Софийски университет „Св. Климент Охридски”

Факултет по математика и информатика

Курсов проект

по Системи за паралелна обработка

Тема: „Изобразяване на фрактал – Mandelbrot set”

Изготвил: Елена Валентинова Георгиева, ФН: 81555, Компютърни науки, курс 3, поток 1, група 1

Ръководител: проф. Васил Георгиев, ас. Христо Христов

Дата: 03.05.2019 Проверка: ……..

1. Увод – цел на проекта, алгоритъм и функционалност

Проектът реализира паралелен алгоритъм, генериращ множеството на Манделброт, дефинирано с формулата:

(1) $$F(Z) = C*e^{-Z} + Z^2$$

в равнината на комплексните числа, тоест множеството от точки C в комплексната равнина, за които редицата $\{Z_0, Z_1, Z_2, ..., Z_n, Z_{n+1}, ...\}$, построена посредством горната формула, е ограничена, като итерирането започва от $Z_0 = 0$. Това условие може да се изрази и по следния начин [1]:

C ∈ M, дефинирано с формула (1) $\Leftrightarrow \lim_{n\to\infty} \sup |Zn| \leq 2$

С други думи, открием ли член на редицата, който надвишава 2 по абсолютна стойност, знаем, че тя клони към безкрайност, т.е е неограничена и съответното комплексно число C не принадлежи на множеството на Манделброт. Този факт е в основата на Escape time алгоритъма [2], който програмата използва, за да генерира множеството: всеки пиксел (x,y), x ∈ [0,width], y ∈ [0,height] , се изобразява в точка C от комплексната равнина с координати (c_re,c_im), след което с тази точка C програмата итерира покоординатно формула (1) , докато или поредното Z_n надхвърли 2 по абсолютна стойност, с което C излиза от множеството на Манделброт (escape condition), или се надхвърли определен максимален брой итерации, зададен предварително, без C да напусне търсеното множество, т.е в този случай C принадлежи на множеството на Манделброт. На този принцип всички пиксели от растера с размери width и height биват отбелязани като принадлежащи или непринадлежащи на множеството на Манделброт, като им се асоциира и съответен цвят (черно за точките от множеството, друг цвят за тези извън него).

Програмата удовлетворява следните функционални изисквания:

- Позволява (разбира от) команден параметър, който задава големината на генерираното изображение, като широчина и височина в брой пиксели. Той има вида**:„-s 640x480“**(или**„-size“**); При невъведен 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 на параметъра се интерпретира както следва:a∈[−2.0,2.0],b∈[−1.0,1.0]. При

невъведен от потребителя параметър програмата приема, че е зададена стойност по подразбиране:

„-2.0:2.0:-2.0:2.0“.

- Команден параметър, указващ грануларността за текущото изпълнение: **„ –g 4”** или **„-gran”**; При невъведен параметър програмата подразбира 1;

- Друг команден параметър, който задава максималния брой нишки (паралелни процеси), на които разделяме работата по генерирането на изображението: **„-t 3“** (или **„-tasks“**); При невъведен от потребителя команден параметър за брой нишки – програмата подразбира **1** нишка;

- Команден параметър, указващ името на генерираното изображение:**„-o zad16.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: **„zad16.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)

2. Проектиране и реализация

Алгоритъмът е имплементиран на Java 11. Реализиран е паралелизъм по данни (Single Program-Multiple Data) или SPMD, като нишките работят асинхронно (независими са една от друга). Архитектурата на програмата е по модела Master-Slaves. Класът TaskRunner.java влиза в ролята на Master. Програмата се стартира, като той бива извикан от командния ред и на него се подават всевъзможните параметри от командния ред. Пример за стартиране:

java TaskRunner –g 8 –t 32

ще стартира програмата с коефициент на грануларност 8 и 32 нишки и всички останали параметри с взети стойности по подразбиране. Класът TaskRunner.java създава обект от тип BufferedImage с размерите, подадени от потребителя в командния ред, и има грижата да стартира зададения от потребителя брой нишки/slaves (инстанции на класа MandelbrotRunnable.java, имплементиращ интерфейса Runnable), подавайки им в конструктора цялата необходима информация като размер на изображението, отрязък от комплексната равнина, който ще бъде разглеждан, номер на нишката, общ брой нишки,

референция към BufferedImage обекта и размер на частите, на които ще бъде разделено изображението при това извикване на програмата (chunk_size).

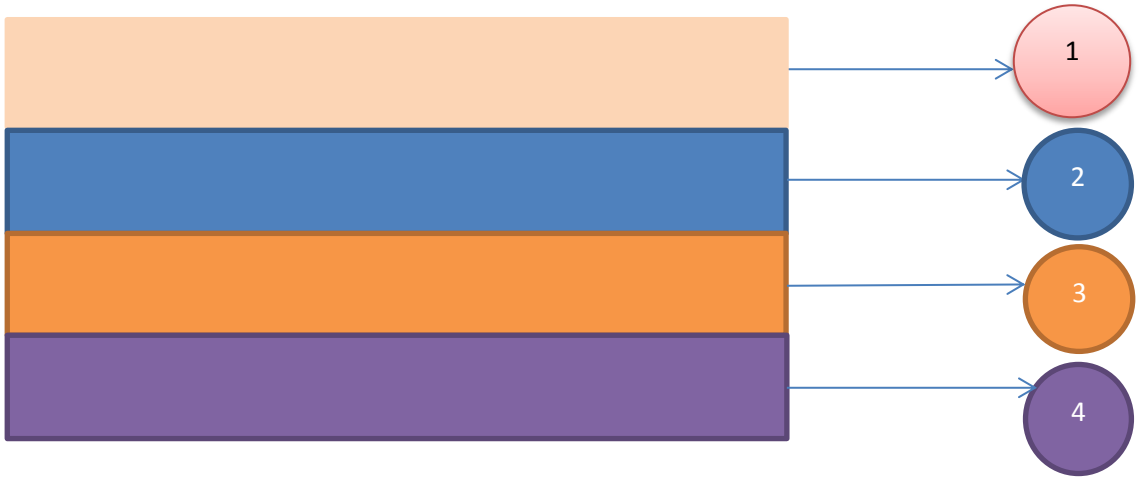
При всяко извикване на програмата се определят брой части/ ивици/ подзадачи (num_chunks) с еднакъв размер (chunk_size), на които ще бъде разделено изображението. Това разделяне става посредством броя нишки threads и коефициента на грануларност gran, подадени в командния ред, по формулата:

num_chunks = gran * threads

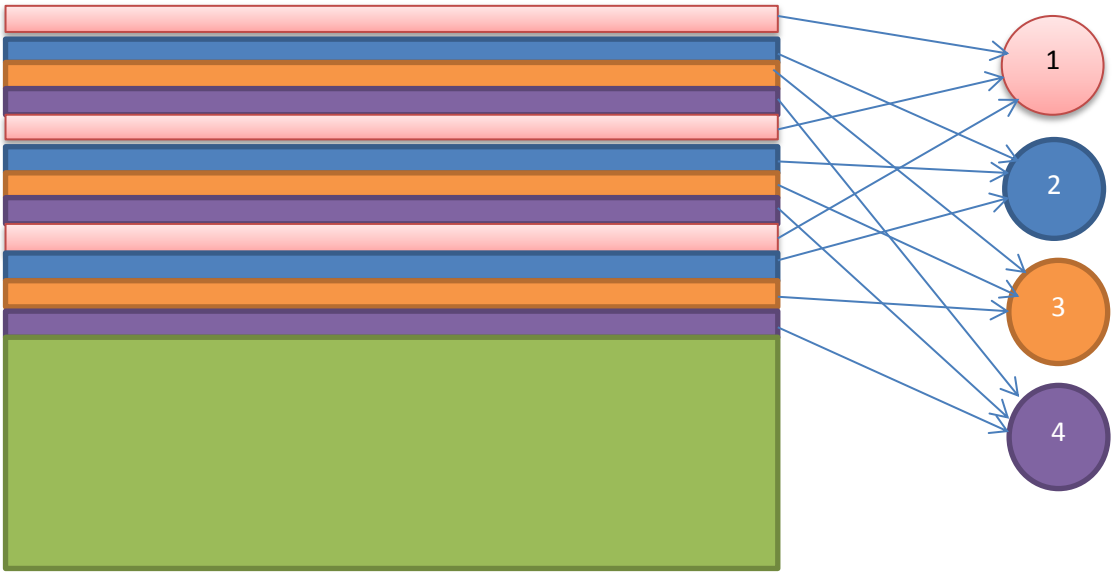
Ясно е тогава, че при gran = 1 имаме най-едрата възможна грануларност (толкова подзадачи, колкото са нишките) и все по-фина с нарастващ коефициент на грануларност.

Разпределянето на подзадачите между процесите (mapping) се извършва циклично в зависимост от номера на нишката и номера на подзадачата, по-точно всяка нишка при стартирането си разглежда всички номера на подзадачи от 1 до num_chunks и обработва само тези подзадачи, чиито номера имат остатък по модул броя нишки, равен на нейния номер. Обработката се състои в това да провери за всеки пиксел от съответната ивица дали принадлежи на множеството на Манделброт, или не, според escape time алгоритъма, описан в точка 1. С помощта на цикличното разпределение и грануларността може да се регулира баланса между натовареността на нишките (load balancing), а оттам и ускорението на паралелната програма.

Примерна схема на разпределението (едра грануларност): thread



Примерна схема на разпределението (фина грануларност): thread



Класът TaskRunner.java има грижата да изчака завършването и на последната нишка и да запише генерираното изображение във файл с името, подадено от командния ред.

3. Тестване

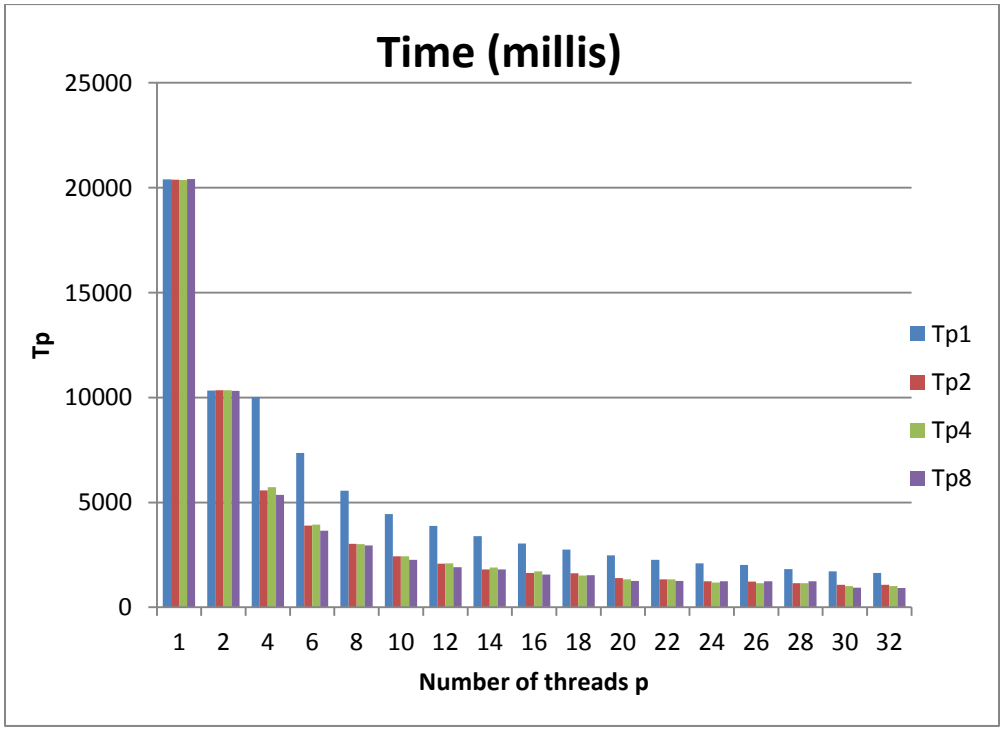
Програмата е тествана на тестовия сървър t5600.rmi.yaht.net. Изследвано е поведението ѝ при коефициенти на грануларност 1, 2, 4, 8 („-g 1", „-g 2", „-g 4", „-g 8") с брой нишки от 1,.....,32 , при стойности по подразбиране за останалите параметри от командния ред. Например при извикване

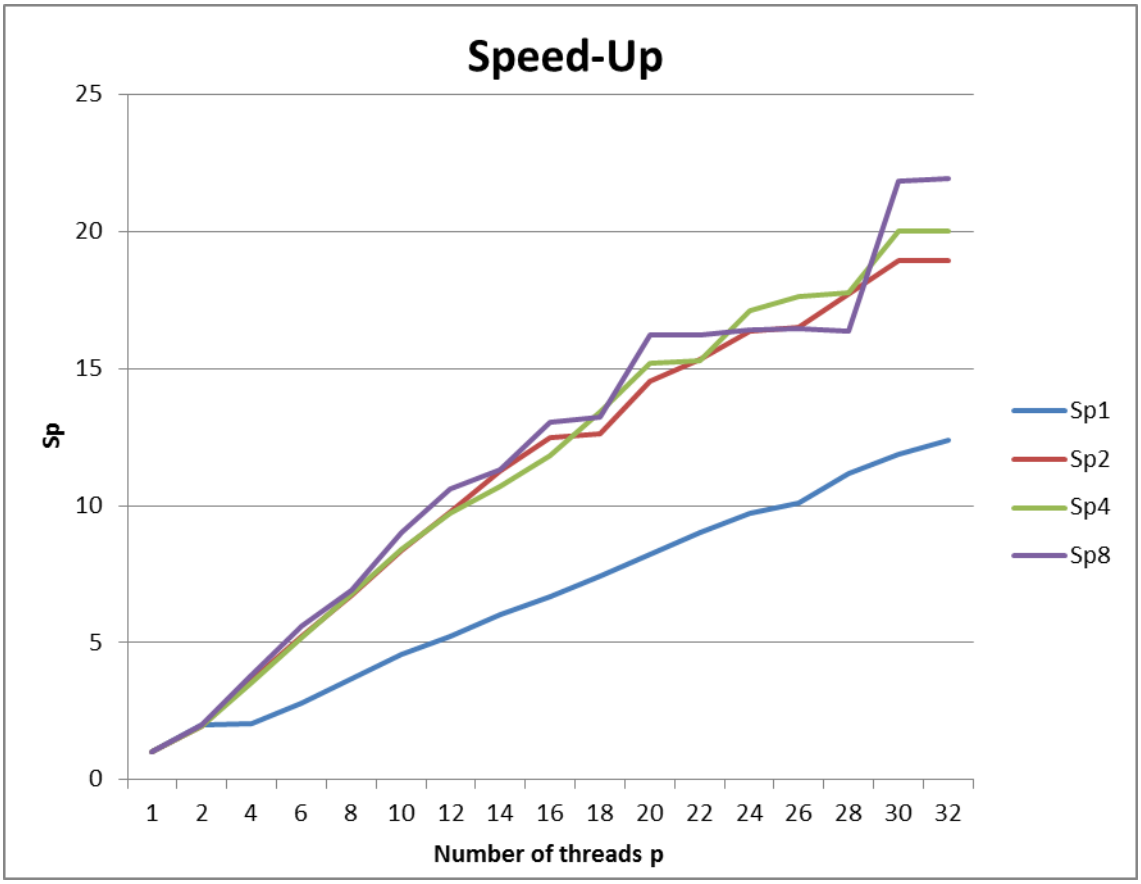
java TaskRunner -g 8 -t 32

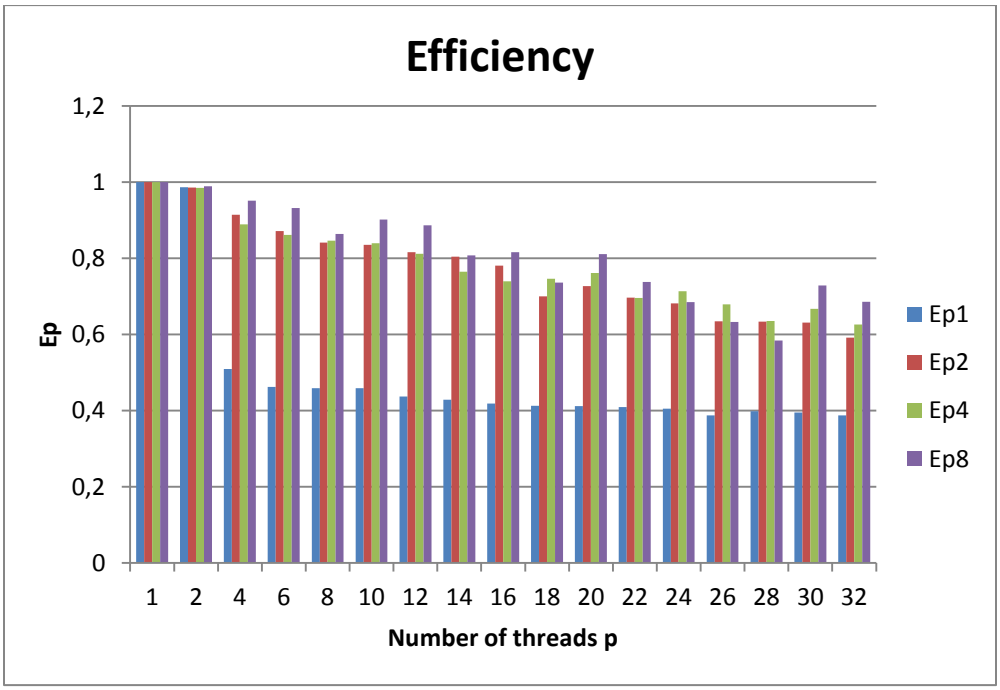
програмата работи с коефициент на грануларност 8 и 32 нишки, тоест изображението се разделя на 8 * броя нишки подзадачи с размер на подзадачата ≈ 2 реда пиксели.

Всяка таблица и всяка диаграма съдържа тестовите резултати и за четирите изследвани грануларности, като Tp1 са резултатите за времето за изпълнение при коефициент на грануларност 1, Tp2 – при грануларност 2, Tp4 и Tp8 – съответно при грануларност 4 и 8 . Аналогично е значението на Sp1, Sp2, Sp4, Sp8 за ускорението и Ep1, Ep2, Ep4, Ep8 за ефективността на програмата при различните грануларности.

p	Tp1	Tp2	Tp4	Tp8	Sp1	Sp2	Sp4	Sp8
1	20400	20381	20364	20410	1	1	1	1
2	10340	10341	10343	10319	1,972921	1,970893	1,968868	1,977905
4	10021	5576	5728	5364	2,035725	3,655129	3,555168	3,804996
6	7359	3900	3943	3650	2,772116	5,225897	5,164595	5,591781
8	5559	3028	3010	2953	3,669725	6,730845	6,765449	6,911615
10	4442	2440	2426	2264	4,592526	8,352869	8,394064	9,015018
12	3887	2082	2090	1919	5,248263	9,789145	9,743541	10,63575
14	3397	1810	1903	1806	6,005299	11,26022	10,701	11,30122
16	3048	1632	1721	1563	6,692913	12,48836	11,83266	13,05822
18	2747	1617	1516	1540	7,426283	12,60421	13,43272	13,25325
20	2474	1402	1338	1259	8,245756	14,53709	15,21973	16,21128
22	2266	1330	1331	1257	9,002648	15,32406	15,29977	16,23707
24	2096	1246	1189	1242	9,732824	16,35714	17,127	16,43317
26	2024	1235	1154	1240	10,07905	16,50283	17,64645	16,45968
28	1828	1149	1145	1248	11,15974	17,73803	17,78515	16,35417
30	1721	1076	1018	934	11,85357	18,94145	20,00393	21,85225
32	1646	1076	1016	930	12,39368	18,94145	20,04331	21,94624

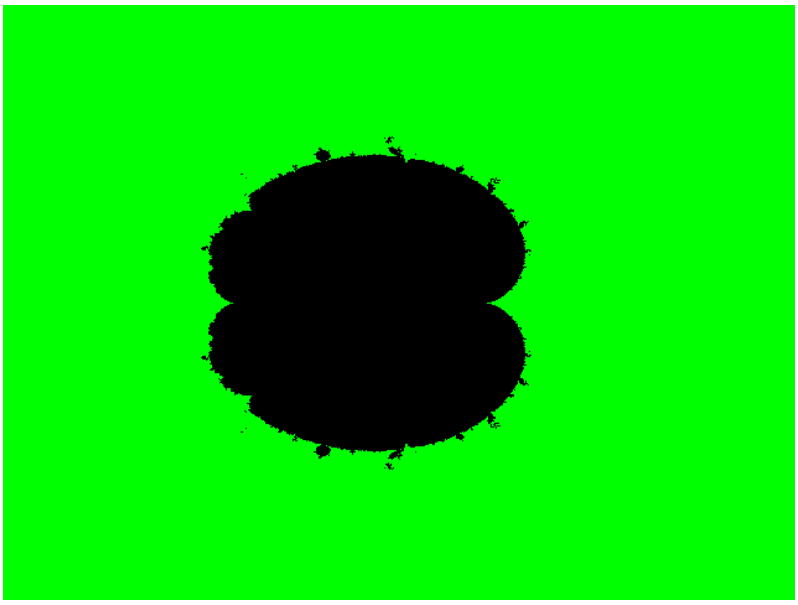



Speed-Up
25
20
15
10
5
0
Sp
1 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32
Number of threads p
Sp1
Sp2
Sp4
Sp8


Efficiency
1,2
1
0,8
0,6
0,4
0,2
0
Ep
1 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32
Number of threads p
Ep1
Ep2
Ep4
Ep8

p	Ep1	Ep2	Ep4	Ep8
1	1	1	1	1
2	0,98646	0,985446	0,984434	0,988952
4	0,508931	0,913782	0,888792	0,951249
6	0,462019	0,870983	0,860766	0,931963
8	0,458716	0,841356	0,845681	0,863952
10	0,459253	0,835287	0,839406	0,901502
12	0,437355	0,815762	0,811962	0,886312
14	0,42895	0,804301	0,764357	0,80723
16	0,418307	0,780522	0,739541	0,816139
18	0,412571	0,700234	0,746262	0,736291
20	0,412288	0,726854	0,760987	0,810564
22	0,409211	0,696548	0,695444	0,738049
24	0,405534	0,681548	0,713625	0,684716
26	0,387656	0,634724	0,67871	0,633065
28	0,398562	0,633501	0,635184	0,584077
30	0,395119	0,631382	0,666798	0,728408
32	0,387303	0,59192	0,626353	0,68582

На диаграмите се вижда, че при най-едрата грануларност (1) ускорението и ефективността са значително по-ниски в сравнение с по-фините грануларности (2,4,8). С нарастване на броя нишки ефектът на грануларността се увеличава, като при 32 нишки най-фината грануларност отчита най-високо ускорение и най-добра ефективност.



T1 – времето за изпълнение на серийната програма (използваща една нишка)

Tp – времето за изпълнение на паралелната програма, използваща p нишки

Sp = T1/Tp – ускорението, което програмата има при използването на p нишки

Ep = Sp/p – ефективността (ефикасността) на програмата при използването на p нишки

4. References

[1] https://en.wikipedia.org/wiki/Mandelbrot_set

[2] Visual Mathematics, Published by: Mathematical Institute of the Serbian Academy of Sciences and Arts, Editor: Ljiljana Radovic, ISSN: 1821-1437, (https://www.mi.sanu.ac.rs/vismath/javier/b2.htm)