Proiect la Simularea si Optimizare Arhitecturilor de Calcul

Proiect 2022-2023

Autori: Alexandru Dan Bucur & Timar Cosmin
Grupa: 243/1

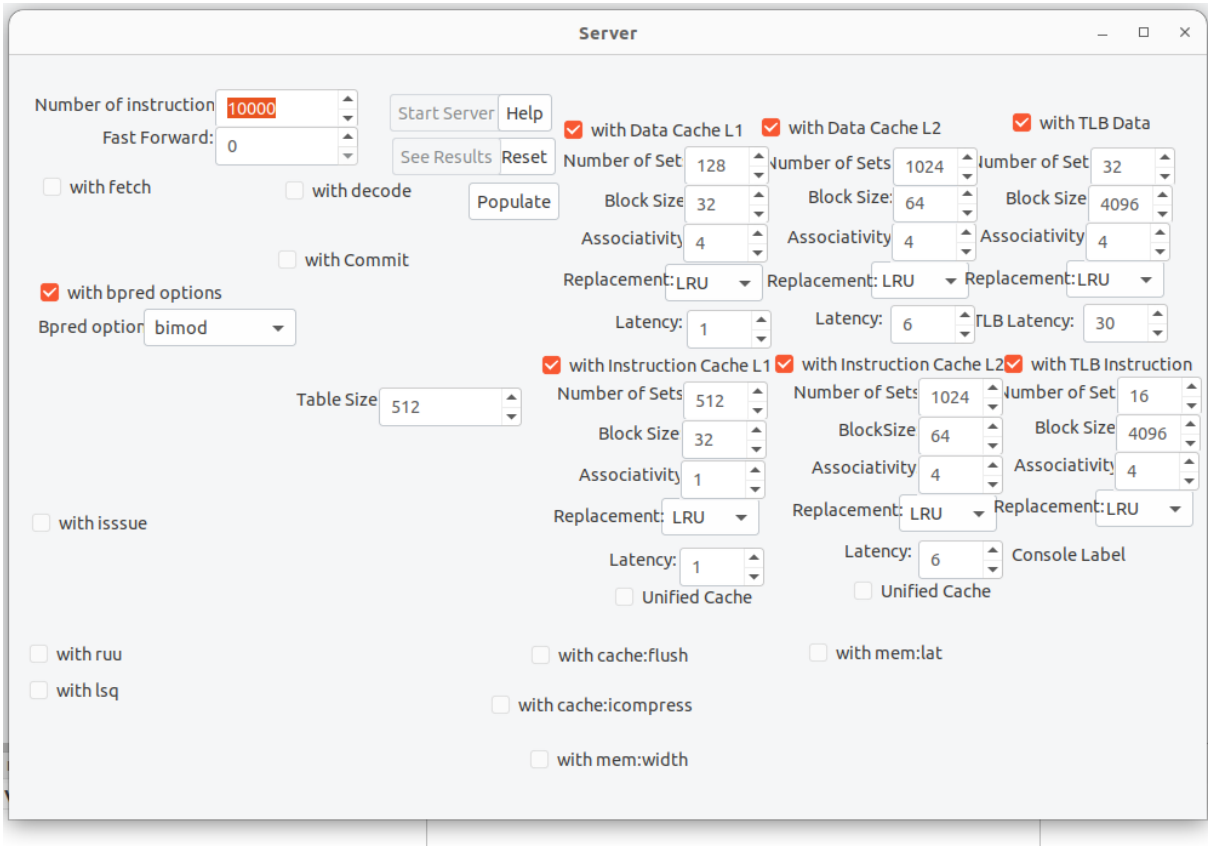
1. Tema proiectului

Tema 9:
Implementarea unei interfeţe vizuale aferente simulatorului sim-outorder care simulează
execuţia out-of-order, interfaţa procesor-cache, predictor avansat de salturi. Se va permite
introducerea parametrilor de editare, simularea în reţea în arhitectură client / server,
implementarea unui help ataşat, 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

Aplicația este foarte cu utilizatorul, iar pașii în utilizarea
aplicației sunt minimaliști: Se porneste aplicatia numita “Server” prin care vom alege din opțiunile de simulare pe care le dorim. Prin bifarea casutelor de pe interfața putem adauga sau scapă de unele opțiuni.

După ce am configurat după cum dorim putem apăsa butonul “Start Server”, moment în care aplicația va porni serverul, care v-a împărți munca Clienților. Pentru ca sistemul sa functioneze se va deschide manual câte o instanță a aplicației “Client” pentru fiecare benchmark pe care dorim să rulăm simularea dorită.



Interfata Server

După ce se pornesc instanțele “Client ” se va utiliza butonul “See Results” pentru a vedea rezultatele simulari.

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

2. Mod de implementare

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

Pe partea de server funcția “Net” facilitează crearea serverului și instantiaza comunicarea între server și client.

```
protected void Net()
{
    // Create a server Socket
    System.Net.Sockets.Socket serverSocket = new System.Net.Sockets.Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
    IPEndPoint endPoint = new IPEndPoint(IPAddress.Parse("127.0.0.1"), 8080);
    serverSocket.Bind(endPoint);
    serverSocket.Listen(10);
    lbConsole.Text += "Server started. Waiting for clients...\n";

    while (true)
    {
        System.Net.Sockets.Socket clientSocket = serverSocket.Accept();
        lbConsole.Text += "Client connected: " + clientSocket.RemoteEndPoint + "\n";

        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);
            lbConsole.Text += "Received data from client: " + receivedData + "\n";
            responses.Add(receivedData);
        }
        clientSocket.Close();
    }
}
```

(Server Side)

```
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.rataHitDTLB} {metrics.rataHitIL1}

    byte[] dataBytes = Encoding.UTF8.GetBytes(data);
    clientSocket.Send(dataBytes);

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

(Client 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.

```
namespace Client
{
    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",
                "applu","apsi","hydro","go","su2cor","swin","tomcatv","cc1" };
            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;
            string benchmark = null;



        Int32 BufferReader = 131072;
        using (StreamReader st = new StreamReader(simulation, Encoding.UTF8,true,BufferReader))
        {
            string line;
            while ((line = st.ReadLine()) != null)
            {
                if (values.Any(line.Contains))
                {
                    var arguments = line.Split(new string[] { " " }, StringSplitOptions.RemoveEmptyEntries).Take(2).ToArray();
                    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;
```

(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.

```
private static string responseData;
private static Metrics metrics = new Metrics();
private static readonly string PATH_TO_SCRIPT = "../Debug/simplesim-3.0/script.sh";
public static void Main(string[] args)
{
    Console.WriteLine("Welcome to 'Defenetly not a virus team'.");
    Net();
    LaunchSim();
}
public static void GenerateScript()
{
    //string commnandBuffer = "./sim-outorder -redir:sim results/simulation.res -redir:prog results/applu_progout.res -max:inst 10000 -fastfwd 0 -fetch:i
    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 LaunchSim()
{
    GenerateScript();
    string readScript = File.ReadAllText(PATH_TO_SCRIPT);

    ProcessStartInfo startInfo = new ProcessStartInfo() { FileName = "/bin/bash", Arguments = "./script.sh", WorkingDirectory = "../Debug/simplesim-3.0"
    Process proc = new Process() { StartInfo = startInfo };
    proc.Start();


    metrics = metrics.parseString("../Debug/simplesim-3.0/results/simulation.res");
    Console.WriteLine("Everiting is done, go in peace my son.");
}
```

(Creare script.sh din aplicatia Client)

3. Resurse necesare

Resursele necesare sunt aceleași ca 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. Concluzi

Deși este o aplicație simplista prin care se poate folosi simulatorul sim-outorder, el prezinta 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.