Proiect la Disciplina „Athitecturi Avansate”

Proiect 2022-2023

Autori:Dan-
Alexandru Bucur & TimarCosmin
Profesor
coordonator: Prof. dr. ing. Florea Adrian
Grupa: 243/1&243/2

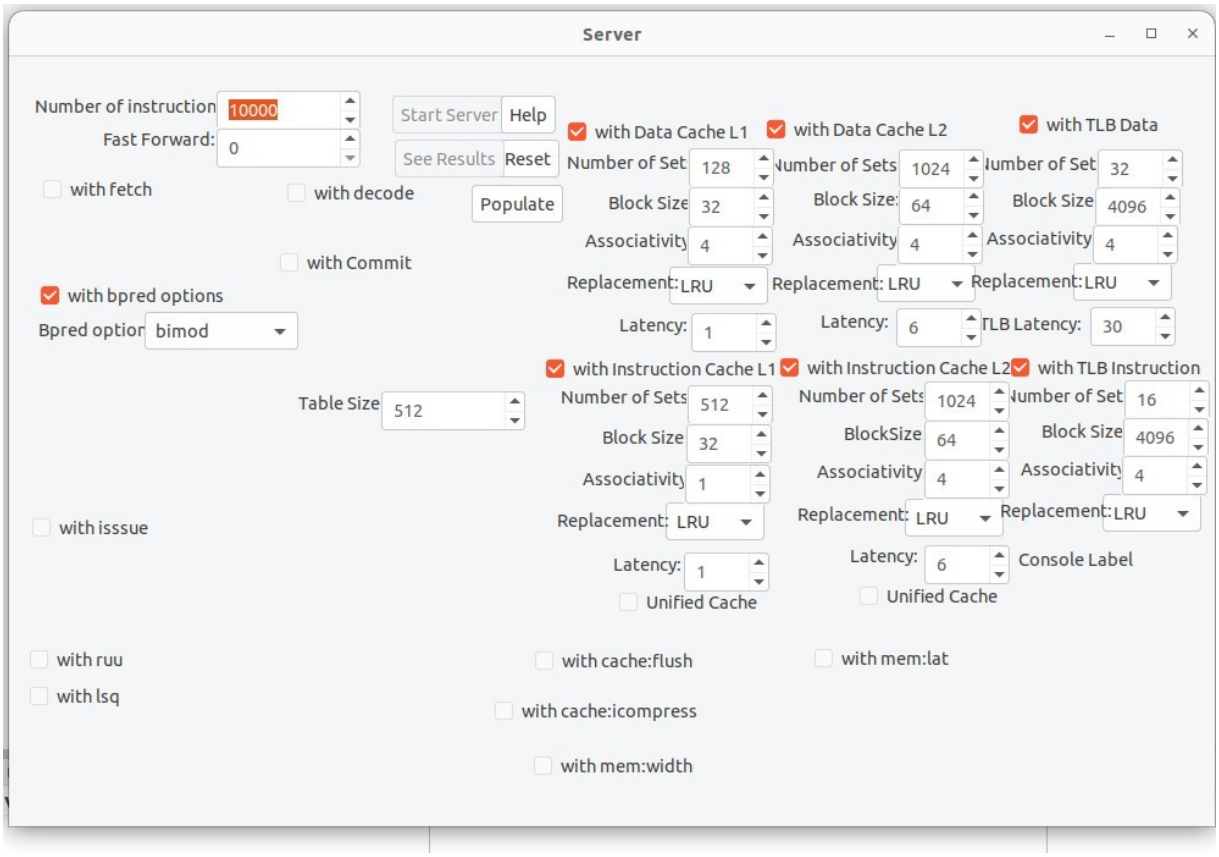
1. Tema proiectului

Tema 9:
Implementarea unei interfețe vizuale aferente simulatorului sim-outorder care simulează execuția out-of-order, interfața procesor-cache, predictor avansat de salturi. Se va permite introducerea parametrilor de editare, simularea în rețea în arhitectură client / server, implementarea unui help atașat, prezentarea grafică a rezultatelor simulării IR(fetch rate), IR(dimensiunea cache-ului). Dezvoltarea se va realiza folosind mediile de programare QT sau MonoDevelop sub SO Linux.

2. Ghid de utilizare

Aplicatia este ușor de folosit, iar pasii în utilizarea aplicației sunt minimalisti: Se porneste aplicatia numita “Server” prin care vom alege din optiunile de simulare pe care le dorim. Prin bifarea check-boxurilor de pe interfata, putem adauga sau elimina unele opțiuni, rămânând la cele prestabilite de simulator.

După ce am configurat după cum am dorit, putem apăsa butonul “Start Server”, moment în care aplicatia va porni serverul, care v-a împarti munca Clientilor. Pentru ca sistemul sa functioneze se va deschide manual câte o instanta a aplicației “Client” pentru fiecare benchmark pe care dorim să rulăm configuratia arhitecturala dorită.



Interfata Server

Dupa ce se pornesc instantele “Client ” se va utiliza butonul “See Results” pentru a vedea rezultatele simularii.

Pentru o mai buna întelegere a modului de utilizare al aplicației, se va folosi butonul de “Help” care va deschide ghidul de utilizare a aplicatiei.

2. Mod de implementare

Pentru realizarea proiectului s-a utilizat mediul de dezvoltare “Monodevelop” sub OS-ul Linux. Pentru controlul versiuni s-a folosit “git”.

Pe partea de server functia “Net” faciliteaza crearea serverului si instantiaza comunicarea între server și client. De asemenea prin variabilele reciveData și responseData se parseaza string-urile trimise

prin TCP/IP folosind un Socket deschis pe portul 8080.
În următoarea imagine se va vedea codul sursa pentru functia Net.

```
public static void Net()
{
    Socket clientSocket = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
    IPEndPoint endPoint = new IPEndPoint(IPAddress.Parse("127.0.0.1"), 8080);

    Console.WriteLine("Connecting to server...");
    clientSocket.Connect(endPoint);
    Console.WriteLine("Connected to server.");

    // Receive response from server
    byte[] responseBytes = new byte[1024];
    int bytesRead = clientSocket.Receive(responseBytes);
    responseData = Encoding.UTF8.GetString(responseBytes, 0, bytesRead);
    Console.WriteLine("Received response from server: " + responseData);

    // Send data to server
    //string data = $"{benchmarkRulatNume} {IR} {RataHitCacheDate} {RataHitCacheInstructiuni}"
    string data = $"{metrics.benchmarkName} {metrics.sim_IPC} {metrics.rataHitDL1} {metrics.rataHitDL2} {metrics.rataH
    Console.WriteLine(data);
    byte[] dataBytes = Encoding.UTF8.GetBytes(data);
    clientSocket.Send(dataBytes);

    // Close the client socket
    clientSocket.Close();
}
```

(Client Side)

```
protected void Net()
{
    // Create a server Socket
    System.Net.Sockets.Socket serverSocket = new System.Net.Sockets.Socket(AddressFamily.InterNetwork,
    IPEndPoint endPoint = new IPEndPoint(IPAddress.Parse("127.0.0.1"), 8080);
    serverSocket.Bind(endPoint);
    serverSocket.Listen(10);

    while (true)
    {
        System.Net.Sockets.Socket clientSocket = serverSocket.Accept();

        string response = commands[i];
        byte[] responseBytes = Encoding.UTF8.GetBytes(response);
        clientSocket.Send(responseBytes);
        i++;

        byte[] buffer = new byte[1024];
        int bytesRead;
        while ((bytesRead = clientSocket.Receive(buffer)) > 0)
        {
            string receivedData = Encoding.UTF8.GetString(buffer, 0, bytesRead);
            responses.Add(receivedData);
        }
        clientSocket.Close();
    }
}
```

(Server Side)

În aplicația “Client” avem clasa “Metrics” care extrage metricile de care avem nevoie, care sunt : IR (instruction rate), Rata de hit în cache-ul de instrucțiuni și Rata de hit în cache-ul de date. În imaginea data se poate observa implementarea de baza folosind membri de tip double pentru o mai buna precizie a valorilor optinute.

```
public class Metrics
{

    public double rataHitDL1 { get; set; }
    public double rataHitDL2 { get; set; }
    public double rataHitIL1 { get; set; }
    public double rataHitIL2 { get; set; }
    public double rataHitDTLB { get; set; }
    public double rataHitITLB { get; set; }

    public string benchmarkName { get; set; }
    public double sim_IPC { get; set; }



    public Metrics parseString(string simulation)
    {
        Metrics data = new Metrics();
        var values = new[] {
            "il1.accesses", "il1.hits",
            "dl1.accesses", "dl1.hits",
            "il2.accesses", "il2.hits",
            "dl2.accesses", "dl2.hits",
            "itlb.accesses", "itlb.hits ",
            "dtlb.accesses", "dtlb.hits",
            "sim_IPC",
            };
        double il1Acc = 0, il2Acc = 0, dl1Acc = 0, dl2Acc = 0, itlbAcc = 0, dtlbAcc = 0;
        double il1Hit = 0, il2Hit = 0, dl1Hit = 0, dl2Hit = 0, itlbHit = 0, dtlbHit = 0,IPC = 0;


        Int32 BufferReader = 131072;
        using (StreamReader st = new StreamReader(simulation, Encoding.UTF8,true,BufferReader))
        {
            string line;
            while ((line = st.ReadLine()) != null)
            {
```

```
    ,,
double il1Acc = 0, il2Acc = 0, dl1Acc = 0, dl2Acc = 0, itlbAcc = 0, dtlbAcc = 0;
double il1Hit = 0, il2Hit = 0, dl1Hit = 0, dl2Hit = 0, itlbHit = 0, dtlbHit = 0,IPC =


Int32 BufferReader = 131072;
using (StreamReader st = new StreamReader(simulation, Encoding.UTF8,true,BufferReade
{
    string line;
    while ((line = st.ReadLine()) != null)
    {
        if (values.Any(line.Contains))
        {
            var arguments = line.Split(new string[] { " " }, StringSplitOptions.Remo
            switch (arguments[0])
            {
                case "il1.accesses":
                    il1Acc = double.Parse(arguments[1]);
                    break;
                case "il2.accesses":
                    il2Acc = double.Parse(arguments[1]);
                    break;
                case "dl1.accesses":
                    dl1Acc = double.Parse(arguments[1]);
                    break;
                case "dl2.accesses":
                    dl2Acc = double.Parse(arguments[1]);
                    break;
                case "itlb.accesses":
                    itlbAcc = double.Parse(arguments[1]);
                    break;
                case "dtlb.accesses":
                    dtlbAcc = double.Parse(arguments[1]);
                    break;

                case "il1.hits":
                    il1Hit = double.Parse(arguments[1]);
                    break;
                case "il2.hits":
                    il2Hit = double.Parse(arguments[1]);
                    break;
                case "dl1.hits":
```

(Clasa Metrics unde se extrag metricile)

După ce se compune și se trimite comanda de la server, clientul creaza un script prin care poate executa o comanda prin care rulează sim-outorder.

```
public static void Main(string[] args)
{
    Console.WriteLine("Welcome to 'Defenetly not a virus team'.");
    Net();
    LaunchSim();
    Console.ReadLine();
}
public static void GenerateScript()
{

    string commnandBuffer = responseData;
    //responseData = string.Empty;
    string cosmin = "/home/timarc/Desktop/MareProiect/SimOutorder/Client/bin/Debug/simplesim-3.0/";
    string alex_desktop = "/home/bellum/Projects/SimOutorder/Client/bin/Debug/simplesim-3.0/";

    File.WriteAllText(PATH_TO_SCRIPT, $"#!/bin/bash\nchmod 777 script.sh\ncd {cosmin}\n" + commnandBuffer);
    File.Exists(PATH_TO_SCRIPT);
}

public static void checkForBenchmark()
{
    string[] benchmark = { "applu_simout.res", "apsi_simout.res", "hydro2d_simout.res", "go_simout.res", "su
    string command = responseData;

    for (int i = 0; i < benchmark.Length; i++)
    {
        if (command.Contains(benchmark[i]))
        {
            theBench = benchmark[i];
        }
    }
    Console.WriteLine(theBench);
}
```

(Creare script.sh din aplicatia Client)

3. Resurse necesare

Resursele necesare sunt aceleași ca si ale aplicației Monodevelop. În crearea proiectului s-au folosit resurse minimaliste și s-au utilizat pachete ale librariei NuGet ca si System.Text, System.Net.Sockets, etc.

Proiectul se poate găsi în arhiva atașată cu proiectul sau în repository-ul de pe github: https://github.com/bellum-games/SimOutorder.

4. Concluzii

Deși este o aplicație simplista prin care se poate folosi simulatorul sim-outorder, el reprezintă doar o interfata usor de folosit și rapidă cand vine vorba de folosirea acestuia. Implementarea client-server este ușor de înțeles și ușor de utilizat dacă este nevoie de extinderea functionalitati, fapt ce se aplica întregii aplicații.

5. Bibliografie

Am folosit acest document cu privire la documentarea noastră în legadura cu mediul de dezvoltare Monodevelop:
https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=c5c8a50f8a80074ff2d795b994182eb4e4c8653f#page=111

De asemenea pentru simulatorul sim-outorder din simple sim 3 am folosit:
https://courses.cs.washington.edu/courses/cse471/07sp/sim/simpleScalar3.0guide.pdf