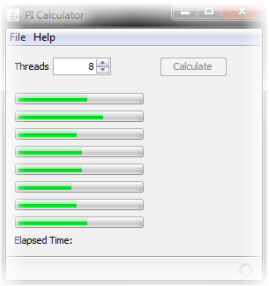
PiCalculator

Програмиране на разпределени, паралелни и системни приложения

Николай Архангелов Архангелов 61187

Йордан Пенев Илиев 61188



доц. **В. Георгиев**

гл. ас. **Г. Георгиев**

ас. **Х. Христов**

17.06.2010г

Задача

Задача

Да се състави програма, която пресмята 1000 цифри на числото π чрез използването на многонишков подход.

Цел

Целта на програмата е да демонстрира разликата във времената за изпълнение при използването на различен брой нишки (самите цифри на π не ни интересуват).

Реализация

За да се пресметне n-тата цифра на числото π ще се използва формулата на Bailey–Borwein–Plouffe.

Нека с π_n означим n-тата цифра на π.

Тогава $\pi_n = \sum_{k=0}^{n} \frac{16^{n-k} \, mod \, 8k+1}{8k+1} + \sum_{k=n+1}^{\infty} \frac{16^{n-k}}{8k+1}$.

Реализациа на алгоритъма на Java:

```
private void Calculate(int n)
{
    //Bailey–Borwein–Plouffe formula
    double sum = 0;

    for (int k = 0; k < n; k++)
    {
        sum += ((Math.pow(16, n - k) % (8 * k + 1)) / (8 * k + 1));
    }

    for (int k = n + 1; k <= 50000; k++)
    {
        sum += Math.pow(16, n - k) / (8 * k + 1);
    }

    sum /= Math.pow(16, n);
}
```

Реализация

В програмата има два основни класа: ThreadManager и CalculatorThread

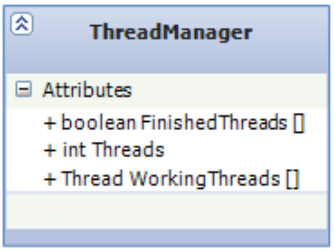
ThreadManager

```
class ThreadManager implements Runnable
```

Това е клас който имплементира интерфейса Runnable. Целта на този клас е да менажира нишките, които изчисляват цифрите на π.

ThreadManager се стартира в самостоятелна нишка.

В реализацията на PiCalculator с потребителски интерфейс, ThreadManager класа следи за прогреса на всяка CalculatorThread нишка и обновява интерфейса.



Основни полета:

- FinishtedThreads (UI реализация) – масив в, който се записва кои изчислителни нишки са приключили работа. Когато всички приключат се спира таймера и потребителя отново може да стартира изчисленията;
- Threads – брой на изчислителните нишки. Задава се от потребителя;
- WorkingThreads – масив съдържащ с инстанции на изчислителните нишки.

Стартиране на изчислителните нишки:

```
private void RunThreads()
{
    for (int threadIndex = 0; threadIndex < _threads; threadIndex++)
    {
        _workingThreads[threadIndex] = new Thread
                (new CalculatorThread(threadIndex, _threads, _digits));
        _workingThreads[threadIndex].start();
    }
}
```

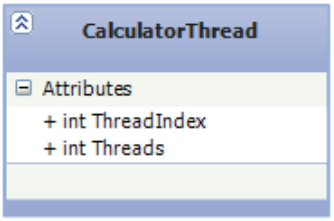
Реализация

CalculatorThread

```
class CalculatorThread implements Runnable
```

Това е клас който имплементира интерфейса Runnable. Целта на този клас е да изчисли цифрите на π според индекса на инстанцията.

Класът ThreadManager разполага с масив, който съдържа инстанции на CalculatorThread класа. Всяка инстанция на CalculatorThread работи в отделна нишка.



Основни полета:

- ThreadIndex – показва индекса на нишката в масива с инстанции WorkingThreads;
- Threads – брой на изчислителните нишки. Задава се от потребителя.

Разпределяне на изчисленията:

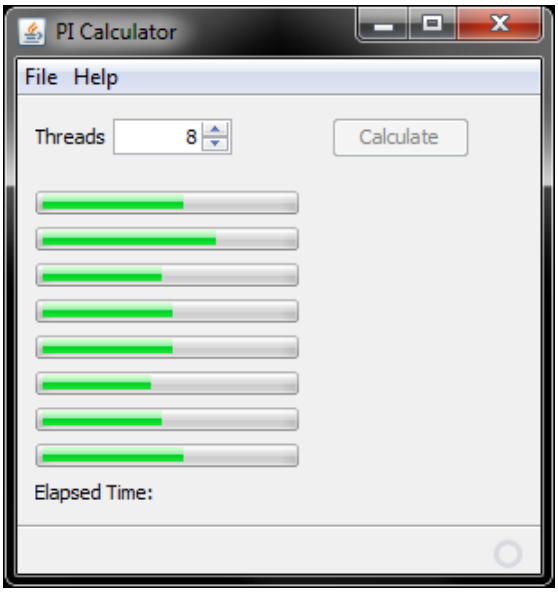
```
private void ManageCalculations()
{
    for (int i = _threadIndex; i < _digits; i += _threads)
    {
        Calculate(i);

        double progress = (((double) i / _digits)) * 100;
        final int finalProgress = (int) Math.ceil(progress);
        //Report progress
    }
}
```

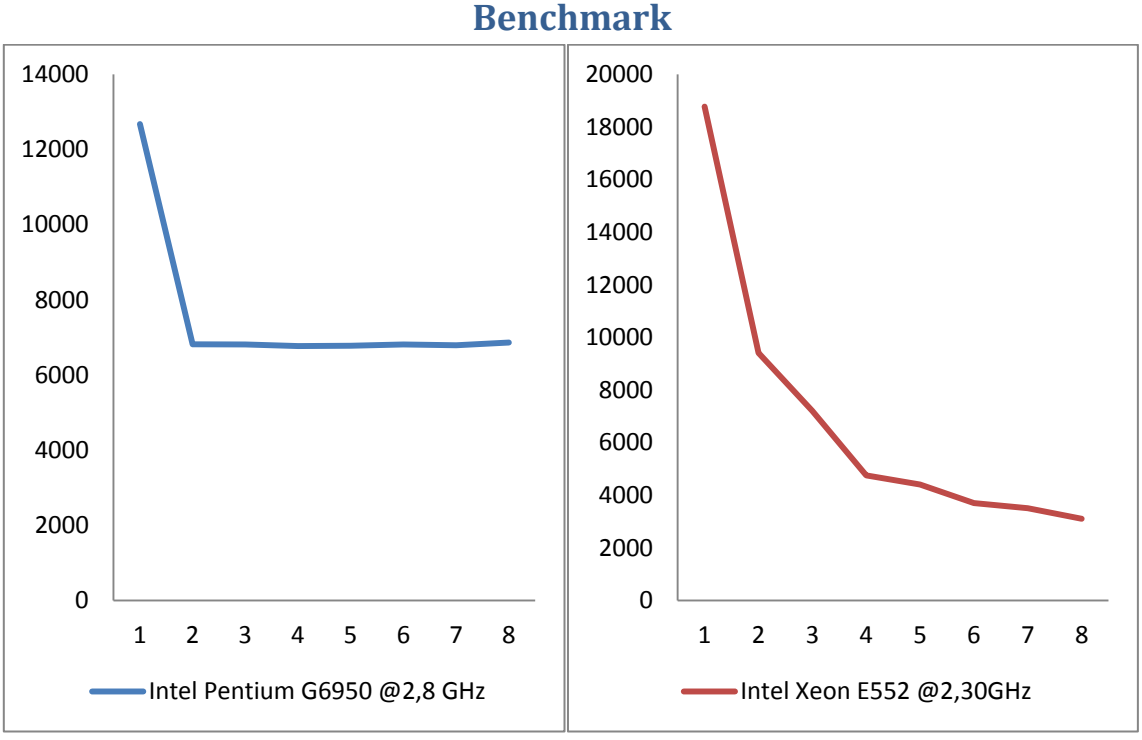
Всяка нишка смята π_n, където $n = ThreadIndex + i * Threads$, за $\forall i: 0 \leq ThreadIndex + i * Threads \leq 1000, i \in N$.

Изчисляването на всяка цифра става по описата формула в началото.

Реализация с UI

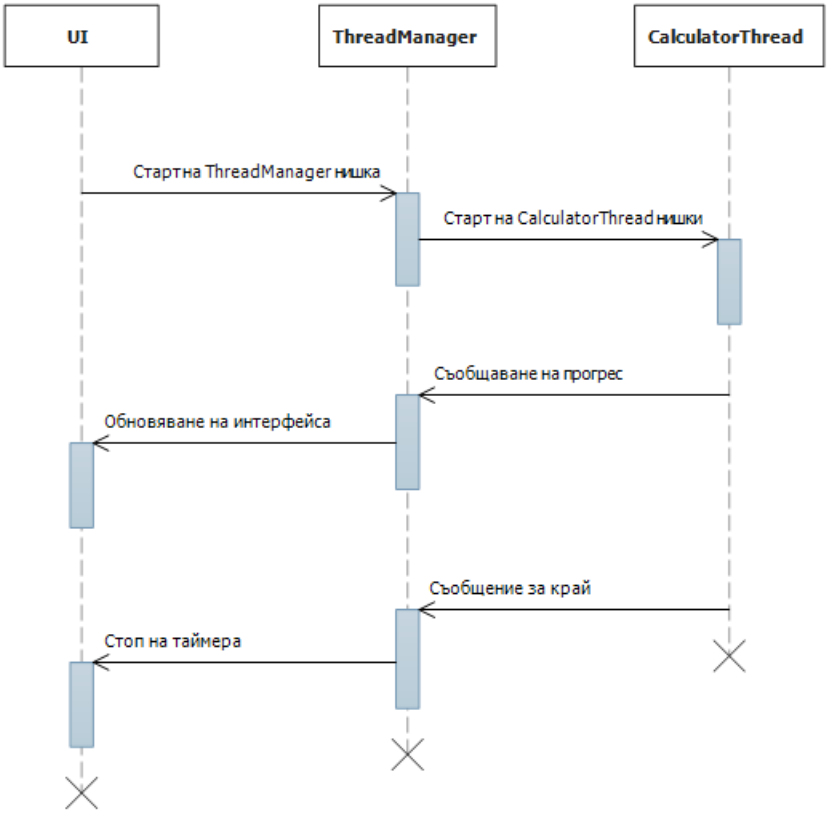


При стартиране на програмата, потребителят може да избере броя на работещите CalculatorThread инстанции чрез Spinner контролата. Изчисленията се стартират с бутона Calculate. По време на изчисленията бутона Calculate преминава в състояние Disabled. Прогреса се наблюдава чрез ProgressBar-овете. След приключване на изчисленията бутона Caclulate минава в състояние Enabled и Label-а Elapsed Time изписва времето за изчисление в милисекунди.

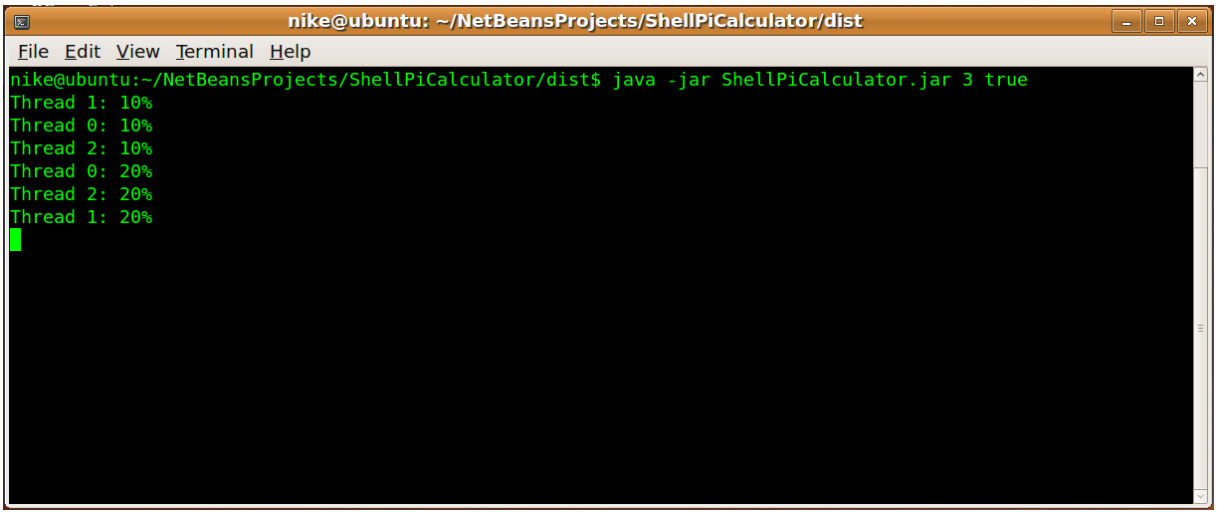


Реализация с UI

UI Sequence Diagram



Конзолна реализация



В конзолен режим програмата се стартира с командата

java –jar ShellPiCalculator <брой нишки**> [**фонов режим**]**