ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1988

Seria:AUTOMATYKA z.94 Nr kol.970

Jacek Błażewicz, Maciej Drozdowski,
Jan Węglarz
Politechnika Poznańska

SZEREGOWANIE ZADAŃ WIELOPROCESOROWYCH W SYSTEMIE JEDNORODNYCH DUOPROCESORÓW

Streszczenie. W pracy przeanalizowano problem szeregowania zadań dwuprocesorowych, to znaczy żądających do swego wykonania jednocześnie dwóch procesorów, w systemie jednorodnych duoprocesorów - czyli par identycznych procesorów. Rozpatrzono minimalizację długości uszeregowania dla zadań podzielnych, niezależnych.

1. Wstęp

Jednym z przyjmowanych zazwyczaj założeń w teorii szeregowania zadań jest wymaganie, by każde zadanie było w każdej chwili wykonywane przez co najwyżej jeden procesor. Jednakże w ostatnich latach wraz z intensywnym rozwojem systemów wieloprocesorowych, założenie to traci na oczywistości i jest podważane. Ostatnio ukazało się szereg prac poświęconych problematyce szeregowania zadań żądających więcej niż jednego procesora w każdej chwili czasu [3-7,11,13], w szczególności w kontekście systemów testujących i diagnostycznych [2,10,11,14]. Przypadek procesorów identycznych był rozważany w pracach [3,4,5]. Udało się sformułować algorytm dokładny szeregujący zadania 1- i k-procesorowe w czasie O(n) [5]. Algorytmy uwzględniające dodatkowe ograniczenia zasobowe sformułowano w [6,7]. W niniejszej pracy rozważymy zagadnienie szeregowania zadań dwuprocesorowych w systemie duoprocesorów jednorodnych.

Rozpatrywać będziemy zbiór procesów jednorodnych $\mathcal{P}=\{P_1,P_2,\ldots,P_m\}$ o prędkościach $s_1=s_2$, $s_3=s_4,\ldots,s_{m-1}=s_m$, gdzie $m=2\cdot l$ i l jest liczbą naturalną. Mamy zatem pary procesorów identycznych, zwane duoprocesorami. Dany jest zbiór zadań niezależnych i podzielnych $\mathcal{Z}$, składający się z dwóch podzbiorów: $\mathcal{T}=\{T_1,\ldots,T_{n_t}\}$ zadań jednoprocesorowych i $\mathcal{W}=\{W_1,W_2,\ldots,W_{n_w}\}$ - zadań dwuprocesorowych - czyli wymagających do swego wykonania jednocześnie dwóch procesorów ($n_t+n_w=n$). Standardowe czasy wykonywania zadań są równe odpowiednio: w zbiorze $\mathcal{T}$ - $t_1,t_2,\ldots,t_{n_t}$, a w zbiorze $\mathcal{W}$ - $w_1,w_2,\ldots,w_{n_w}$.

W rozdziale 2 zostanie przedstawiony algorytm szeregowania dla powyższego problemu, o złożoności O(nlogn). Rozdział 3 zawiera przykład zastosowania tego algorytmu.

2. Algorytm szeregowania

Dla uszeregowania zadań W (potrzebujących dwóch procesorów jednocześnie) i zadań T (jednoprocesorowych), dolne ograniczenie czasu wykonania może być wyznaczone przez rozważenie dwóch niezależnych, osłabionych wersji tego problemu:

1) tylko zadań typu W,
2) zadań typu T i W, gdzie każde zadanie $W_j \in \mathcal{W}$ jest traktowane jak dwa niezależne zadania jednoprocesorowe.

Niech zbiór duoprocesorów będzie uporządkowany tak, że $s_1=s_2 \geq s_3 \ldots \geq s_{m-1} = s_m$, a zbiór zadań zgodnie z nie rosnącymi czasami wykonania. Długość uszeregowania dla wersji 1 i 2 będzie wynosić odpowiednio:

$$C(1) = \max \left\{ \max_{1\leq l\leq m/2} \left\{ \sum_{j=1}^{l} w_j / \sum_{j=1}^{l} s_{2j} \right\}, \sum_{j=1}^{n_W} w_j / \sum_{j=1}^{m/2} s_j \right\}$$

$$C(2) = \max \left\{ \max_{1\leq l\leq m} \left\{ \sum_{j=1}^{l} t_j / \sum_{j=1}^{l} s_j \right\}, \sum_{j=1}^{n} t_j / \sum_{j=1}^{m} s_j \right\},$$

gdzie t_j jest rozumiane jako czas wykonywania zadania $T_j \in \mathcal{T}$ lub jednego z dwóch zadań reprezentujących zadanie $W_j \in \mathcal{W}$.

Stąd $C=\max\{C(1), C(2)\}$ jest dolnym ograniczeniem czasu wykonania zbioru zadań. Niech moc obliczeniowa każdego procesora w przedziale $[\emptyset,C]$ przed szeregowaniem będzie określona jako $PC_i = s_i \cdot C$. Idea podejścia wywodzi się z algorytmu Gonzalesa-Sahniego [9,12]. Dla naszych celów postąpimy następująco. Szeregujemy najpierw zadania ze zbioru $\mathcal{W}$, rozważając zbiór procesorów $\mathcal{P} = \{P_{2i} | i=1,\ldots m/2\}$. Następnie stosując ten sam algorytm do zbioru procesorów $\mathcal{P} = \{P_i | i=1,\ldots,m\}$ pozostałych po uszeregowaniu zadań ze zbioru $\mathcal{W}$, szeregujemy zadania ze zbioru $\mathcal{T}$. Poniżej przedstawione zostaną reguły szeregowania zadań. Dla uproszczenia oznaczymy tu przez T_j zarówno $W_i \in \mathcal{W}$, jak i $T_i \in \mathcal{T}$, gdyż zadania obu typów są traktowane jak jednoprocesorowe. Stosownie do poniższych warunków, algorytm stosuje następujące reguły.

Reguła 1 (gdy $t_j = PC_k$).

Uszereguj T_j na P_k, tak, że w przedziale $[\emptyset,C]$ jest on całkowicie zajęty. $PC_k := \emptyset$ i przenumeruj zbiór procesorów zgodnie z rosnącymi mocami obliczeniowymi.

Reguła 2 (gdy $PC_k > t_j > PC_{k+1}$)

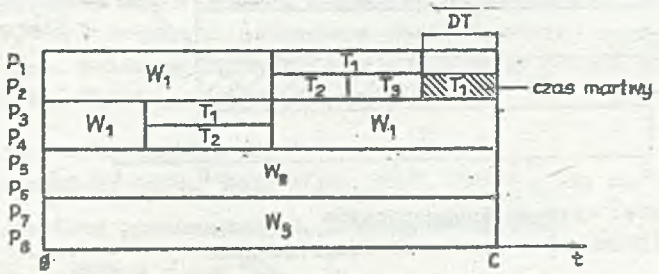
Oblicz moment z taki, że T_j jest uszeregowane w spójnych przedziałach $[\emptyset,z]$ na P_k i $[z,C]$ na P_{k+1}. Połącz P_k i P_{k+1} w jeden złożony procesor o mocy $PC_k := PC_k + PC_{k+1} - t_j$.

$PC_{k+1}:=\emptyset$ i przenumeruj zbiór procesorów zgodnie z nie rosnącymi mocami obliczeniowymi.

Reguła 3 (gdy $t_j < PC_k$ i $P_k=P_m$ lub $PC_{k+1}=\ldots=PC_m=\emptyset$)

Uszereguj T_j z lewa na prawo, aż do osiągnięcia momentu $1 \leq C$ takiego, że P_k jest zajęty wykonywaniem zadania T_j w spójnym przedziale czasu $[\emptyset,1]$. $PC_k:= PC_k - t_j$.

Zatem po wyznaczeniu dolnego ograniczenia czasu wykonania C możemy uszeregować uporządkowany zbiór $\mathcal{W}$ na m/2 duoprocesorach. Ponieważ $C \geqslant C(1)$, dlatego uszeregowanie zbioru $\mathcal{W}$ jest zawsze możliwe. Z warunku $C \geqslant C(2)$ wynika, że istnieje wystarczająca moc obliczeniowa, aby uszeregować także zadania ze zbioru $\mathcal{T}$. Może jednak zaistnieć sytuacja taka, że uszeregowanie wszystkich zadań ze zbioru $\mathcal{T}$, po zadaniach ze zbioru $\mathcal{W}$, nie będzie możliwe. Wynika to z faktu, iż pewne zadanie $T_j \in \mathcal{T}$ jest na tyle długie, że uniemożliwia dopuszczalne uszeregowanie.
Z sytuacją taką związane jest pojęcie czasu martwego (DT) [3] , który definiuje się jako część czasu wykonania, w którym zadanie jednoprocesorowe przydzielone jest do więcej niż jednego procesora (rys.1).



Rys.1. Ilustracja czasu martwego
Fig.1. Example of a dead time

Czas martwy rozumieć można także jako niedobór mocy obliczeniowej, wskutek którego zadanie jednoprocesorowe jest przydzielone do więcej niż jednego procesora. Należy zwrócić uwagę na fakt, że w rozważanym przypadku zadań jedno- i dwuprocesorowych tylko dla jednego zadania T_j może wystąpić czas martwy.

Oznaczmy przez $PC_i^{\mathcal{W}}$ moc obliczeniową pozostałą na P_i po uszeregowaniu zadań ze zbioru $\mathcal{W}$ przy wykorzystaniu jedynie reguł 1,2 i 3. Niech $pc_i^{\mathcal{W}}$ będzie mocą obliczeniową P_i, dla jakiegokolwiek innego dopuszczalnego uszeregowania zadań W. Mamy wówczas następujące twierdzenie [8] .

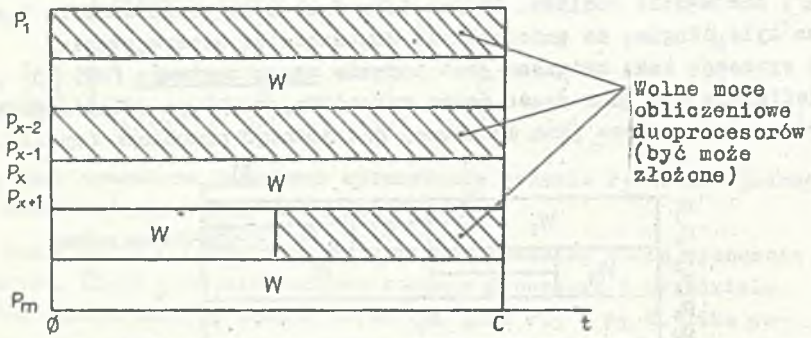
Twierdzenie 1

Przy podanych założeniach

$$\sum_{j=1}^{i} PC_j^{W} \geq \sum_{j=1}^{i} pc_j^{W} \qquad \text{dla } i=1\ldots m$$

Oznacza to, że nie można uzyskać większej mocy obliczeniowej po uszeregowaniu zadań W, niż wynika to z zastosowania reguł 1,2,3.

Po uszeregowaniu zadań typu W, system duoprocesorów można podzielić na dwa zbiory: $\{P_1 \ldots P_{x-1}\}$ i $\{P_x \ldots P_m\}$ (rys.2). W pierwszym z nich znajdują się tylko duoprocesory (być może złożone) całkowicie zajęte lub całkowicie wolne w przedziale $[\emptyset, C]$, przy czym musi istnieć co najmniej jeden duoprocesor (P_{x-2}, P_{x-1}) całkowicie wolny. Na drugi zbiór składają się duoprocesory (być może złożone) zajęte całkowicie w przedziale $[\emptyset, C]$ i co najwyżej jeden duoprocesor zajęty częściowo - w przedziale $[\emptyset, 1]$ $(1 < C)$.



Rys.2. Podział systemu duoprocesorów
Fig.2. Partition of a system of duoprocessors

Niech wszystkie moce PC_i^{W} będą uporządkowane wg malejących wartości. Jak wiemy z [9,12], jeżeli spełniony jest warunek

$$\max\left\{\max_{1\leq k<m}\left\{\sum_{j=1}^{k} t_j \Big/ \sum_{j=1}^{k} PC_j^{W}\right\}, \sum_{j=1}^{n_t} t_j \Big/ \sum_{j=1}^{m} PC_j^{W}\right\} \leq 1 \qquad (1)$$

to zadania ze zbioru $\mathcal{T}$ mogą zostać uszeregowane w sposób dopuszczalny. W przeciwnym razie istnieje zadanie T_j, które powoduje powstanie czasu martwego DT. Można wykazać [8] poprawność następującego twierdzenia

Twierdzenie 2

Jeżeli nie istnieje dopuszczalne uszeregowanie dla zbioru zadań $\mathcal{T}$ $(DT > \emptyset)$, to długość optymalnego uszeregowania jest dana wzorem:
$C^* = C + \delta$, gdzie

$$\delta = DT \Big/ \left(\sum_{i=1}^{x-1} s_i + \frac{1}{2}\sum_{i=x}^{m} s_i\right)$$

Możemy teraz przystąpić do ostatecznego sformułowania naszego algorytmu.

Algorytm

1. Uporządkuj $\mathcal{T}$ i $\mathcal{W}$ według czasów wykonywania.
2. Wyznacz C(1), C(2), C, PC_i, i=1,2,...,m.
3. Uszereguj zbiór zadań, korzystając z reguł 1 - 3.
4. Sprawdź warunek (1). Jeżeli jest spełniony, to przejdź do kroku 7.
5. Wyznacz DT (np. z zależności (1)), $\mathcal{T}$, C*.
6. Ponownie uszereguj zadania typu W.
7. Uszereguj zbiór zadań $\mathcal{T}$.

Złożoność algorytmu jest O(nlogn), co wynika z sortowania zadań w kroku 1. Kroki 3,6 i 7 mają bowiem złożoność O(n), gdyż całkowita liczba przełączeń zadań jest rzędu O(2m) [9,12] . Tego samego rzędu jest więc całkowita złożoność wyznaczenia chwil przełączenia dla kolejnych zadań (zakładamy $m \leq \log n$).

3. Przykład

Dane są zbiory zadań $\mathcal{W}$ i $\mathcal{T}$ o czasach wykonywania $\overline{w} = [20,5]$, $\overline{t} = [10,10,10]$. Prędkości procesorów są następujące: $s_1=s_2=2$, $s_3=s_4=s_5=s_6=1$. Dolne ograniczenie czasu wykonania zbioru zadań, $C = \max\{C(1), C(2)\}$, gdzie

$C(1) = \max\{20/2, 25/3\} = 10,$

$C(2) = \max\{20/2, 40/4, 50/5, 60/6, 70/7, 80/8\} = 10,$ czyli C=10

Moce obliczeniowe poszczególnych procesorów są równe

$PC_{1,2} = 20, \quad PC_{3,4,5,6} = 10.$

Przystępujemy do szeregowania zadań W.

- j=1 (zadanie W_1), $w_1=20=PC_2$. W_1 zajmuje zatem procesory P_1 i P_2.
- j=2 (zadanie W_2), $w_2=5 < PC_6=10$ - postępujemy zgodnie z regułą 3. Zadanie W_2 zajmuje P_5 i P_6 w przedziale $[0,5]$, $PC_6=10-5=5$.

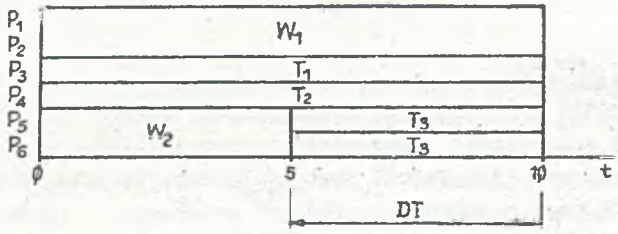
Sprawdzamy teraz warunek (1). Po uszeregowaniu zbioru $\mathcal{W}$ otrzymaliśmy następujący rozkład mocy obliczeniowych: $PC_{3,4} = 10$, $PC_{5,6} = 5$. Mamy stąd następujące nierówności:

$PC_3=10 \geq t_1=10$, czyli $t_1/PC_3 \leq 1$

$PC_3 + PC_4 = 20 \geq t_1+t_2 = 20$, czyli $(t_1+t_2)/(PC_3+PC_4) \leq 1$

$PC_3+PC_4+PC_5=25 < t_1+t_2+t_3=30$, czyli $(t_1+t_2+t_3)/(PC_3+PC_4+PC_5) > 1$.

Ostatnia nierówność warunku (1) nie jest zatem spełniona. Czas nartwy wynosi DT=5, co ilustruje rys.3.



Rys.3. Wyznaczenie czasu martwego w rozpatrywanym przykładzie
Fig.3. Definition of the dead time in the considered example

Obliczamy zatem C^*:

$$\delta = DT/\Big(\sum_{i=1}^{x-1} s_i + \frac{1}{2}\sum_{i=x}^{m} s_i\Big) = DT/\Big(\sum_{i=1}^{4} s_i + \frac{1}{2}\sum_{i=5}^{6} s_i\Big) = 5/(6+1)$$

$C^* = 10\,5/7$, stąd $PC_{1,2} = 21\,3/7$, $PC_{3,4,5,6} = 10\,5/7$.

Przystępujemy do ponownego szeregowania zadań ze zbioru $\mathcal{W}$.

- j=1. $w_1=20$, stąd $PC_2 > w_1 > PC_4$ i postępujemy zgodnie z regułą 2.
  Wyznaczamy moment t_A przełączenia zadania W_1

$$w_1 = t_A s_2 + (C - t_A)s_4, \text{ stąd } t_A = \frac{w_1 - PC_4}{s_2 - s_4} = \frac{20 - 10\,5/7}{1} = 9\,\frac{2}{7}$$

Z procesorów P_2 i P_4 formujemy procesor złożony P'_2.

$PC'_2 = PC_2 + PC_4 - w_1 = 32\,1/7 - 20 = 12\,1/7$.

- j=2. $w_2=5 < PC_6 = 10\,5/7$, zatem stosujemy regułę 3 i $PC'_6 = PC_6 - w_2 = 5\,5/7$.

Uzyskujemy rozkład mocy: $PC'_{1,2} = 12\,1/7$, $PC'_{5,6} = 5\,5/7$.

Wyznaczamy uszeregowanie zadań ze zbioru $\mathcal{T}$.

- j=1. $t_1=10$, stąd $PC'_2 = 12\,1/7 > t_1 > PC'_5 = 5\,5/7$ i stosujemy regułę 2.

Moment przełączenia: $t_B = \frac{t_1 - PC'_5}{s_4} = 10 - 5\,5/7 = 4\,2/7$

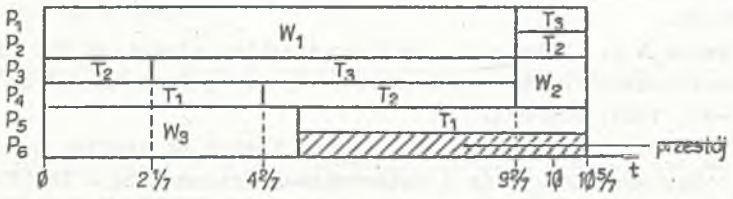
$PC'_{\frac{1}{2}} = PC'_2 + PC'_5 - t_1 = 17\,6/7 - 10 = 7\,6/7$

- j=2. $t_2=10$, zatem $PC'_1 = 12\,1/7 > t_2 > PC'_{\frac{1}{2}} = 7\,6/7$ i stosujemy regułę 2.

Moment przełączenia: $t_C = \frac{t_2 - PC'_{\frac{1}{2}}}{s_3} = 10 - 7\,6/7 = 2\,1/7$; $PC'_1 = 20 - 10 = 10$.

- j=3. $t_3=10=PC'_1$. Zatem wybieramy regułę 1.
  $PC'_6 = 5\frac{5}{7}$

Uzyskane uszeregowanie ilustruje rysunek 4.



Rys.4. Optymalne uszeregowanie zadań w rozpatrywanym przykładzie
Fig.4. Optimal schedule for the considered example

LITERATURA

[1] Aho A.V., Hopcroft J.E., Ullman J.D.: Projektowanie i analiza algorytmów komputerowych, PWN, Warszawa 1983.

[2] Avizienis A.: Fault tolerance: the survival attribute of digital systems, Proc. of IEEE 66, No.10, 1978, 1109-1125.

[3] Błażewicz J., Drabowski M., Węglarz J.: Scheduling independent 2-processor tasks to minimize schedule length. Information Processing Letters, 18, No.5, 1984, 267-273.

[4] Błażewicz J., Drabowski M., Węglarz J.: O niektórych problemach szeregowania zadań wieloprocesorowych, Zeszyty Naukowe Politechniki Śląskiej, s.Automatyka, No.74, 1984, 39-48.

[5] Błażewicz J., Drabowski M., Węglarz J.: Scheduling multiprocessor tasks to minimize schedule length, IEEE Transactions on Computers, Vol. C-35, No.5, May 1986, 389-393.

[6] Błażewicz J., Drabowski M., Węglarz J.: Szeregowanie zadań wieloprocesorowych z ograniczeniami zasobowymi, Zeszyty Naukowe Politechniki Śląskiej, s.Automatyka, No.84, 1986, 27-33.

[7] Błażewicz J., Drabowski M., Ecker K., Węglarz J.: Multiprocessor task scheduling with single resource constraints, Proc. 1st European Workshop on Parallel Processing Techniques, Manchester, 1985.

[8] Błażewicz J., Drozdowski M., Schmidt G., de Werra D.: Scheduling independent two-processor tasks on a uniform duoprocessor system - w przygotowaniu do druku.

[9] Gonzalez T., Sahni S.: Preemptive scheduling of uniform processor systems, J. ACM 25, 1978, 92-101.

[10] Hakimi S.L., Amin A.T.: Characterization of connection assignment of diagnosable systems, IEEE Trans. on Computers C-23, No.1, 1974, 86-88.

[11] Krawczyk H., Kubale M.: An approximation algorithm for diagnostic test scheduling in multicomputer systems, IEEE Trans. on Comput. C-34, 1985, 869-872.

[12] Lawler E.L.: Recent results in the theory of machine scheduling in Bachem et al. (eds.) Mathematical Programming - The State of the Art, Springer Verlag 1983, 200-234.

[13] Monma C.L.: A scheduling problem with simultaneous machine requirement, TIMS XXVI, Copenhagen 1984.

[14] Preparata F.P., Metze G., Chien R.T.: On the connection assignment problem of diagnosable system, IEEE Trans. on Electronic Comput., EC-16, No.6, 1967, 848-854.

Recenzent:Doc.dr hab.inż.J.Kłamka

Wpłynęło do Redakcji do 1988-04-30.

РАСПИСАНИЕ МНОГОПРОЦЕССОРНЫХ ЗАДАЧ В СИСТЕМЕ ОДНОРОДНЫХ ДУОПРОЦЕССОРОВ

Р е з ю м е

В работе даётся анализ проблем составления расписания двупроцессорных задач , т.е. задач , требующих для выполнения одновременно двух процессоров, в системе однородных дуопроцессоров, т.е. пар одинаковых процессоров. В работе рассмотрена проблема минимизации длительности ожидания для получения результатов выполнения расписания прерываемых и независимых задач.

SCHEDULING MULTIPROCESSOR TASKS IN A SYSTEM OF UNIFORM DUOPROCESSORS

S u m m a r y

In this paper we analyse the problem of scheduling two-processor tasks, i.e. requiring two processors at a time, in a system of uniform duoprocessors, i.e. pairs of identical processors. The minimization of the schedule length is considered for splittable, independent tasks.