Софийски Университет „Св. Климент Охридски”



Факултет по математика и информатика

**Проект**

по Разпределени софтуерни архитектури

# Пресмятане на фрактал

**F(Z) = C \* COS(Z)**

Христо Станиславов Христов, СИ 3 курс, група 5

Научен ръководител: ас. Христо Христов

Дата: 25.06.2015

Проверка: \_\_\_\_\_\_\_\_\_\_\_\_

1. **Цел на проекта**

Създаване на програма, която да пресмята и изгражда изображение на фрактала **F(Z) = C \* COS(Z)**. Тя трябва да разбира и от следните команди:

* Програмата да позволява (разбира от) команден параметър, който задава големината на генерираното, широчина и височина в брой пиксели. Той има вида: „-s“ (или „-size“ ); При не-въведен от потребителя команден параметър, за големина на изображението, програмата подразбира - широчина (width) 640px и височина (height) 480px
* Команден параметър, който да задава частта от комплексната равнина, в която ще търсим визуализация на множеството на Манделброт: „-r -2.0:2.0:-1.0:1.0“ (или„-rect“). Стойността напараметъра се интерпретира както следва: а∈ [−2.0, 2.0], b∈ [−1.0, 1.0]. При не въведен от потребителя параметър програмата приема че е зададена по подразбиране: „-2.0:2.0:-2.0:2.0“
* Друг команден параметър, който задава максималния брой нишки (паралелни процеси) на които разделяме работата по генерирането на изображението: „-t 3“ (или „-tasks“); При не-въведен от потребителя команден за брой нишки – програмата подразбира 1 нишка
* Команден параметър указващ името на генерираното изображение: „-o zad18.png“ (или „-output“). Съответно програмата записва генерираното изображение в този файл. Ако този параметър е изпуснат (не е зададен от потребителя), се избира име по подразбиране: **„zad18.png“**
* Програмата извежда подходящи съобщения на различните етапи от работата си, както и времето отделено за завършване на всички изчисления по визуализирането на точките от множеството на Манделброт (пресмятане на множеството на Манделброт)
* Да се осигури възможност за „quiet“ режим на работа на програмата, при който се извежда само времето през което програмата е работила (без „подходящите“ съобщения от предходната точка). Параметърът за тази цел нека да е „-q“ (или „-quiet“); Тихият режим не отменя записванетона изображението във изходният файл;

1. **Описание на алгоритъма**

Алгоритъмът е паралелизиран по следния начин: създава се списък от четворки координати, чрез които се ограждат правоъгълници в дадена част на изображението. Тези правоъгълници не се застъпват и заедно покриват цялото изображение. Активира се необходимият брой нишки, като им се дава синхронизиран достъп до списъка, при което те сами си изтеглят сегменти, които да смятат. Накрая изображението се генерира серийно, поради използваната библиотека.

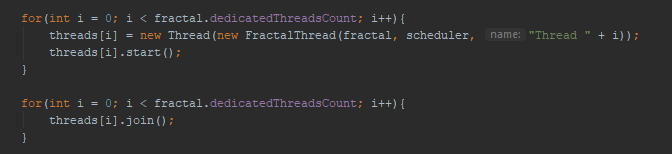
1. **Реализация**

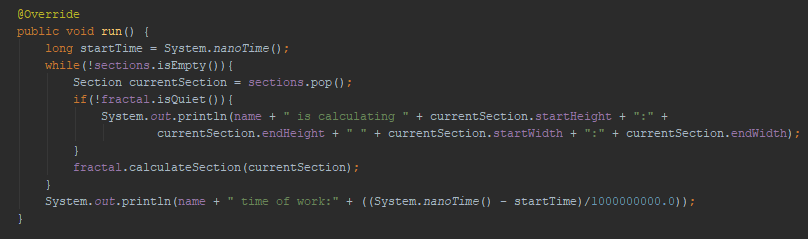
Алгоритъмът е реализиран на Java JDK 8.

В началото на програмата се генерира обект, който представлява фрактала, и носи всичката информация за него- размер на изображението, отместване от центъра, брой нишки, с които разполага, име на генерираното изображение и т.н. чрез подаване на args[] директно на конструктора му. След прочитане на args[], се генерират и помощни променливи, които се изчисляват на базата на новополучените.

След това се генерира списък от всички секции, на които е разделено изображението. Към момента повечето секции са квадрати със страни по 150 пиксела. Този списък се обвива в Scheduler, който синхронизира достъпа на нишките до него.



Влиза се в цикъл, който генерира максималния брой нишки, запазва ги в масив и ги „включва“ чрез start(), след което основната нишка стига до цикъл, в който се join()-ва подред към всички нишки( т.е. първо се join-ва към първата, при което спира. Ако тя се освободи, основната нишка се join-ва към следващата и т.н.). Така се гарантира, че основната нишка ще изчака и другите да приключат работата си. 

Това, което новите нишки правят, е следното: Всяка една от тях проверява дали списъка е празен, след което( ако не е празен) взимат сегмент от него и съобщават за това в конзолата( освен ако програмата не работи под “quiet” режим). След като вземе секцията, нишката преминава през функцията за смятане, в края на която се пише върху изображението. След изпълняването ѝ, нишката отново проверява дали има свободна секция и цикълът се повтаря.

След като всички нишки приключат работата си, основната нишка се освобождава и записва изображението в паметта.

1. **Стартиране на програмата**

При стартиране на програмата, се генерира фрактал със стойности по подразбиране. Ако в списъка с аргументи има аргумент, който променя някоя от тези стойности, то тя я презаписва.

Възможни команди:

* -t <число> -> задава броя нишки, които са достъпни за програмата
* -q -> указва на програмата да извежда съкратено множество от съобщение
* -s <ширина**Х**височина> -> определя резолюцията на изображението
* -r <число:по- голямо число, число:по-голямо число> -> определя отместването на дисплея
* -o <име.png> -> определя името на генерираното изображение

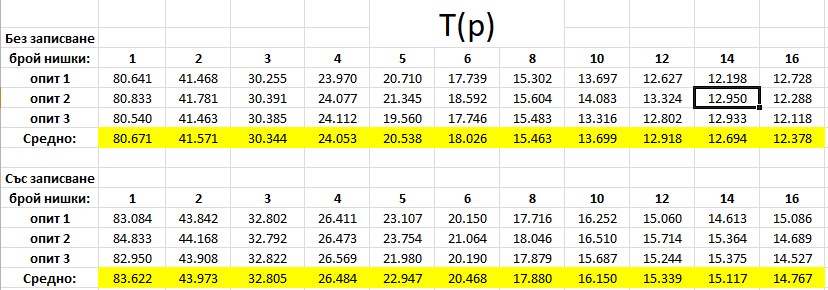
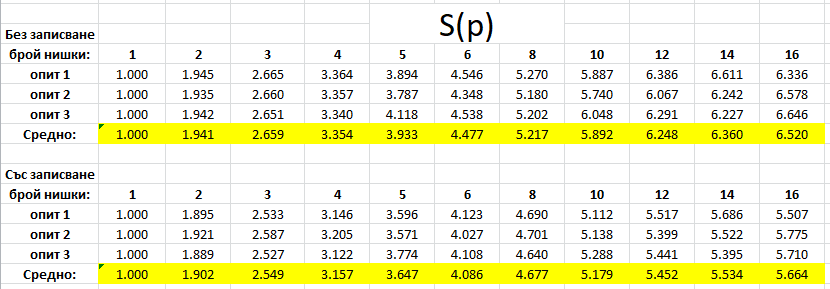
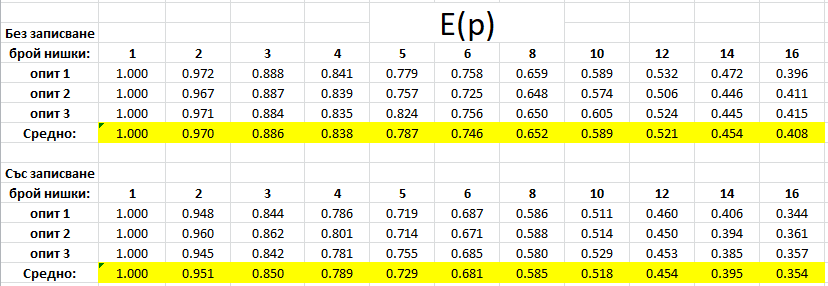
1. **Резултати**

Всичките тестови случаи са били направени с резолюция 4096**Х**4096. Останалите свойства на фрактала са тези по подразбиране.

Понеже програмата има както паралелна( сметките), така и серийна част( принтирането на изображението), ще представя два резултата- един показващ само ефекта от ускорението в паралелната част, без да се взима предвид принтирането или инициализацията на алгоритъма, и един, в който се измерва времето от началото до края на изпълнението на програмата.

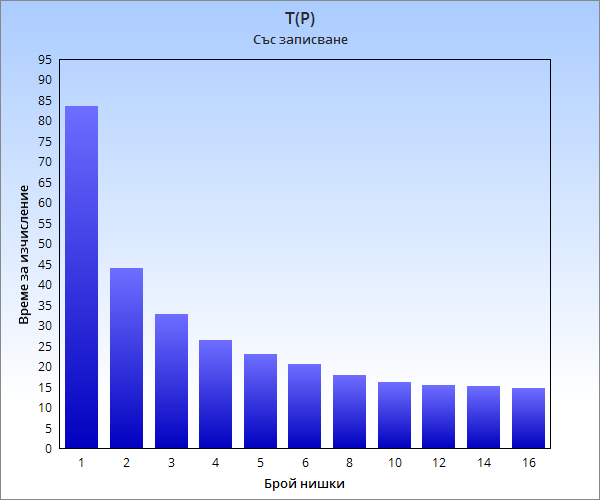
Алгоритъмът спира да се ускорява горе- долу при достигането на 16 едновременно изпълнявани нишки и за това те не са показани в статистиката. Друг фактор за безсмислието на тестовете с повече нишки е, че performance-a на сървъра се променя постоянно, което заличава и без това малките ускорения.

Таблици:

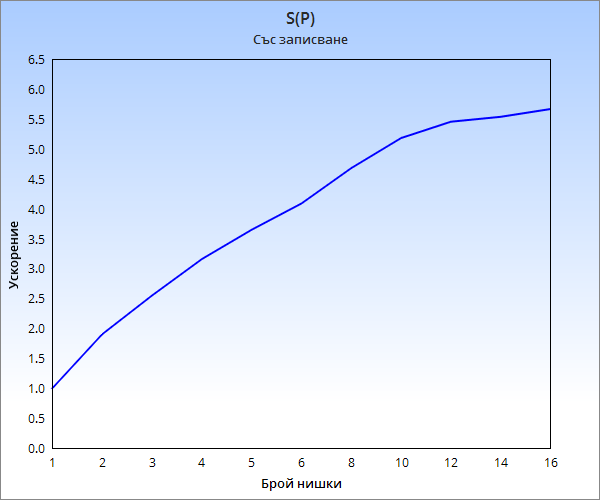
* Време на изпълнение( в секунди)
* Ускорение
* Ефективност

Графики на резултатите от паралелизацията със записване на изображението( няма смисъл да се покаже и без записване, идеята е същата):

* Време за изчисление



* Ускорение



* Ефикасност

