Проект

по Системи за паралелна обработка

Паралелен алгоритъм за пресмятане на Pi

Изготвил: Цанка Стефанова Енчева
Компютърни науки, фн 80303, 4 група

Задал: ас. Христо Христов

Проверил:.....
(ас. Христо Христов)

Условие и идея за решаването на задачата

PI е математическа константа, чиято стойност е отношението на обиколката на коя да е окръжност към диаметъра ѝ. Това е същата стойност като отношението на площта на кръг към квадрата на радиуса му. PI е приблизително равно на 3.14159 в десетичен запис. PI е широко използвана константа в математиката, науката и техниката, поради което ще разгледаме един метод за изчислението ѝ.

Ще изчислим константата PI по метода „Монте Карло", който е статистически. Ако имаме вероятност P, че събитие ще настъпи при определени условия, можем да използваме компютър да генерира тези условия многократно. Броят на случаите, когато настъпва събитието, разделен на броя на генерираните случаи трябва да е приблизително равно на P.

Ако кръг с радиус R е вписан в квадрат с дължина на страната 2R, тогава лицето на кръга ще е PI*R*R, а на квадрата – (2R)*(2R). Отношението между лицето на кръга към лицето на квадрата ще е PI/4. Следователно, ако изберем N произволни точки в квадрата, около N*PI/4 ще бъдат вътре в кръга.

Въпреки, че методът „Монте Карло" е полезен при решаването на физични и математични задачи, които не могат да бъдат решени аналитично, той е бавен при изчислението на PI.

Програмата генерира случайни точки в квадрата. Проверява дали те са и в кръга, следи общо колко са генерирани до момента(N) и колко от тях са в кръга(M). Тогава PI ~ (4*M)/N. Имаме множество еднотипни задачи, които са и независими помежду си – генериране на случайни точки (в нашия случай чрез генерирането на случайни числа, които са техните координати). Следователно можем да генерираме точките паралелно като всяка подзадача генерира сходен брой точки като другите подзадачи, използвайки максимално ресурсите на машината. Така времето за изпълнение на всяка от подзадачите ще е сходно. Броят на подзадачите, на които ще разделим задачата, както и броят на точките, които ще генерираме, ни е известен. Подзадачите ще се изпълняват асинхронно и паралелно, за да имат достъп до общите ресурси. Т.е. задачата ще завърши тогава, когато завършат всички подзадачи. Накрая ще сумираме получените в кръга точки за всяко от паралелните изчисления.

Реализация на алгоритъма

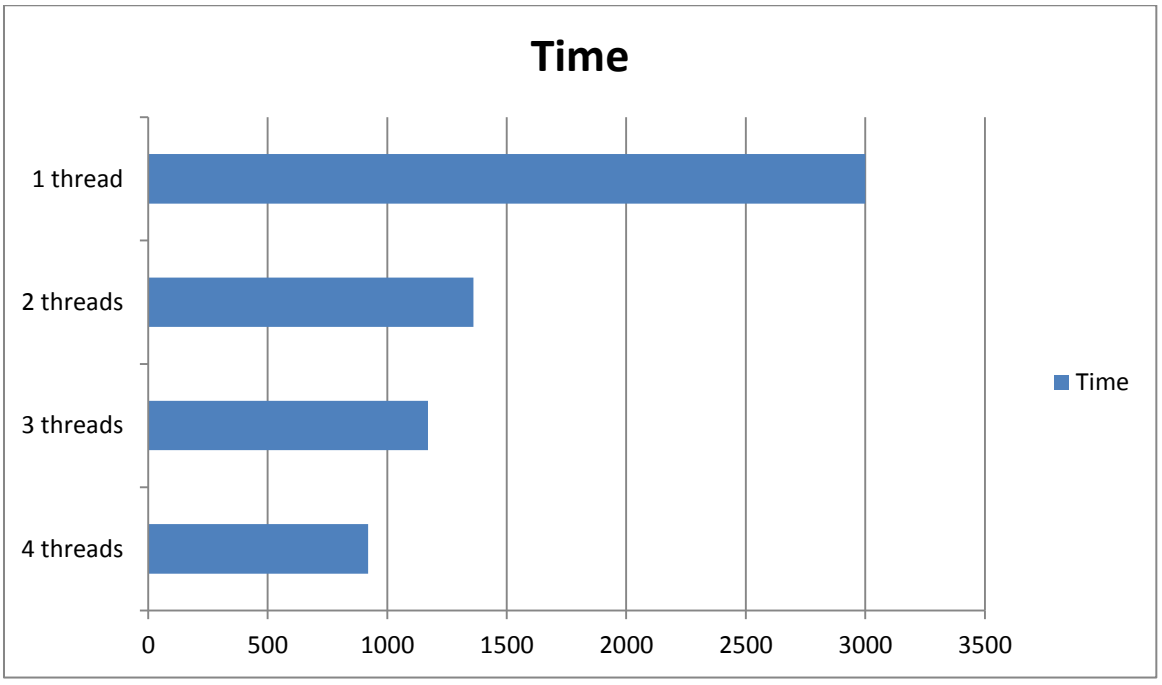
Алгоритъма за намиране на PI реализираме с JAVA multithreading. Използваме метода „разделяй и владей" като организацията на нишките е

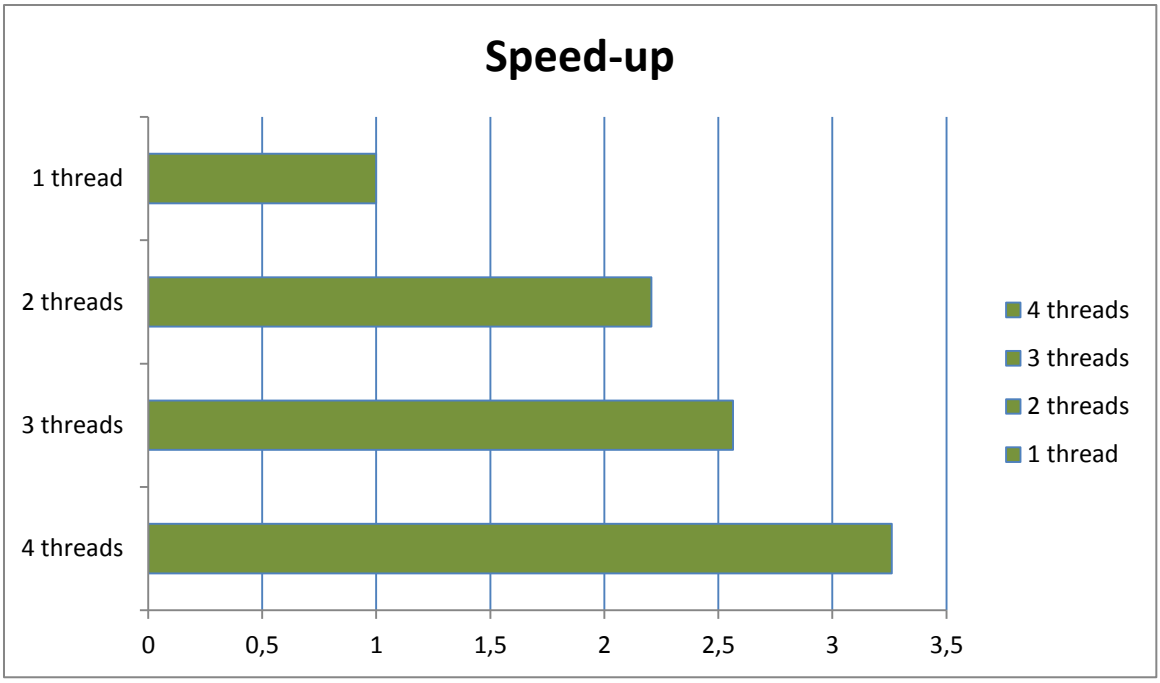
master-slave. Имаме една основна нишка на програмата, която получава входните данни за броя на нишките и броя на генерираните точки и извежда резултата от изчисленията. Всяка от нишките генерира по N/T произволни точки, където N – брой на точките, които трябва да се генерират общо, T – брой на нишките. Последната нишка генерира колкото точки останат за нея. От командния ред се въвеждат с опция –s броят точки, които общо ще генерираме; -t брой нишки, който ще работят; -q за quiet режим на работа без съобщения кога са започнали и завършили нишките. Използва се java.util.Random за генериране на координатите, защото не е синхронизиран, за разлика от Math.random. Всяка нишка изчислява колко от точките, които се генерират, са в кръга. След стартирането им нишките биват изчакани да завършат от главната нишка, за да бъдат събрани всички резултати работата им. Накрая точките в кръга, получени от всяка нишка, се сумират.

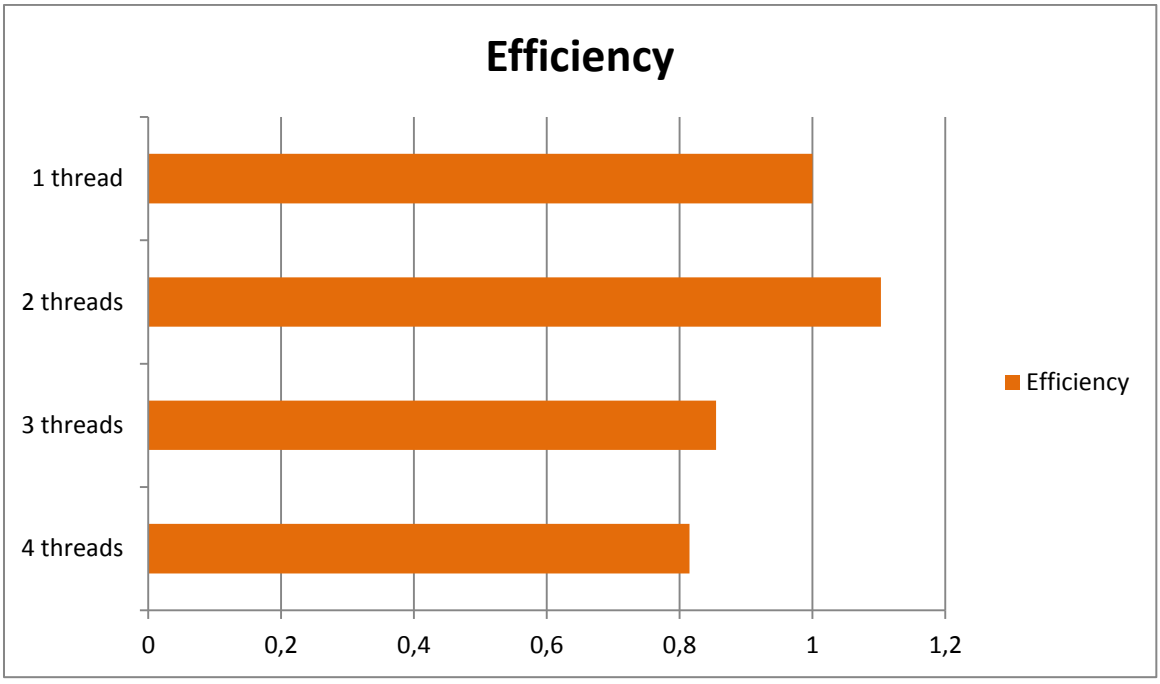
Измерване на време, ефективност и ускорение

Пускаме програмата на 2 ядрен процесор с 4 логически ядра, за да изчислим времето на изпълнение за p на брой нишки T(p), ускорението на p нишки S(p) и ефективността E(p). Тестваме с 10 000 000 точки. Времето измерваме в милисекунди.

$$S(p) = T(1)/T(p)$$
$$E(p) = S(p)/p$$




Speed-up
1 thread
2 threads
3 threads
4 threads
0
0,5
1
1,5
2
2,5
3
3,5
4 threads
3 threads
2 threads
1 thread


Efficiency
1 thread
2 threads
3 threads
4 threads
0
0,2
0,4
0,6
0,8
1
1,2
Efficiency