Automatic Design Space Explorer (D3)

Simularea si optimizarea arhitecturilor de calcul

Radulescu Robert-Valentin (244/1)

Mihai Corneliu-Stefan (244/1)

Tema proiectului

Un Automatic Design Space Explorer (ASDE) este un sistem automatizat care poate explora spatiul de proiectare al unui sistem sau a unei aplicatii.

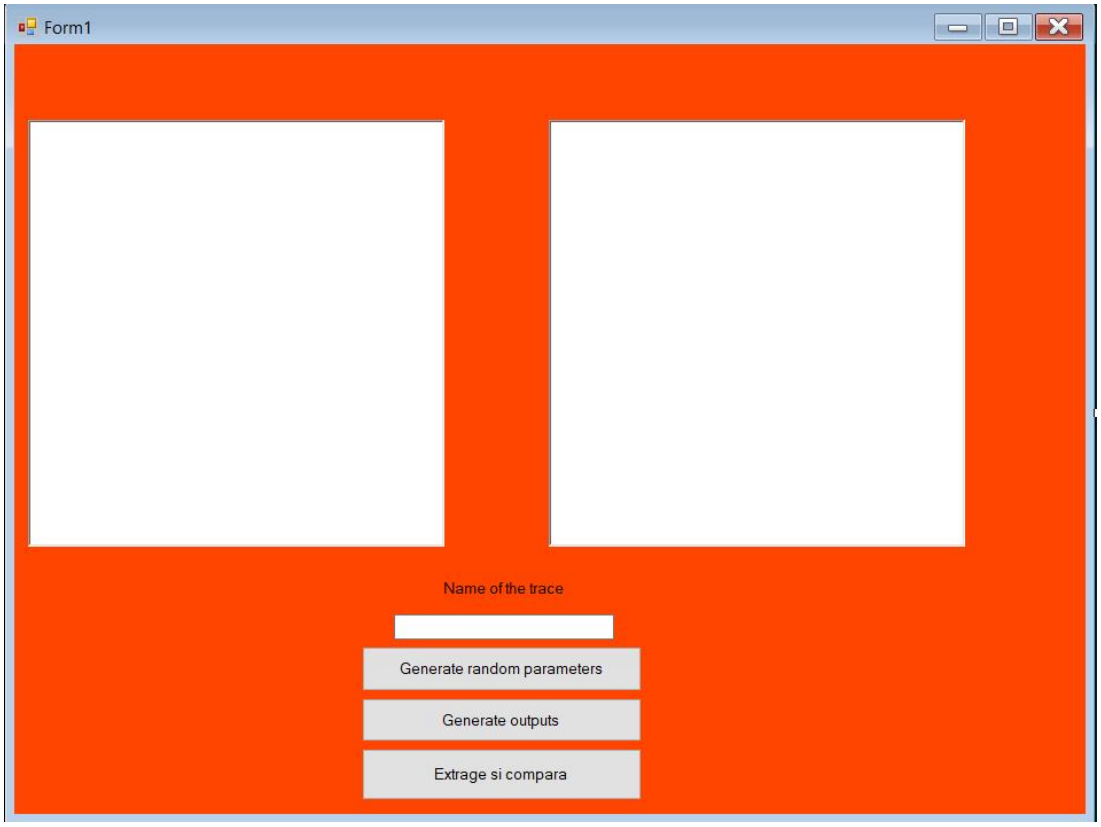
La acest proiect am avut de implementat un Automatic Design Space Explorer aferent unei arhitecturi superscalare pentru analiza performantei si a consumului de energie folosind o tehnica de optimizare non-Pareto.

Tehnica de optimizare non-Pareto se refera la abordarea de a optimiza mai multe obiective simultan, in care nici un obiectiv nu poate fi imbunatatit fara a a afecta negativ alte obiective. Aceasta se numeste non-Pareto pentru ca nu exista nicio solutie “Pareto-optimala”.

Pentru a implementa algoritmul de optimizare non-Pareto ne-am folosit de un prag de referinta pentru IPC si Power-ul nostru. Am extras valoarea maxima pentru IPC si Power, am stabilit un prag si am pastrat toate valorile atata timp cat acestea sunt mai mici decat valoarea pragului.

Descrierea implementarii

Interfata grafica a proiectului



In textbox-ul din stanga vom afisa toate valorile sortate crescator pentru IPC si Power. In cel din dreapta vom afisa toate valorile non-Pareto care se afla sub valorea pragului.

Sub label-ul “Name of the trace” avem un textbox in care utilizatorul poate declara numele trace-ului cu care doreste sa faca simularea.

Butonul “Generate random parameters” ne va genera 20 de configuratii cu valori aleatoare.

Butonul “Generate outputs” ne va genera 20 de output-uri pe baza configuratiilor in urma simularii cu simulatorul PSATSim_con.

Butonul “Extrage si compara” afiseaza pe textbox-uri valorile ordonate crescator pentru IPC si Power in textbox-ul din stanga si cele non-Pareto in cel din dreapta.

Structura codului

```
public class Config
{

    public List<string> configuration = new List<string>();

    public string trace = String.Empty;

    public string output = String.Empty;
    public int frequency;
    public double latency;
    public double codeHitrate;
    public double dataHitrate;
    public double speculation_accuracy;



    1 reference
    public Config(string trace, int frequency, double latency, double codeHitrate, double dataHitrate,double speculation_accuracy)
    {
        this.trace = trace;
        this.frequency = frequency;
        this.latency = latency;
        this.dataHitrate = dataHitrate;
        this.codeHitrate = codeHitrate;
        this.speculation_accuracy = speculation_accuracy;
    }
```

Figura 1. Clasa si constructorul.

In figura 1 am declarat clasa Config care contine toti membrii care vor fi generati aleator (respectiv frequency, latency, codeHitrate, dataHitrate si speculation_accuracy) si variabila output care va stoca cate o configuratie generata aleator pe rand.

Dupa ce am declarat variabilele clasei am initializat un constructor in care am dat ca parametru fiecare variabila ce va fi utilizata in generarea unui obiect cu valori aleatore.

```
private void button1_Click(object sender, EventArgs e)
{
    string citit = textBox1.Text;
    citeste();

    for (int i = 0; i < 20; i++)
    {
        Random rnd = new Random();
        int frec = rnd.Next(200, 900);
        double latency = rnd.Next(10, 30);
        double dataHitrate = rnd.NextDouble() * 0.09 + 0.9;
        double codeHitrate = rnd.NextDouble() * 0.09 + 0.9;
        double speculation_accuracy= rnd.NextDouble() * 0.490 + 0.500;


        Config obj = new Config(citit + ".tra", frec,latency,codeHitrate,dataHitrate,speculation_accuracy);


        obj.output = string.Format(text, obj.trace, obj.frequency,obj.latency,obj.codeHitrate,obj.dataHitrate,obj.speculation_accuracy);
      //  obj.configuration.Add(obj.output);
        obj.scrie(obj.output, i);

    }


}
```

Figura 2. Generarea configuratiilor.

In prima faza am apelat metoda citeste() pentru a citi fisierul prototip default_cfg.txt care va contine modelul de configurare pentru toate trace-urile noastre.

```
public string citeste()
{

    using (StreamReader reader = new StreamReader(@"C:\Users\Stefan\Desktop\ProiectD3-Final\ProiectD3-Final\default_cfg.txt"))
    {
        text = reader.ReadToEnd();
    }
    return text;

}
```

```
<psatsim>
    <config name="Default">

        <general
            superscalar="4" rename="16" reorder="20"
            rsb_architecture="distributed" rs_per_rsb="2"
            speculative="true" speculation_accuracy="{5}"
            separate_dispatch="true" seed="0"
            trace="{0}" output="output.xml"
            vdd="2.2" frequency="{1}" />

        <execution architecture="standard" integer="2" floating="2" branch="1" memory="1" />


        <memory architecture="l2">
            <l1_code hitrate="{3}" latency="1" />
            <l1_data hitrate="{4}" latency="1" />
            <l2 hitrate="0.990" latency="3" />
            <system latency="{2}" />
        </memory>
    </config>
</psatsim>
```

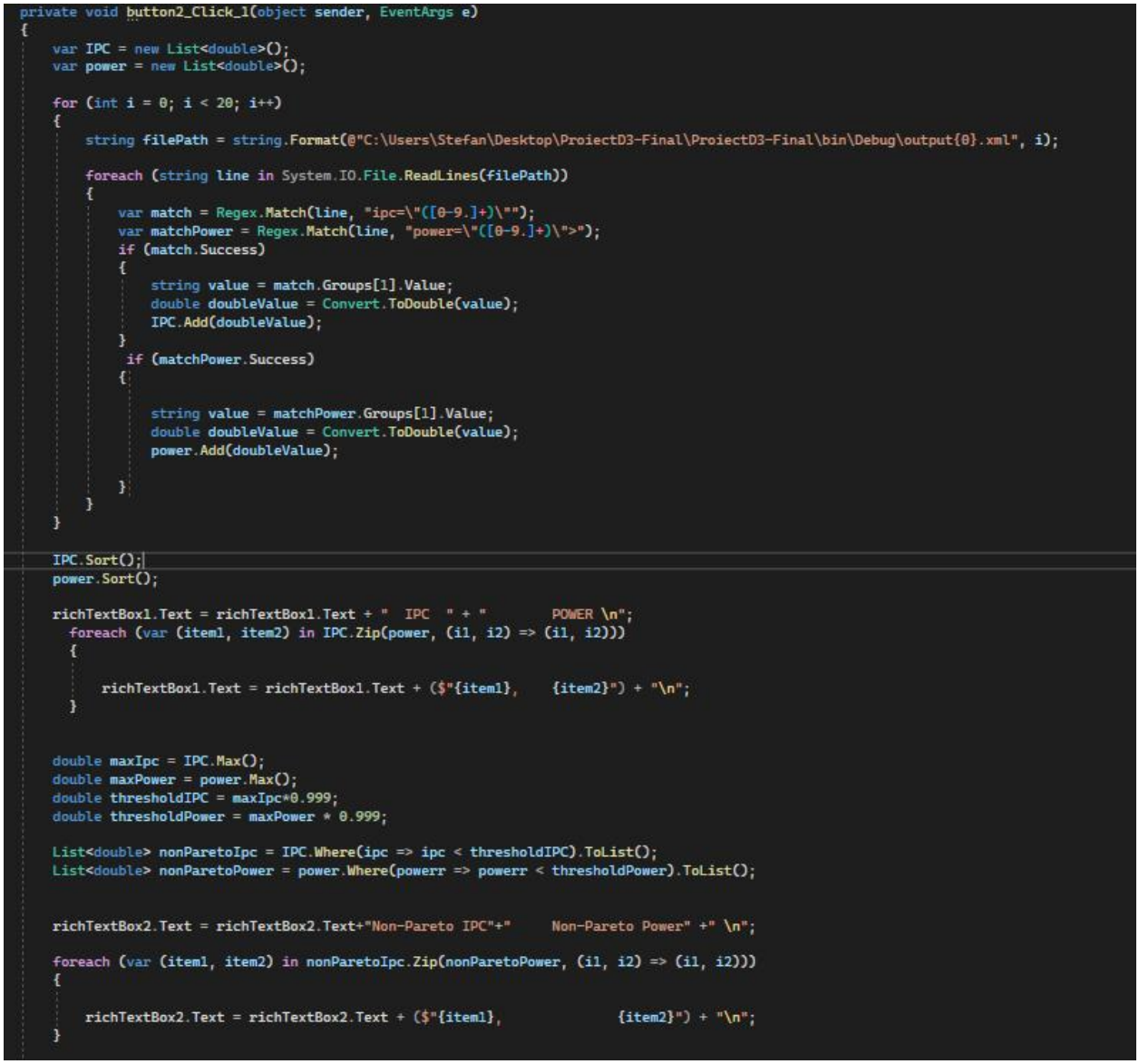
Dupa cum se poate observa in figura 2, am generat 20 de configuratii pentru membrii clasei Config. In membrul output al clasei Config am stocat configuratia aleatoare. De asemenea, in fisierul de configuratie default_cfg am implementat un placeholder pentru toate zonele care vor fi generate automat. Pentru a inlocui zonele unde am pus placeholdere, ne-am folosit de functia string.Format(). La finalul for-ului pentru fiecare configuratie, am apelatat functia scrie() care va scrie output-ul configuratiei la indexul dat de i.

```
public void scrie(string output, int index)
{
    string configuratie = String.Format("default_cfg{0}.xml", index);

    using (StreamWriter writer = new StreamWriter(configuratie))
    {
        writer.Write(output);
    }
}
```

```
private void button2_Click(object sender, EventArgs e)
{
    string copie;

    // System.Diagnostics.Process.Start("cmd.exe");
    for (int i = 0; i <= 20; i++)
    {
        System.Threading.Thread.Sleep(1000);
        Process p = new Process();
        ProcessStartInfo startInfo = new ProcessStartInfo();
        startInfo.FileName = "cmd.exe";
        copie = String.Format(@"/c psatsim_con.exe default_cfg{0}.xml output{0}.xml", i);
        startInfo.Arguments = copie;
        p.StartInfo = startInfo;
        p.Start();

    }

}
```

Figura 3. Generare outputs.

In aceasta functie generam output-urile pentru toate cele 20 de configuratii. Functia va deschide un cmd in care deschidem simulatorul psatsim_con.exe si parsam ca parametrii configuratia (input) si output-ul la care avem un placeholder pentru fiecare fisier.



```
private void button2_Click_1(object sender, EventArgs e)
{
    var IPC = new List<double>();
    var power = new List<double>();

    for (int i = 0; i < 20; i++)
    {
        string filePath = string.Format(@"C:\Users\Stefan\Desktop\ProiectD3-Final\ProiectD3-Final\bin\Debug\output{0}.xml", i);

        foreach (string line in System.IO.File.ReadLines(filePath))
        {
            var match = Regex.Match(line, "ipc=\"([0-9.]+)\"");
            var matchPower = Regex.Match(line, "power=\"([0-9.]+)\">");
            if (match.Success)
            {
                string value = match.Groups[1].Value;
                double doubleValue = Convert.ToDouble(value);
                IPC.Add(doubleValue);
            }
             if (matchPower.Success)
            {

                string value = matchPower.Groups[1].Value;
                double doubleValue = Convert.ToDouble(value);
                power.Add(doubleValue);

            }
        }
    }

    IPC.Sort();
    power.Sort();

    richTextBox1.Text = richTextBox1.Text + "  IPC  " + "        POWER \n";
      foreach (var (item1, item2) in IPC.Zip(power, (i1, i2) => (i1, i2)))
      {

          richTextBox1.Text = richTextBox1.Text + ($"{item1},    {item2}") + "\n";
      }


    double maxIpc = IPC.Max();
    double maxPower = power.Max();
    double thresholdIPC = maxIpc*0.999;
    double thresholdPower = maxPower * 0.999;

    List<double> nonParetoIpc = IPC.Where(ipc => ipc < thresholdIPC).ToList();
    List<double> nonParetoPower = power.Where(powerr => powerr < thresholdPower).ToList();


    richTextBox2.Text = richTextBox2.Text+"Non-Pareto IPC"+"     Non-Pareto Power" +" \n";

    foreach (var (item1, item2) in nonParetoIpc.Zip(nonParetoPower, (i1, i2) => (i1, i2)))
    {

        richTextBox2.Text = richTextBox2.Text + ($"{item1},              {item2}") + "\n";
    }
```

Figura 4. Extragerea, sortarea, afisarea si aplicarea algoritmului non-Pareto.

In prima faza, am declarat 2 liste in care o sa avem toate valorile pentru IPC si respectiv Power. In continuarea, parcurgem fiecare fisier xml de output si extragem valorile pentru IPC si Power, pe care le adaugam in liste. Dupa aceea, sortam crescator listele si afisam continutul lor in textbox-ul din partea stanga.

In continuare, incepem implementarea algoritmului non-Pareto. Punem valorile maxime gasite pentru IPC si Power in 2 variabile noi si declaram un prag (threshold) in care luam valoarea maxima pentru fiecare si inmultim cu o valoare de prag aleasa de noi. Dupa, declaram 2 liste nonParetoIPC si nonParetoPower in care vom stoca toate valorile pentru IPC si Power care sunt mai mici decat pragul nostru. In final, afisam valorile din listele non-Pareto in textbox-ul din dreapta.

Concluzii

Realizand acest proiect am invatat pasii pentru a implementa un Automatic Design Space Explorer si am incercat sa gasim cea mai buna simulare in raport cu valorile pentru IPC si Power, in proces familiarizandu-ne cu mai multi algoritmi non-Pareto.

Bibliografie

1. Cartea SOAC Matrix – Adrian Florea
2. https://en.wikipedia.org/wiki/Design_space_exploration