ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 2008
Seria: AUTOMATYKA z. 150 Nr kol. 1796

Artur BANUL, Konrad WALA
Wyższa Szkoła Biznesu w Dąbrowie Górniczej

HARMONOGRAMOWANIE STATYCZNE ZALEŻNYCH ZADAŃ OBLICZENIOWYCH W HOMOGENICZNYM SYSTEMIE WIELOPROCESOROWYM

Streszczenie. Przedstawiono model formalny statycznego problemu harmonogramowania zależnych zadań obliczeniowych w homogenicznym systemie wieloprocesorowym. Opisano sześć algorytmów konstrukcyjnych harmonogramowania, a następnie, biorąc pod uwagę szereg ważnych kryteriów oceny jakości, zaprezentowano wyniki badań komputerowych ich efektywności.

STATIC SCHEDULING OF DEPENDENT COMPUTATIONAL TASKS IN HOMOGENEOUS MULTIPROCESSOR SYSTEM

Summary. A formal model of static scheduling problem of dependent computational tasks in homogeneous multiprocessor system is presented. We give a description of six constructive scheduling algorithms and than, taking into account a number of important efficiency criteria, we picture the results of computational investigations of their performance.

1. Wstęp

NP-trudny problem harmonogramowania zadań obliczeniowych polega na takim przydziale zadań do równolegle pracujących procesorów, który minimalizuje terminy zakończenia zadań. Prezentowana praca dotyczy problemu harmonogramowania statycznego, kiedy to harmonogram jest wyznaczony przed wykonaniem aplikacji na jednostce wieloprocesorowej za pomocą algorytmów klasy APN (ang. *Arbitrary Processors Network*), tj. algorytmów przydzielających zadania dla dowolnej architektury sieci procesorów.

W punkcie 2 przedstawiono model formalny optymalizowanego harmonogramu, w punkcie 3 opisano badane algorytmy konstruujące dopuszczalne harmonogramy, a punkt 4 zawiera wybrane wyniki badań komputerowych. W zakończeniu podano krótkie podsumowanie badań komputerowych.

2. Opis problemu

Rozważany jest wieloprocesorowy system komputerowy złożony ze zbioru $M = \{1, 2, ..., m\}$ identycznych procesorów, z których każdy posiada własną pamięć RAM

oraz taką samą moc obliczeniową, tj. czas wykonania danego zadania na dowolnym procesorze jest jednakowy. Procesory systemu komunikują się między sobą przez dwukierunkowe połączenia o jednakowej przepustowości. Struktura sieci procesorów jest jednym z parametrów problemu i w szczególnym przypadku może być typu „każdy z każdym". Przyjęto, że czas tracony na komunikację zadań przydzielonych do tego samego procesora jest pomijany, gdyż operacje te wykonywane są w jego pamięci RAM. Komunikaty od zadań nadawczych do zadań odbiorczych przydzielonych do różnych procesorów są przesyłane kanałami komunikacyjnymi, przy czym czasy przesyłu są zależne od rozmiaru komunikatu i długości kanału estymowanej liczbą procesorów pośredniczących w procesie przesyłu komunikatu, stąd czas przesyłu komunikatu pomiędzy zadaniami alokowanymi do różnych procesorów jest zwielokrotniany przez liczbę procesorów pośredniczących w procesie przesyłania danych (por. [2], [4]).

Równoległy program komputerowy jest podzielony na elementarne, niepodlegające dalszemu podziałowi, zadania obliczeniowe wykonywane na pojedynczych procesorach, bez przerwań podczas procesu realizacji. Program jest modelowany za pomocą ważonego acyklicznego digrafu WTPD (ang. *Weighted Task Precedence Digraph*) $G = (V, E)$, gdzie zbiór wierzchołków V reprezentuje zbiór zadań, a zbiór łuków E relację poprzedzania w zbiorze V. Zakładamy znajomość *a priori* funkcji p: $V \rightarrow R^+$ określającej średni czas wykonania zadań oraz funkcji a: $E \rightarrow R^+$ określającej wagę łuków definiującą średni jednostkowy czas przesłania komunikatu pomiędzy zadaniami przydzielonymi do różnych procesorów, gdzie czas przesłania komunikatu pomiędzy konkretnymi procesorami badanej architektury jest równy czasowi jednostkowemu pomnożonemu przez liczbę procesorów pośredniczących w procesie przesyłania (por. [2], [5]).

3. Algorytmy konstrukcyjne

Daną wejściową dla algorytmu konstrukcyjnego jest digraf ponumerowanych topologicznie zadań obliczeniowych. Algorytm, na podstawie heurystycznych reguł, przydziela dla każdego procesora podzbiór uporządkowanych liniowo w podciąg zadań, gdzie kolejność zadań w podciągu jest zgodna z porządkiem częściowym wprowadzonym w zbiorze zadań V przez digraf G. Na tej podstawie wyznaczane są terminy zakończenia zadań C_j, $j \in V$, przy czym uwzględniane są czasy przesyłania komunikatów między procesorami charakterystyczne dla badanej architektury połączeń procesorów w systemie wieloprocesorowym.

W pracy podano wyniki badan następujących algorytmów konstrukcyjnych:

1. Algorytm BSA (ang. *Bubble Scheduling and Allocation*) Ahmada i Kwoka, działa na zasadzie przydzielenia wszystkich, odpowiednio uszeregowanych (szczegóły w [5]), zadań do procesora centralnego o największej liczbie połączeń z sąsiednimi. W kolejnych krokach algorytmu zadania przydzielane są do procesorów sąsiednich, jeżeli nie spowoduje to zwiększenia terminu zakończenia zadań.

2. Algorytm RL (ang. *Ready List*) bazuje na technice listy gotowych zadań. Na początku przydziela zadania niemające poprzedników do procesora o największej liczbie połączeń z procesorami sąsiednimi. W kolejnych iteracjach algorytm wyznacza podzbiór zadań, których poprzedniki zostały już przydzielone do procesorów.

Następnie zadania podzbioru są szeregowane zgodnie z niemalejącą wartością rezerw czasowych zadań w digrafie ważonym (*G*, *p*, *a*) i przydzielane do procesorów gwarantujących im najwcześniejszy termin rozpoczęcia (por. [1], [5]).

3. Algorytm PL (ang. *Priority List*) jest oparty na koncepcji listy priorytetów wyznaczanej dla wszystkich zadań na początku działania algorytmu. W pracy przebadano algorytm z funkcją priorytetu *t_level(i)* określającą najdłuższą ścieżkę od wierzchołka początkowego do wierzchołka *i* (z wyłączeniem *i*) w digrafie ważonym (*G*, *p*, *a*). W pierwszej iteracji algorytmu zadania bez poprzedników przydzielone są do procesora o największej liczbie połączeń z procesorami sąsiednimi. W następnych iteracjach wybierane są zadania o największej wartości priorytetu z podzbioru zadań nieprzydzielonych (por. [1], [5]).

4. Algorytm LC (ang. *Linear Clustering*, [3], [2]) Kim'a i Browne'a, który w pracy przystosowano do klasy algorytmów *APN*. Algorytm *LC* przydziela w pierwszej kolejności zadania leżące na ścieżce krytycznej digrafu ważonego (*G*, *p*, *a*). W kolejnych iteracjach przydzielone zadania usuwane są z digrafu i ścieżka krytyczna wyznaczana jest od nowa. W algorytmie została zastosowana dodatkowa procedura PRECEDE szeregująca zadania na procesorze zgodnie z porządkiem częściowym określonym przez digraf.

5. Algorytm EZ (ang. *Edge Zeroing*, [4],[2]) Sarkara, przystosowany także do klasy algorytmów APN, przydziela zadania do procesorów, pomiędzy którymi jest największy czas przesyłu komunikatów, przy czym do szeregowania zadań przydzielonych do procesora zastosowano procedurę PRECEDE.

6. Algorytm LCEZ (ang. *Linear Clustering & Edge Zeroing*) stanowi hybrydowe połączenie dwóch opisanych powyżej. W pierwszej kolejności działa algorytm LC, a następnie kontrolę przejmuje algorytm EZ; i w tym algorytmie zastosowano procedurę PRECEDE.

4. Badania komputerowe

Do badań komputerowych wygenerowane zostały 72 klasy digrafów zadań po 100 egzemplarzy w każdej z klas. Grafy zadań sklasyfikowane zostały według: liczby zadań w grafie: {50; 75; 100}, współczynników gęstości łuków: {0,1; 0,3; 0,5; 0,7}, współczynnika CCR^1 : {0,1; 0,2; 1; 2; 5; 10}, przy czym czasy zadań były losowane z przedziału wartości [100, 500]; w sumie 7 200 digrafów.

Do badań zastosowano następujące architektury wieloprocesorowe: 9-procesorowe połączenie typu „każdy z każdym", pierścień 4, 5, 10 procesorów, torus 16 procesorów, 6-procesorowa struktura gwiazdy, 9-procesorowa sieć 3x3, 27-procesorowa hiperkostka 3x3x3.

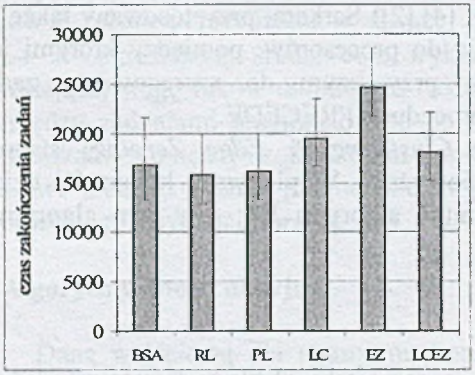
Jednym z kryterium oceny rozwiązań wyznaczonych przez algorytmy konstrukcyjne, dalej nazywane krótko rozwiązaniami, jest procentowy stopień poprawy rozwiązań przez klasyczny algorytm TS (ang. *Tabu Serach*) realizujący 100 iteracji poprawy dla czasu tabu 10, gdzie czas tabu jest parametrem pamięci krótkoterminowej algorytmu TS, określającym liczbę iteracji tabu atrybutów ruchu generowanych rozwiązań bazowych.

---

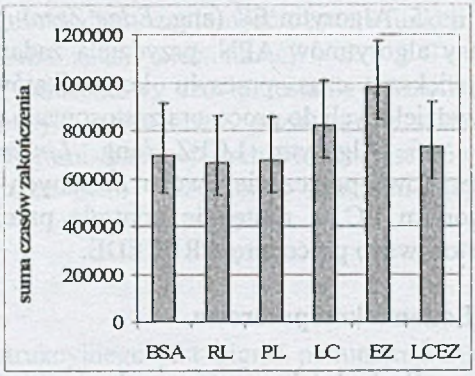
[1] *CCR – (ang. Communication to Computation Ratio)* – iloraz średniego czasu przesyłu komunikatów do średniego czasu wykonania

Na wykresach, rysunki 1 – 8, słupki przedstawiają wartości średnie, natomiast odcinki pionowe przylegające do słupków zakresy odchyleń od wartości średniej wskaźników jakości rozwiązań wyznaczonych przez badane algorytmy, ze względu na następujące kryteria:

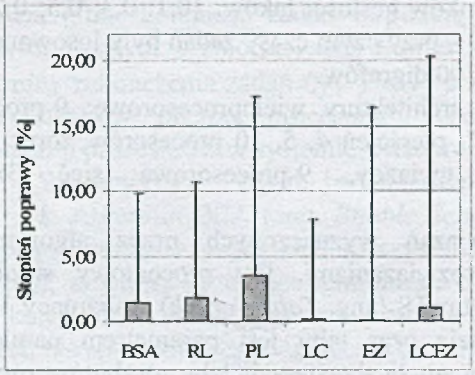
1. Średni termin $C_{max} = \max_j C_j$ zakończenia zadań (rys. 1);
2. Średnia suma terminów $\sum_j C_j$ zakończenia zadań (rys. 2);
3. Procentowa poprawa rozwiązań przez algorytm popraw TS (rys.3);
4. Efektywność algorytmów definiowana jako stosunek przyspieszenia programu do liczby zastosowanych procesorów w wyznaczonym rozwiązaniu (rys.4);
5. Liczba najlepszych rozwiązań wyznaczonych przez badane algorytmy dla każdego diagrafu (rys. 5);
6. Liczba najgorszych rozwiązań wyznaczonych przez badane algorytmy dla każdego diagrafu (rys. 6);
7. Całkowity czas trwania obliczeń wykonany przez poszczególne algorytmy konstrukcyjne dla wszystkich badanych diagrafów (rys.7);
8. Przyspieszenie programu (ang. *speed - up*) (rys. 8).



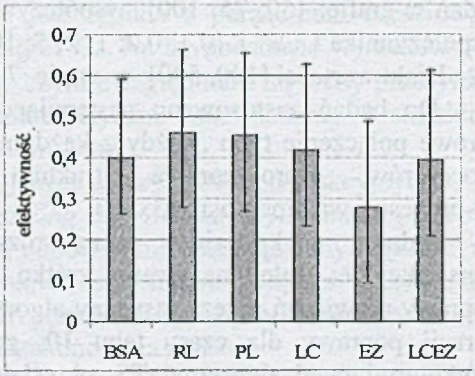
Rys. 1. Czasy zakończenia zadań



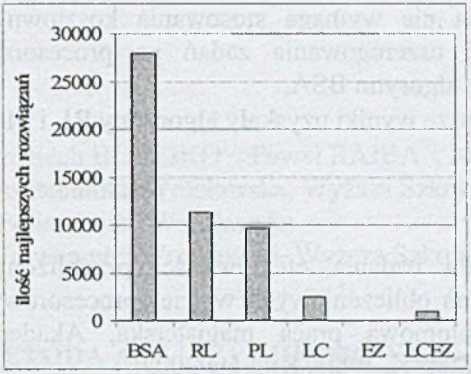
Rys.2. Suma czasów zakończenia zadań



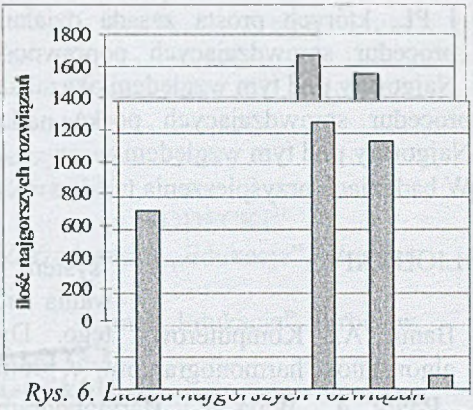
Rys. 3. Procentowa poprawa przez alg. TS



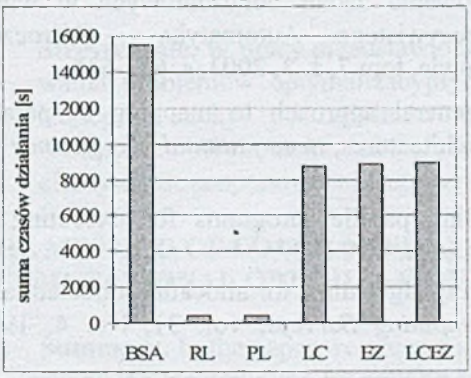
Rys.4. Efektywność algorytmów



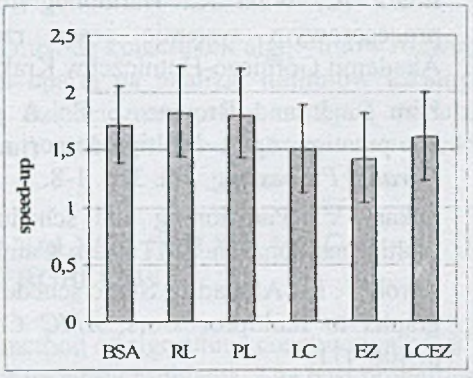
Rys. 5. Liczba najlepszych rozwiązań



Rys. 6. Liczba najgorszych rozwiązań



Rys. 7. Suma czasów działania



Rys. 8. Przyspieszenie programu

5. Zakończenie

Przedstawione powyżej wyniki badań algorytmów konstrukcyjnych, wykonane dla szerokiego spektrum architektur i klas digrafów, umożliwiają sformułowanie następujących wniosków:

- Algorytmy RL, PL i LCEZ generują najlepsze rozwiązania pod względem całkowitego terminu zakończenia zadań i sumy terminów zakończenia zadań.
- Najlepiej poprawianymi rozwiązaniami przez algorytm TS są rozwiązania wyznaczone przez algorytm PL.
- Największą efektywność, mierzoną rozproszeniem zadań obliczeniowych w systemie wieloprocesorowym, zapewniają algorytmy RL i PL. Algorytm BSA, w którego konstrukcji założona jest migracja zadań, nie wyróżnił się w tym zakresie w przeprowadzonych eksperymentach.
- Algorytmy RL i PL, z jednej strony nie wyznaczają rozwiązań najgorszych, ale za to, z drugiej strony, dużą liczbę zadań najlepszych. Algorytm BSA wyznaczał największą liczbę rozwiązań najlepszych, ale i dużą liczbę zadań najgorszych. Zauważmy także, że wprawdzie algorytmy LC i EZ wyznaczały dużą liczbę rozwiązań najgorszych, ale ich hybryda, algorytm LCEZ, już znaczenie mniejszą.
- Ważnym kryterium dla systemów czasu rzeczywistego jest czas wyznaczania rozwiązań. Biorąc pod uwagę to kryterium, najlepszymi okazały się algorytmy RL

i PL, których prosta zasada działania nie wymaga stosowania kosztownych procedur sprawdzających poprawność uszeregowania zadań na procesorach. Najgorszy pod tym względem okazał się algorytm BSA.

- W badaniach przyśpieszenia także najlepsze wyniki uzyskały algorytmy RL i PL.

BIBLIOGRAFIA

1. Banul A.: Komputerowy system do badania efektywności przybliżonych algorytmów harmonogramowania zadań obliczeniowych w wieloprocesorowym systemie czasu rzeczywistego, Dyplomowa praca magisterska, Akademia Górniczo-Hutnicza w Krakowie, promotor: K. Wala, Kraków 2007.
2. Banul A., Wala K.: Harmonogramowanie zadań obliczeniowych w wieloprocesorowym systemie czasu rzeczywistego. Automatyka – Półrocznik Akademii Górniczo-Hutniczej w Krakowie, tom 7, z. 3, 2003, s. 627-635.
3. Kim S. J. and Browne J. C.: A general approach to mapping of parallel computation upon Multiprocessor architectures, *International Conference on Parallel Processing*, vol. 3, p. 1-8.
4. Sakara V.: Partitioning and scheduling parallel programs for executing on multiprocessors, The MIT Press, Cambridge, MA 1989.
5. Kwok Y. K., Ahmad I.: Static scheduling algorithms for allocating directed tasks graphs to multiprocessors, *AMC Computing Surveys*, vol. 31, No. 4, 1999, p. 406-471.

Recenzent: Prof. dr hab. inż. Jerzy Józefczyk

Abstract

A formal model of static scheduling problem of dependent computational tasks in homogeneous multiprocessor system is presented. The dependent computational tasks are modeled by acyclic weighted task precedence degraph $G = (V, E)$, where V is a task set, E describes the precede relation in set V and functions p: $V \rightarrow R^+$, a: $E \rightarrow R^+$ define the mean task execution and message transmission time, respectively. We give a description of six constructive scheduling algorithms for schedule calculation before application software execution by means of the algorithms of APN (Arbitrary Processors Network) class. Taking into account a number of important efficiency criterions, we picture the results of computational investigations of their performance. The computational results are discussed at the end of the paper.