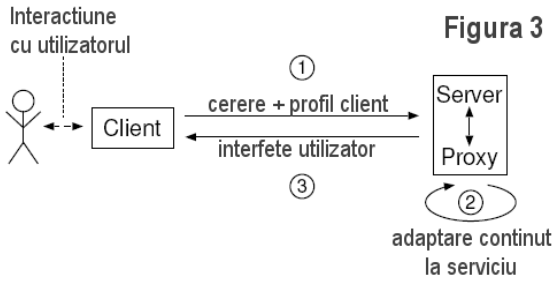
Arhitectura client-server a fost implementată în numeroase sisteme de calcul existente la ora actuală, unul dintre cele mai simple fiind protocolul HTTP de transfer a informației pe Internet. Clientul (web-browserul de pe maşina client) face o cerere la serverul HTTP (care rulează pe maşina server), urmând ca serverul să îi răspundă cu conținutul text al paginii pe care utilizatorul încearcă să îl acceseze. De notat însă că această arhitectură nu este two-tier, în general între client şi server interpunându-se numeroare alte servere (de exemplu DNS).

Figura 2



În contextul modelului client-server de tip cerere HTTP, un model este redat în figura 3:

Figura 3



Un alt exemplu tipic îl reprezintă sistemele de mesagerie instantă, ca de exemplu ICQ sau AIM, sisteme care se bazează pe o arhitectură two-tier simplă, compusă dintr-un server care gestionează conexiunile de la clienți (programele de mesagerie care rulează pe calculatoarele client), respectiv acceptă mesajele utilizatorilor client şi le redirecționează spre clientul destinație, totul în timp real.

Sistemul pe care îl abordăm în lucrarea de față îşi propune să faciliteze accesul la serviciile oferite de o stație dedicată procesării grafice (intens computaționale) de mare performanță, considerată ca server al arhitecturii (de obicei multi-procesor, şi rulând sistemul de operare UNIX), unui număr de clienți (reprezentând stații cu performanțe mici sau medii, pe care rulează sistemul de operare Windows) conectați la server şi interacționând cu acesta pentru procesarea digitală a unora sau mai multor imagini pe care clientul le solicită serverului. Până la ora actuală, nu se cunoaşte o altă inițiativă

open-source pentru un astfel de model client-server.

3 Componente de bază

Componentele de bază ale unei arhitecturi client-server sunt:

- o combinație de front-end pe partea de client, care interacționează cu utilizatorul, şi un server în fundal, care interacționează cu resursa partajată;
- sarcinile front-end şi back-end au în general necesități diferite pentru a calcula viteza procesorului, memoria, viteza discului şi capacitățile acestuia, şi dispozitivele de intrare/ieşire;
- mediul este în general eterogen şi multivendor; platforma hardware şi sistemele de operare ale clientului şi serverului în general nu sunt aceleaşi; clientul şi serverul comunică printr-o interfață standard de comunicare (API) şi apeluri de proceduri la distanță (RPC).

Aplicația pe care am dezvoltat-o se numeşte eRIMP (enhanced Remote Image Manipulation Program) şi implementează o arhitectură client-server pentru a permite manipularea imaginilor la distanță. Serverul a fost construit pentru platformele UNIX în principal, însă există o variantă de server (care însă nu va fi menținută în versiunile ulterioare), dezvoltată în Delphi 7 Enterprise, care facilitează rularea serverului şi pe sistemele de operare Windows.

4 Aplicația client

Aplicația client este importantă mai ales din punct de vedere ergonomic, ea oferind un mediu de lucru utilizatorui final. Interacțiunea presupune mai mulți paşi, dintre care:

- selectarea unui efect grafic din cele disponibile pe server, pentru procesarea imaginii;
- selectarea imaginii care se doreşte a fi procesată pe partea de server;
- arhivarea imaginii şi trimiterea ei la server pentru procesare;
- aşteptarea imaginii procesată de la server;
- salvarea imaginii de către utilizator la o destinație dorită.

Din punct de vedere funcțional, aplicația client funcționează pe baza unui fişier de configurare care specifică adresa IP a serverului din rețea, respectiv portul unde clientul se poate conecta la server. În fişierul de inițializare se specifică de asemenea şi setările pentru procesarea grafică efectuată ultima oară de acea stație client.

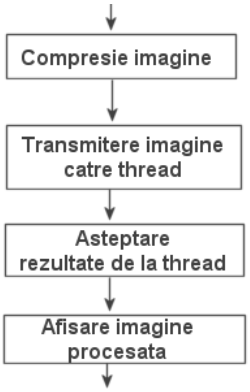
Fişierul de configurare este în format Windows INI standard, având câmpurile de mai jos:

```
[Effects]
EffectParameter=25
[Connection]
ServerIP=127.0.0.1
Port=7170
ReadTimeout=0
RecBufferSize=32768
SendBufferSize=32768
```

Printre elementele configurable pe partea de client se numără şi timpul de time-out al aplicației în cazul în care serverul nu răspunde la o cerere de conectare din partea clientului, respectiv dimensiunea zonei tampon implicite pentru transferul de date, dimensiune care depinde în general de arhitectura rețelei în care sistemul este folosit, precum şi de calitatea conexiunii şi de parametrii de performanță

ale dispozitivelor (routere, switch-uri, etc.) folosite pentru transfer; nu în ultimul rând, gradul de încărcare al rețelei şi dimensiunea zonei tampon implicite a sistemului de operare afectează aceşti doi parametrii.

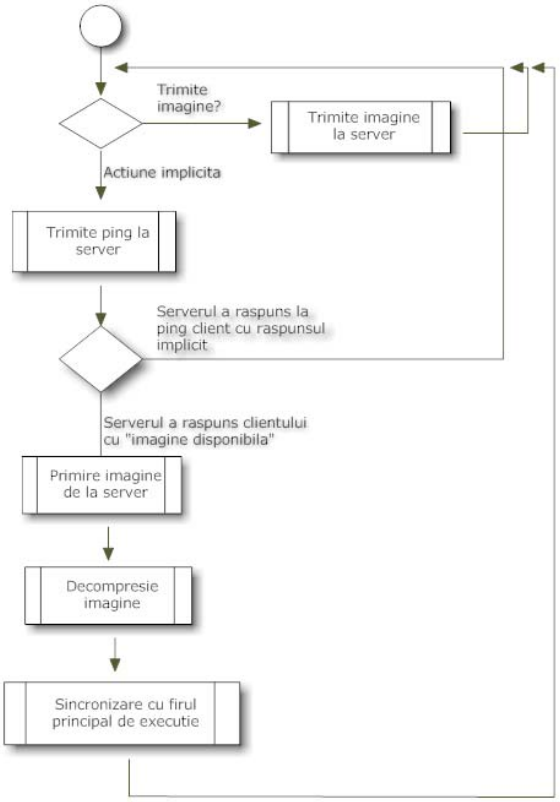
Algoritmul implementat pe partea de client este redat schematic mai jos:



Fiecare proces client care rulează pe stațiile client are un fir de execuție (implementat printr-un obiect, instanță a unei clase TWorkThread care extinde TThread) care se ocupă cu două sarcini importante:

- Are rol de menținere activă a conexiunii (pentru a nu permite time-out), prin ping-uri consecutive la server, la intervale stabilite de timp.
- În cazul setării de către firul principal a flag-ului pentru trimitere imagine, va trimite imaginea către server şi va reveni la ciclul normal de funcționare.
- În cazul recepției, în urma unui ping la server, a unui răspuns pentru primirea unei imagini, o va recepționa (arhivată ZLib), o va dezarhiva şi, într-o metodă sincronizată cu firul de execuție principal, va permite refresh-ul pe partea de client a interfeței, pentru a afişa noua imagine, procesată digital, aşa cum a fost ea recepționată de client.

Schematic, acest algoritm este prezent în figura de mai jos:



După ce procesul client a trimis imaginea la server, în timp ce firul de execuție aşteaptă primirea imaginii procesate de la acesta, utilizatorul va fi informat că aplicația aşteaptă răspuns de la server prin intermediul unei căsuțe de dialog. Sincronizarea celor două fire de execuție este vitală şi previne deadlock-ul şi bottleneck-urile posibile în situațiile de transfer, respectiv de actualizare a ferestrei principale. Pentru a asigura sincronizarea, mediul de dezvoltare pune la dispoziție methoda Synchronize(), care permite accesul exclusiv al unui fir de execuție la interfața grafică (GUI).

5 Aplicația server

Aplicația server este nucleul funcționării corecte şi eficiente a arhitecturii propuse. Layer-ul de comunicare cu clienții este realizat printr-un obiect instanță a unei

clase TIdTCPServer, care pune la dispoziție funcții trigger (declanşatoare, sau "evenimente" în limbaj standard Delphi) pentru conectarea unui client la server, deconectarea lui, etc.

Aplicația server a fost scrisă pe un sistem de operare Mandrake 8.2, folosind Kylix C++ 3 Enterprise. Au fost necesare şi librăriile QT pentru anumite obiecte din program.

Serverul menține în permanență o listă cu clienții conectați, precum şi un flag care specifică dacă un client a trimis o sarcină serverului sau nu. De asemenea, la selectarea unui client din lista de clienți vor fi disponibile informații despre data conectării, data ultimei solicitări, nume host şi nume calculator client. Fiecare client conectat are un fir de execuție asociat pe partea de client, fir care este plasat într-un obiect instanță a clasei TThreadList.

În momentul conectării unui client, serverul creează un fir de execuție în lista curentă pentru el, asociind cu acest fir informații despre client; ulterior, serverul răspunde clientului cu lista de efecte digitale disponibile, acestea fiind afişate (şi configurate corespunzător) pe partea de client. Lista de efecte este disponibilă într-un fişier de inițializare Windows standard (INI), cu structura:

```
[Grayscale]
Author=Ciprian Pungila
Description=Grayscale the image
Customizeable=No
```

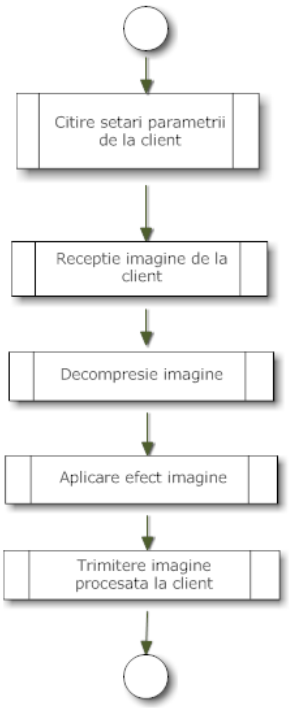
În momentul deconectării unui client de la server, acesta îşi actualizează lista internă de fire de execuție (denumită Clients), eliminând firul de execuție asociat cu clientul.

Evenimentul OnExecute al TIdTCPServer permite execuția unei funcții handler pentru o comandă recepționată de la un client, al cărui fir de execuție este transmis ca şi parametru (TIdPeerThread). În acest moment sunt actualizate informațiile despre client, precum timpul ultimei acțiuni, iar serverul răspunde adecvat comenzii primite:

- dacă este o comandă de tip *ping*, serverul răspunde cu răspunsul implicit;
- dacă este o comandă *Process Image*, serverul va crea un fir de execuție pentru clientul curent, prin crearea unei instanțe a TProcessThread.

Obiectele de tip TProcessThread sunt instanțe ale unei superclase a TThread, cu un constructor propriu, pentru a specifica informații despre clientul asociat cu firul de execuție al sarcinii.

Modul în care firul de execuție asociat unei sarcini provenite de la un client îşi îndeplineşte sarcinile este redat în diagrama de mai jos:



Datorită modului de implementare a acestei arhitecturi pe partea de server, funcționarea lui este net superioară dacă se utilizează sisteme UNIX cu o implementare eficientă a firelor de execuție la nivelul kernelului, implementare care, pe un sistem

multi-procesor, ar permite o partajare echilibrată a sarcinilor pe unul din procesoare. În sistemele actuale, procesoarele pun la dispoziție nuclee interne separate pentru execuția în paralel a instrucțiunilor, iar un sistem de operare dedicat foloseşte aceste facilități pentru programarea sarcinilor adecvat. Unul din aceste sisteme de operare este Sun Solaris, el fiind folosit pe câteva stații grafice dedicate în mod curent.

Profilul performanță al serverului a fost superior pe un sistem Linux Mandrake 8.2, comparativ cu rularea aceluiaşi server pe un sistem Windows XP, configurația sistemelor de test fiind aceeaşi.

6 Obiective propuse

În viitor ne propunem să îmbunătățim arhitectura propusă, următorul pas constând în scalarea ei la o arhitectură three-tier şi implementarea suportului RPC pentru client şi server.

Alte îmbunătățiri semnificative care vor fi implementate vor fi:

- algoritm optimal pentru controlul încărcării pe partea de server intermediar în arhitectura three-tier;
- modularizare server şi suport pentru plugin-uri;
- mai mulți algoritmi avansați de prelucrare grafică digitală (e.g. "aged newspaper", "buttonize", "chisel", "multiple light source shadowing").

7 Concluzii

Lucrarea de față îşi propune să prezinte un model de arhitectură client-server two-tier portabilă pentru gestiunea sarcinilor la distanță, model care este deopotrivă flexibil şi util pentru aplicațiile practice.

Corelarea arhitecturii cu un sistem de calcul multiprocesor dedicat prelucrărilor de mare putere, dotat cu un sistem de operare capabil să folosească performanțele puse la dispoziție de către acesta duce la o creştere a performanței semnificativă a întregului sistem şi permite centralizarea algoritmilor implementați pe partea de server pe un singur calculator, deşi ne propunem ca versiunile următoare să suporte un sistem complet distribuit.

Aplicația portabilă eRIMP (enhanced Remote Image Manipulation Program), cu server rulând în principal pe stațiile UNIX dedicate şi clienți compuşi din stații Windows, implementează această arhitectură, aşa cum există ea în stadiul curent de dezvoltare, urmând ca versiunile ulterioare să o îmbunătățească şi să ducă la scalarea acesteia într-un mod care să eficientizeze atât sarcina programatorului, cât şi sarcina sistemului de calcul.

8 Referințe

[1] *A client-server architecture for providing client-specific, interactive network services on the application layer*, Roland Haratsch, Technical University of Munich, 2002

[2] *Client/Server Architectures for Business Information Systems*, Klaus Renzel, Wolfgang Keller, 1997

[3] *Client-Server Architecture*, Lawrence Chung, Computer Science Program, The University of Texas, Dallas, 2004