|  |  |  |
| --- | --- | --- |
| Artur Zochniak | 184725 | 28. maja 2012 |
| Michał Lankof | 184774 | Prowadzący: dr inż. Maciej Lichtenstein  dr inż. Tomasz Krysiak |

**Projekt 2:**

**Struktury danych i złożoność obliczeniowa**

**Szeregowanie zadań na dowolnej ilości procesorów**

# Wstęp

Celem projektu było zaimplementowanie algorytmu szeregowania zadań na dowolnej ilości procesorów i zmierzenie czasu działania tego algorytmu.

Istnieją algorytmy obliczające takie rozdzielenie zadań na procesory, aby rozwiązanie było przybliżone do optymalnego, takie jak np. LPT (longest processing time), który sortuje zadania malejąco po czasie wykonywania i przyporządkowuje od najdłuższej do najkrótszej prace kolejnym wolnym procesorom.

Celem projektu było jednak zaimplementowanie optymalnego algorytmu szukania rozwiązania.

# Opis algorytmu

Idea algorytmu polega na stworzeniu (m+1) wymiarowej przestrzeni, gdzie m to ilość maszyn, którymi dysponujemy. W przestrzeni jeden wymiar będzie reprezentował dochodzące zadania, a m wymiarów będzie reprezentować czas pracy każdego procesora (każde zadanie można przydzielić jednemu z m procesorów co oznacza, że można dane zadanie skierować na jeden z m kierunków).

Algorytm można rozumieć jak przeszukiwanie grafu w celu znalezienia najoptymalniejszego rozwiązania.

Mając m+1 wymiarową przestrzeń zawierającą wartości logiczne należy wypełnić prawie wszystkie pola wartością fałsz. Wszystkie za wyjątkiem pola o współrzędnych (0,0,0,…,0). Wypełnienie tego pola wartością prawda oznacza to, że w tej chwili nie wykonujemy żadnej pracy oraz praca przyporządkowana do każdego procesora nie trwa ani chwili.

Następnie, patrząc z punktu (0,0,…,0) uzupełniamy tablice następnego zadania [rozpoczynającą się od (0+1,…)] przestrzeń wartościami prawda wszędzie tam, gdzie można dostać się za pomocą zwiększenia tylko jednej współrzędnej punktu (0,0,…,0) o długość trwania zadania. Np. mając zadanie trwające t=2 jednostki czasu i szeregowanie na m=4 procesorach i j=5 zadaniach drugim krokiem – po uzupełnieniu punktu (0,0,0,0,0) wartością prawda - jest ustawienie wartości prawda w punktach (1, 2, 0, 0, 0), (1, 0, 2, 0, 0), (1, 0, 0, 2, 0), (1, 0, 0, 0, 2). Operacje te powtarzamy do wypełnienia tablicy ostatniego zadania (o indeksie j=5).

Kolejnym etapem jest *backtracking*. Polega on na tym, że z tych wszystkich kroków, które pozostawił algorytm wstawiając w tablicę (m+1)-wymiarową wartości prawda – należy wrócić do punktu (0,0,…,0). Należy rozpocząć z miejsca ostatniej tablicy zadania (tablicy o pierwszej współrzędnej równej j=5) z punktu, którego wszystkie współrzędne są jak najmniejsze (tj. nie większe niż T. Minimalność współrzędnych w ostatnich m-wymiarach tablicy oznacza to, że podział prac, który doprowadził program do tego punktu kończy się najwcześniej, a ściślej mówiąc wszystkie procesory (maszyny) pracują nie dłużej niż T jednostek czasu).

Optymalne uszeregowanie zadań powstaje poprzez obserwowanie każdego u z zadań równolegle do której osi w przestrzeni przemieścił się punkt. Jeśli punkt przemieścił się równolegle do osi X, to oznacza, że procesor reprezentowany przez te oś powinien podjąć się realizacji zadania u.

Po przejściu wszystkich tablic zadań (adresowanych pierwszą współrzędną) można jednoznacznie określić jakie jest optymalne przyporządkowanie zadań do procesorów.

# Złożoność algorytmu

Algorytm ma złożoność , gdzie to ilość zadań do uszeregowania, to maksymalna praca przypadająca średnio na jeden procesor, a to ilość procesorów. obliczamy jako maksimum ze zbioru {średnia praca na każdy procesor dana wzorem , gdzie to czas trwania zadania, a to ilość zadań; najdłuższa praca}.

# Pomiary ze stałą ilością zadań i zmiennymi czasami trwania zadań

Poniżej prezentujemy wyniki pomiaru czasów trwania zadań. Średnią długość trwania zadania oznacza to, że każde zadanie trwa tyle, ile wynosi średnia długość jego trwania.

Należy jednak pamiętać o tym, że to, w uogólnieniu średnia praca przypadająca na jeden procesor, w związku z tym jeśli zadań jest więcej, to na każdy procesor przypada dłuższy okres średniej pracy. Algorytm opiera się na przeszukiwaniu przestrzeni, której większość wymiarów jest uzależniona właśnie od średniej pracy na każdy procesor. Złożoność rośnie, w przypadku dwóch procesorów wykładniczo zależnością o podstawie do potęgi ilości procesorów.

# 2 maszyny

Rysunek 1. czas trwania szeregowania 5 i 10 zadań na 2 procesorach o dlugości zadań od 1 do 300.

Rysunek 2. czas trwania szeregowania 5, 10, 15, 20, 25, 30, 35 zadań na 2 procesorach o długości zadań od 1 do 300. Przy 7 krotnym zwiększeniu ilości zadań (z 5 do 35) czas wykonania wzrósł z 1,965s do 87,579s. Jest to ponad 20-krotne zwiększenie czasu wykonywania.

## 3 maszyny

Rysunek 3. czas trwania szeregowania 5, 10, 15 zadań na 3 procesorach o długości zadań od 1 do 100. Przy 3 krotnym zwiększeniu ilości zadań (z 5 do 15) czas działania algorytmu wzrósł z 0,359s do 24,70s (jest to prawie 70-krotny wzrost czasu pracy).

# 4 maszyny

Rysunek 4. Przy 5-krotnym wzroście ilości zadań (z 2 do 10) - czas trwania algorytmu zwiększył się z 9612854 taktów procesora do 1499860436 taktów procesora. Daje to ponad 150-krotny wzrost czasu działania.

# 5 maszyn

Rysunek 5. Przy 5-krotnym zwiększeniu ilości zadań algorytm podziału zadań dla 5 procesorów zwiększył swój czas działania ponad 1400 razy (z 0,016s do 12,355s). Przy osi czasu wyskalowanej liniowo już trudno dostrzec linię reprezentującą czas działania algorytmu szeregowania 2 zadań.

Rysunek 6. Dopiero po wprowadzeniu skali logarytmicznej można dostrzec, że ilość taktów procesora, które upłynęły podczas pracy algorytmu dla 2 zadań w stosunku do czasu pracy procesora dla 10 zadań to wartość rzędu ponad 1010/107=103.

# 6 maszyn

Rysunek 7. W przypadku 6 procesorów 3,5-krotny wzrost ilości zadań (z 2 do 7) spowodował wzrost czasu trwania algorytmu z 0,016s do 19,6s (ponad 1700-krotny wzrost czasu wykonywania). Jednak proces mierzenia zakończono nadal przy przy 10 zadaniach, ale przy zmniejszonym maksymalnym zakresie danych wejściowych (średni czas trwania zadania nie trwa 10 jednostek czasu, a 7). Dla porównania z szeregowaniem zadań na 5. procesorach - wzrost ilości zadań z 2 do 7 wpłynął na czas działania zwiększając go z 9920656 do 4391341350 taktów (ponad 440-krotny wzrost).

# Czasy szeregowania 10 zadań w zależności od maksymalnych danych (średniego czasu trwania pojedynczego zadania) na różnej ilości procesorów

Rysunek 8. Wykres pokazujący jak wielka jest różnica w czasie obliczania optymalnego uszeregowania na 2 procesory w porównaniu z czasem obliczania optymalnego uszeregowania dla 6 procesorów.

Rysunek 9. Ponownie - dopiero zmiana skalowania osi czasu trwania algorytmu z liniowego na logarytmiczne pozwala porównać czasy trwania algorytmów. Zwiększenie ilości procesorów z 3 do 5, przy 10. zadaniach i średnim czasie trwania każdego zadania 10 jednostek czasu, powoduje zwiększenie czasu obliczania optymalnego uszeregowania z ok. 10^8 taktów do ok. 10^10, co daje stosunek rzędu 100-wielokrotności (dokładnie zwiekszenie czasu działania algorytmu 269 razy).

# Wnioski

* problem szeregowania zadań jest istotnym problemem, jednak jeśli wszędzie wykorzystywano by algorytm znajdujący optymalne rozwiązanie to wszystkie urządzenia i usługi działałyby zdecydowanie wolniej. Rozsądnym wydaje się, szczególnie w przetwarzaniu w czasie rzeczywistym, zrezygnowanie z najlepszej jakości otrzymanych wyników w celu obliczenia rozwiązania w zdecydowanie krótszym czasie,
* problem szeregowania zadań ma złożoność pseudowielomianową dlatego że czas obliczenia optymalnego rozwiązania zależy zarówno od ilości danych wejściowych jak i od ich zakresu w znaczącym stopniu.

# Literatura

1. J. Błażejewicz, K. Ecker, E. Pesch, G. Schmidt, J. Węglarz, *Handbook on Scheduling*, Springer, Berlin Heidelberg 2007, str. 137-143 *Minimize Schedule Length*.