**KAUNO TECHNOLOGIJOS UNIVERSITETAS**

**INFORMATIKOS FAKULTETAS**

Lygiagretus programavimas (P170B328)

Individualus darbas

Atliko:

IFF-4/3 gr. studentas

Ovidijus Stukas

Priėmė:

Lekt. Barisas Dominykas

Doc. Marcinkevičius Romas

KAUNAS, 2016

Turinys

[1. Užduotis 3](#_Toc469346028)

[2. Užduoties analizė. 3](#_Toc469346029)

[3. Programos aprašymas 3](#_Toc469346030)

[4. Programos pagrindinės dalys 3](#_Toc469346031)

[5. Testavimas ir programos instaliavimo bei vykdymo instrukciją 4](#_Toc469346032)

[6. Vykdymo laiko kitimo tyrimas 5](#_Toc469346033)

[7. Išvados 6](#_Toc469346034)

[8. Literatūra: 6](#_Toc469346035)

# Užduotis

Matricų daugybos algoritmo įgyvendinimas ir tyrimas naudojant N paralelinių procesų kiekį ir Java JCSP biblioteką.

# Užduoties analizė.

Matricų daugybos algoritmas naudoją kairės pusės matricos eilutę „e“ ir dešinės pusės stulpelį „s“ kiekviena pora ei ir si yra sudauginama ir sumuojama taip apskaičiuojama rezultatų matricos [i ; j] pozicijos reikšmę.



# Programos aprašymas

Pagrindinė klasė „StukasO“ sugeneruoja atsitiktinių reikšmių nurodyto dydžio 2 matricas, kurias naudos skaičiavimams. Taip pat ši klasė nusprendžia ir naudotojui leidžia pasirinkti kiek JSCP procesų paleisti skaičiavimams. „Manager“ klasės procesas turi any2One JSCP kanalą iš kurio skaito suskaičiuotas matricos dalis ir rezultatus deda į galutinę matricą. „Worker“ klasės procesas turi kanalą į „Manager“ suskaičiavęs vienos pozicijos reikšmę iš duotų kairės pusės matricos eilučių ir dešinės pusės stulpelių.

# Programos pagrindinės dalys

Pagrindinis procesas „Worker“:

static class Worker implements CSProcess {  
 private final ChannelOutput<MatrixPart> channelOutput;  
 private final double[][] lhsRows;  
 private final double[][] rhsCols;  
 private final int indexAddition;  
  
 Worker(ChannelOutput<MatrixPart> channelOutput, double[][] lhsRows, double[][] rhsCols, int indexAddition) {  
 this.channelOutput = channelOutput;  
 this.indexAddition = indexAddition;  
 this.lhsRows = lhsRows;  
 this.rhsCols = rhsCols;  
 }  
  
 @Override  
 public void run() {  
 for (int i = 0; i < lhsRows.length; i++) {  
 double[] row = lhsRows[i];  
 for (int j = 0; j < rhsCols.length; j++) {  
 double[] col = rhsCols[j];  
  
 double sum = 0.0;  
 for (int k = 0; k < row.length; k++)  
 sum+=row[k] \* col[k];  
  
 MatrixPart matrixPart = new MatrixPart(i + indexAddition, j, sum);  
 channelOutput.write(matrixPart);  
 }  
 }  
  
 channelOutput.write(null);  
 }  
}

Kontroliuojantis procesas „Manager“

static class Manager implements CSProcess {  
 private final AltingChannelInput<MatrixPart> workerChannelInput;  
 private final Matrix matrix;  
 private int workerFinishedCount;  
  
 Manager(AltingChannelInput<MatrixPart> workerChannelInput) {  
 this.workerChannelInput = workerChannelInput;  
 this.matrix = new Matrix(*MATRIX\_SIZE*, *MATRIX\_SIZE*);  
 this.workerFinishedCount = 0;  
 }  
  
 @Override  
 public void run() {  
 final Guard[] guards = { workerChannelInput };  
 final Alternative alternative = new Alternative(guards);  
  
 while (workerFinishedCount != *WORKER\_THREAD\_COUNT*) {  
 switch (alternative.fairSelect()) {  
 case 0:  
 handleWorker();  
 break;  
 }  
 }  
  
 long endTime = System.*nanoTime*();  
 System.*out*.println("Uztruko: " + ((endTime - *TIMER\_START*) / 1000000.0) + " ms");  
 }  
  
 private void handleWorker() {  
 MatrixPart matrixPart = workerChannelInput.read();  
 if (matrixPart != null) {  
 matrix.values[matrixPart.storeM][matrixPart.storeN] = matrixPart.result;  
 } else {  
 workerFinishedCount++;  
 }  
 }  
}

# Testavimas ir programos instaliavimo bei vykdymo instrukciją

Norint pasileisti programą reikalinga JCSP 1.1- RC5 biblioteka. Pagrindine klasė sukompiliuojama ir paleidžiama. Norint keisti matricos dydi reikia programoje pakeisti „MATRIX\_SIZE“ dydį ir jei norint keisti skaičiuojančių procesų kiekį, reikia pakeisti kintamojo „WORKER\_THREAD\_COUNT“ kiekį, tačiau matricos dydis turi dalintis be liekanos iš šio kintamojo.

# Vykdymo laiko kitimo tyrimas

Bandymas su 10x10 matrica:

|  |  |
| --- | --- |
| Procesų skaičius | T (5 bandymų vidurkis) **ms** |
| 1 | 4,01 |
| 2 | 4,45 |
| 5 | 3,66 |
| 10 | 6,42 |

Bandymas su 100x100 matrica:

|  |  |
| --- | --- |
| Procesų skaičius | T (5 bandymų vidurkis) **ms** |
| 1 | 97,40 |
| 2 | 109,49 |
| 5 | 95,23 |
| 10 | 129,49 |

Bandymas su 300x300 matrica:

|  |  |
| --- | --- |
| Procesų skaičius | T (5 bandymų vidurkis) **ms** |
| 1 | 744,53 |
| 2 | 823,46 |
| 5 | 851,39 |
| 10 | 853,44 |

# Išvados

Įgyvendinus matricų daugyba su JCSP biblioteka, nebuvo pasiekta daug papildomos galios. Matricų daugybą parealizuojant su 5 procesais buvo pasiekta iki 10-15% laiko sumažinimo.

# Literatūra:

<https://en.wikipedia.org/wiki/Matrix_multiplication>

<http://matrix.reshish.com/multiplication.php>