Tomasz Kuliński 94329

Wojciech Mioduszewski 94344

Sprawozdanie

Temat: Pakowanie palet

Opis

Całe obliczenia polegają na zmianie kolejności przydzielania paczek do palet przez następujący algorytm:

function przydziel\_paczki

Przetwarzanie realizowane jest równolegle przez jeden wątek główny, który odpowiada za synchronizację pozostałych wątków i zapisywanie wyników, natomiast nie przeprowadza właściwych obliczeń i cztery wątki obliczające.

Główny wątek wykonuje następujące czynności:

function wątek\_główny

W pierwszej kolejności lista wczytanych paczek sortowana jest zgodnie z malejącą objętością – otrzymana w ten sposób kolejność paczek przyjmowana jest jako kolejność domyślna. Następnie tworzony jest obiekt współdzielony między wszystkie wątki, przechowuje on najlepszy wynik, podstawowe dane do komunikacji głównego wątku z pozostałymi i kilka semaforów wykorzystywanych do synchronizacji wątków. Dalej następuje utworzenie wątków. Kolejne czynności wykonywane są w pętli: wątek główny oczekuje na zakończenie wykonywanych przez pozostałe wątki obliczeń. Później zmienia temperaturę i ewentualne informuje wątków o zakończeniu przetwarzania (przy ostatniej iteracji), a także zapisanie nowego wyniku, o ile się takowy pojawi (ta ostatnia czynność jest wykonywana po zwolnieniu wątków obliczających, ale dodanie kolejnego semafora zwiększałoby tylko skomplikowanie pseudokodu) i zwolnienie wątków.

Łącznie wykonywane jest 8 iteracji po 120ms, natomiast reszta czasu jest zarezerwowana na potrzeby początkowego przetwarzania i synchronizacji w trakcie obliczeń.

Każdy wątek obliczający wykonuje następujący algorytm:

function oblicz

function wątek\_liczący

Każdy wątek realizuje losowe błądzenie ze zmniejszającą się liczbą zmian w kolejności paczek (metoda losuj wykonywana na liście zmienia pozycję każdego elementu z prawdopodobieństwem równym temperaturze).

Wyniki (11 uruchomień programu dla każdej instancji):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Instancja** | **suma pól** | **objętość** | **min(iteracje)** | **mediana(iteracje)** | **max(iteracje)** |
| **pp101.in** | 576 | 720 | 268314 | 306862 | 316029 |
| **pp102.in** | 864 | 1200 | 268314 | 306862 | 316029 |
| **pp103.in** | 384 | 480 | 3008634 | 3652952 | 3839115 |
| **pp104.in** | 432 | 1440 | 3359585 | 3488989 | 3658890 |
| **pp105.in** | 192 | 480 | 736341 | 1039389 | 1078289 |
| **pp106.in** | 192 | 720 | 1265612 | 1760972 | 1833212 |
| **pp107.in** | 240 | 960 | 840324 | 1138337 | 1167167 |
| **pp108.in** | 288 | 240 | 2088176 | 2837216 | 2957259 |
| **pp109.in** | 432 | 1200 | 309493 | 414884 | 461967 |
| **pp110.in** | 384 | 1200 | 861025 | 948973 | 995081 |

Ze względu na identyczne wyniki dla każdej instancji w każdym uruchomieniu, zdecydowaliśmy się dodatkowo zaprezentować statystyki dotyczące liczby przejrzanych (niekoniecznie unikalnych) kolejności paczek.

Instancja z pliku *pp109.in* podczas rozwoju algorytmu stwarzała najwięcej problemów, gdyż od czasu do czasu dawała różne wyniki w drugim kryterium optymalizacji (wartość 1200 była zastępowana wartością 1440). Dało to podstawy do oceny wprowadzanych rozwiązań poprzez sprawdzanie odsetka wystąpień pożądanej wartości (1200), przy ograniczeniu czasu obliczeń.

Badaliśmy kilka zmian w algorytmie:

* zaimplementowanie symulowanego wyżarzania, co pociągnęłoby za sobą akceptowanie jedynie kolejności polepszających wynik (a także rozwiązań niewiele gorszych – margines zależny od temperatury)
* wprowadzanie co kilka iteracji kolejności losowej, żeby poszukać lepszych rozwiązań nie-lokalnych
* różna liczba i czas iteracji

Wyniki były bardzo losowe, ale nie wskazywały na wyższość któregokolwiek rozwiązania, dlatego zdecydowaliśmy się pozostać przy zaprezentowanej wersji, ze względu na jej prostotę.

Przeprowadzony konkurs pokazał, że decyzja była słuszna (ale też chyba nie miała zbytniego znaczenia) – na 90 uruchomień programu jedynie raz program poprawił wynik osiągnięty w pierwszej iteracji (oczywiście dla pliku *pp109.in*), a wszystkie osiągnięte rezultaty pokrywają się z tymi zaprezentowanymi w powyższej tabeli.

Specyfikacja środowiska:

Procesor: Intel Xeon 1230v3

Pamięć RAM: 16 GB

Java: 1.7.0\_40-b43

Wykorzystywane rdzenie: 4 rdzenie