ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1998
Seria: AUTOMATYKA z. 123 Nr kol. 1389

Włodzimierz GŁAZEK
Politechnika Gdańska

SZEREGOWANIE ZADAŃ JEDNORODNYCH W HIPERKOSTKACH DLA MODELU KOMUNIKACJI JEDNOPORTOWEJ

Streszczenie. W pracy rozważany jest problem szeregowania zadań jednorodnych w systemach wieloprocesorowych z siecią połączeniową o strukturze hiperkostki. Celem jest określenie podziału zadania na mniejsze części oraz przesłanie ich do poszczególnych procesorów w sposób, który minimalizuje łączny czas komunikacji i przetwarzania zadania w systemie.

SCHEDULING DIVISIBLE LOADS ON HYPERCUBES IN ONE PORT COMMUNICATION MODEL

Summary. We study the problem of scheduling divisible loads in multiprocessor systems with hypercube interconnection topology. The goal is to partition a load into fractions and communicate them to processors in such way so that the entire load is processed in the shortest possible time.

1. Wstęp

W literaturze poświęconej zagadnieniu szeregowania zadań w systemach wieloprocesorowych dużo wyników dotyczy modeli, w których przyjęto założenie o zerowym koszcie komunikacji pomiędzy procesorami. Wiadomo jednak, że założenie to nie znajduje dobrego odbicia w praktyce inżynierskiej. Tematem tej pracy są algorytmy szeregowania zadań w modelu systemu, w którym przesyłanie danych pomiędzy procesorami wprowadza opóźnienie komunikacyjne. Narzędziem modelowania będzie pojęcie zadania jednorodnego sformułowane przez Chenga i Robertazziego w pracy [3]. Zadanie *jednorodne* jest abstrakcyjnym odpowiednikiem rzeczywistych zadań obliczeniowych o wysokiej ziarnistości. Charakteryzuje się ono ciągłą podzielnością na części oraz brakiem zależności logicznych pomiędzy częściami, dzięki czemu każda z nich może być przetwarzana osobno i niezależnie od pozostałych. Naszym celem jest określenie podziału zadania na mniejsze części oraz przesłanie ich do poszczególnych procesorów w sposób, który minimalizuje *łączny* czas komunikacji i przetwarzania zadania w systemie o zadanej strukturze połączeniowej oraz

wybranym modelu komunikacyjnym przy uwzględnieniu mocy obliczeniowej procesorów i opóźnień czasowych transmisji danych pomiędzy procesorami.

Nietrudno jest wskazać obszary zastosowania koncepcji zadań jednorodnych. Rozważmy plik tekstowy złożony z wielu tysięcy słów. W celu wyszukania pożądanego wzorca w sposób rozproszony możemy podzielić plik pomiędzy procesory systemu z ziarnistością, która jest mała w porównaniu z rozmiarem pliku. Można wtedy uznać, że plik jest zadaniem jednorodnym. Zadania jednorodne dobrze modelują duże wolumeny danych pomiarowych poddawane filtrowaniu, FFT itp. Podobne podejście ma również zastosowanie w przetwarzaniu informacji uzyskanych poprzez obserwacje satelitarne: dane pochodzące ze stacji orbitalnych przesyłane są do stacji naziemnych, gdzie z kolei następuje ich podział i przesłanie do ośrodków obliczeniowych w celu ich przetworzenia. Ośrodki mogą być zlokalizowane w różnych punktach kuli ziemskiej, a opóźnienia komunikacyjne związane z przesyłem danych będą miały istotny wpływ na całkowity czas przetwarzania. Mówiąc ogólnie, proponowane podejście znajduje zastosowanie do rozwiązania problemu równoważenia obciążeń w systemach równoległych i rozproszonych.

W pracy przedstawiamy dwa algorytmy szeregowania zadań jednorodnych w systemach wieloprocesorowych z siecią połączeniową o topologii hiperkostki. Wcześniejsze metody rozwiązania tego problemu były oparte na wielu podejściach: dystrybucji w kolejności rosnących odległości odbiorców [2,4], dystrybucji z „antypodów" [4], podziale rekurencyjnym sieci [4]. We wszystkich metodach przyjęto założenie, że każdy procesor może równocześnie komunikować się ze wszystkimi sąsiadami. Niestety, taki model komunikacyjny jest czuły na operację skalowania sieci połączeniowej, ponieważ wymaga, aby liczba jednocześnie dostępnych portów komunikacyjnych rosła wraz z rozmiarem sieci. Jest to raczej trudne do osiągnięcia w rzeczywistych systemach. Dlatego proponowane tutaj dwa algorytmy są oparte na modelu komunikacji jednoportowej. Algorytmy nawiązują do wyników uzyskanych wcześniej dla analogicznego problemu w systemach kratowych [4,5]. Początkowo procesory systemu grupowane są w zbiory według określonej reguły, następnie części zadania rozsyłane są do tych zbiorów w celu ich przetworzenia, przy czym pierwszy algorytm rozdziela części zadania do procesorów w kolejności rosnących odległości zbiorów, a drugi w kolejności malejącej liczebności zbiorów.

2. Model systemu

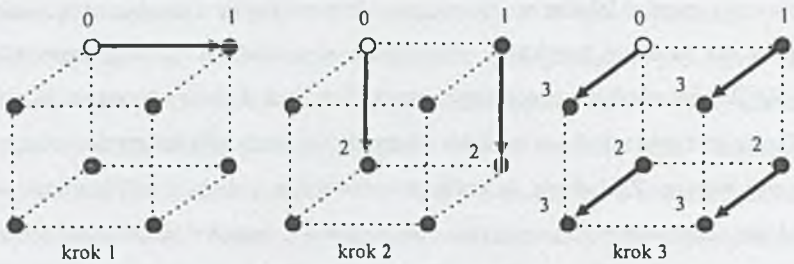
System wieloprocesorowy składa się z N elementów przetwarzających, z których każdy zawiera procesor, pamięć lokalną oraz koprocesor komunikacyjny z przyłączem do sieci. Sieć połączeniowa ma strukturę hiperkostki. *Hiperkostka* d-wymiarowa H_d (ang. hypercube) jest grafem o $N = 2^d$ wierzchołkach oznaczonych przez 2^d różnych d-bitowych ciągów binarnych w taki sposób, że dwa wierzchołki są sąsiednie wtedy i tylko wtedy, gdy ich etykiety różnią się na jednej pozycji bitowej. Zakładamy, że każdy procesor może w danej chwili komunikować się tylko z jednym procesorem, przy czym czas transmisji porcji danych a jest równy $S + Ca$, gdzie S jest czasem inicjalizacji transmisji, a C jest odwrotnością prędkości przesyłu danych. Natomiast czas przetwarzania porcji danych a wynosi Aa, gdzie A jest odwrotnością prędkości przetwarzania pojedynczego procesora. Symbolem V oznaczać będziemy wielkość zadania, które początkowo umieszczone jest w pamięci wybranego procesora zwanego dalej *inicjatorem*. Symbolem a_k, $0 \le k \le d$, oznaczać będziemy część zadania przeznaczoną do przetwarzania na jednym procesorze należącym do k-tego zbioru.

3. Klasyczny algorytm rozsyłania danych w hiperkostce H_d

Przedstawiony dalej proces rozdziału porcji danych jest modyfikacją klasycznego algorytmu dla problemu przesłania różnych komunikatów identycznej długości z wyróżnionego procesora do pozostałych procesorów w hiperkostce H_d, por. [7]. W 1 kroku tego algorytmu inicjator przesyła 2^{d-1} komunikatów do swego sąsiada przez łącze 1 wymiaru. W następnym k-tym kroku tego algorytmu, $2 \le k \le d$, każdy procesor, który już otrzymał dane, wysyła 2^{d-k} komunikatów do sąsiada przez łącze k-go wymiaru. Grupa procesorów otrzymująca dane w k-tym kroku algorytmu tworzy zbiór zwany *warstwą* o liczebności 2^{k-1}. Na rysunku 1 przedstawiony jest proces rozsyłania komunikatów w hiperkostce H_3.

Adaptacja tego algorytmu dla naszych potrzeb wymaga zmian związanych ze specyfiką problemu. Zauważmy, że w powyższym zagadnieniu każdy procesor musi otrzymać ustalony komunikat. Ponadto, wszystkie komunikaty mają równą długość, a za kryterium optymalności przyjęty jest czas rozsyłania komunikatów. W naszym problemie dane nie są adresowane do ustalonych odbiorców. Co więcej, nie jest ważne to, aby wszystkie procesory sieci uczestniczyły w obliczeniach, lecz raczej to, aby całe zadanie było przetworzone tak szybko, jak to tylko możliwe. Zmienione kryterium optymalizacji wymaga nowego określenia podziału

całości danych inicjatora na części (niekoniecznie równe) oraz ustalenia kolejności ich przesłania do odbiorców. Będzie to przedmiotem kolejnych dwóch rozdziałów tej pracy.



Rys.1. Jednoportowy algorytm rozsyłania komunikatów w H_3
Fig.1. 1-port scattering algorithm in H_3

4. Algorytm NLF

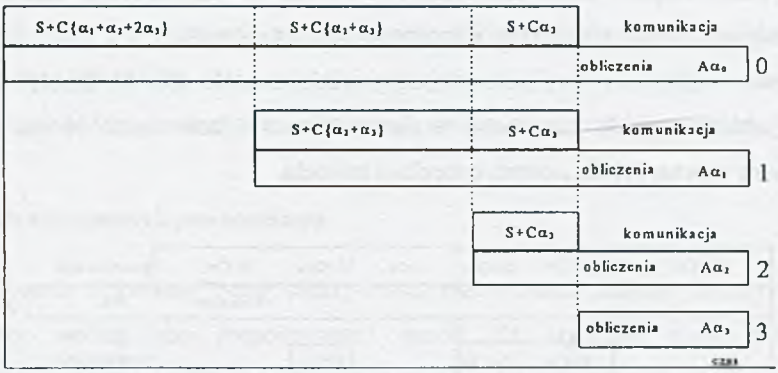
Algorytm NLF rozsyła części zadania do procesorów w oparciu o opisany w rozdziale 3 schemat komunikacyjny. Reguła NLF zakłada, że porcje danych będą przesyłane do procesorów w kolejności rosnących numerów warstw (ang. nearest layer first). Innymi słowy, w k-tym kroku tego algorytmu, $1 \le k \le d$, każdy procesor warstwy k otrzymuje dane, z których część zatrzymuje do przetwarzania lokalnego, resztę zaś dzieli pomiędzy swych następców i wysyła według zasady: w kroku $k+1$ dane do warstwy $k+1$, w kroku $k+2$ dane do warstwy $k+2$ itd. Bez utraty ogólności naszych rozważań przyjmijmy, że wyniki przetwarzania nie są przesyłane z powrotem do inicjatora. Wiadomo, że w optymalnym podziale zadania wszystkie procesory powinny zakończyć pracę w tej samej chwili [3]. Przykładowy wykres czasowy procesu komunikacji i przetwarzania zadania w hiperkostce H_3 według algorytmu NLF przedstawiony jest na rysunku 2.

Uwzględniając powyższe założenia, czas obliczeń inicjatora równy jest sumie czasu transmisji do warstwy 1 danych adresowanych do warstw 1, 2, ..., d oraz czasu obliczeń w warstwie 1. Analogicznie, czas obliczeń na procesorze warstwy $k-1$ równy jest sumie czasu transmisji do warstwy k danych przeznaczonych dla warstw $k, k+1, ..., d$ oraz czasu obliczeń w warstwie k. Stąd, dla każdego k, $0 \le k \le d$, porcja danych a_k przetwarzana na jednym procesorze w warstwie k może być wyznaczona z następującego układu równań

$$Aa_{d-1} = S + (A + C)a_d$$

$$Aa_{d-k} = S + Aa_{d-k+1} + C\left(a_{d-k+1} + \sum_{j=2}^{k} 2^{j-2} a_{d-k+j}\right); \quad k = 2,3,\ldots, d$$

$$V = a_0 + \sum_{k=1}^{d} 2^{k-1} a_k \quad ; \quad a_0, a_1, \ldots, a_d > 0. \tag{1}$$



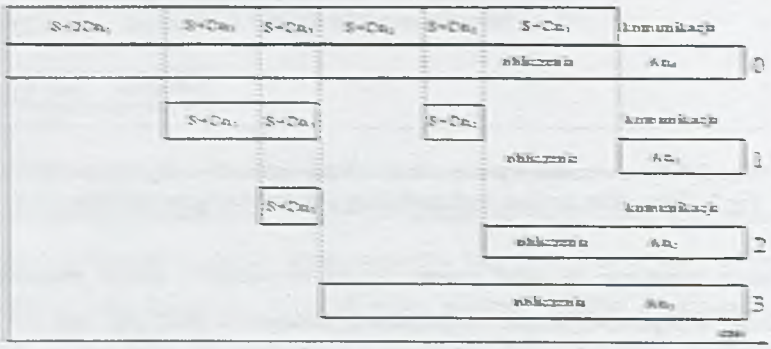
Rys.2. Wykres czasowy procesu komunikacji i obliczeń w algorytmie NLF
Fig.2. Time chart for data communicating and processing in algorithm NLF

Należy zaznaczyć, że układ równań (1) jest identyczny z opisem podziału zadania jednorodnego w p-portowej kracie 3-wymiarowej, podanym w pracy [4], przy $p = 1$. Układ równań (1) może nie mieć rozwiązania dla każdej wartości parametru d. Taka możliwość zachodzi w sytuacji, gdy zadanie ma stosunkowo niewielki wolumen, a prędkość transmisji danych jest mała w stosunku do prędkości przetwarzania. Może się wtedy okazać, że czas konieczny na przesłanie porcji danych do przetwarzania w dalej położonych warstwach nie jest zrekompensowany przez przyspieszenie związane z takim przetwarzaniem. Musimy wtedy zrezygnować z części procesorów i przetwarzać dane na procesorach położonych bliżej inicjatora. Oznacza to przyjęcie mniejszej liczby wymiarów d. Znalezienie optymalnej liczby wymiarów jest możliwe poprzez analityczne rozwiązanie modelu lub metodą przeszukiwania binarnego zakresu 1, 2,..., d i podstawiania do układu równań (1).

5. Algorytm LLF

Algorytm LLF również wykorzystuje schemat komunikacji opisany w rozdziale 3. Reguła LLF zakłada, że porcje danych będą przesyłane procesorom w kolejności malejących liczebności warstw (ang. largest layer first). Zatem procesory warstwy k otrzymują dane do

przetwarzania lokalnego dopiero po przesłaniu danych dalszym warstwom $k+1$, $k+2$,..., d. Oznacza to, że w tym algorytmie dane adresowane do kolejnych warstw nie będą łączone podczas komunikacji, lecz zawsze wysyłane osobno. Tak więc podczas pierwszych d kroków komunikacji dane wysłane przez inicjator dotrą do warstwy d, w kolejnych $d-1$ krokach dane wysłane przez inicjator dotrą do warstwy $d-1$ itd. W konsekwencji liczba kroków komunikacji wzrośnie w stosunku do algorytmu NLF z d do $d+d-1+...+1 = d(d+1)/2$. W zamian za to większe warstwy sieci otrzymają szybciej dane do obróbki. Przesyłanie części zadania jednorodnego do procesorów w osobnych transmisjach zostało po raz pierwszy zastosowane w pracy [1] dla systemu o topologii łańcucha.



Rys.3. Wykres czasowy procesu komunikacji i obliczeń w algorytmie LLF
Fig.3. Time chart for data communicating and processing in algorithm LLF

Podobnie jak poprzednio przyjmijmy, że wszystkie procesory kończą pracę równocześnie, a wyniki przetwarzania nie są przesyłane z powrotem do inicjatora. Przykładowy wykres czasowy procesu komunikacji i przetwarzania zadania w hipercubie H_3 według algorytmu LLF przedstawiony jest na rysunku 3. Tym razem czas obliczeń inicjatora równy jest sumie czasu transmisji danych od inicjatora do warstwy d oraz czasu przetwarzania w warstwie d. Podobnie, czas obliczeń na procesorze warstwy $k+1$ dla $1 \le k \le d-1$ równy jest sumie czasu transmisji danych od inicjatora do warstwy k oraz czasu obliczeń w warstwie k. Stąd, dla każdego k, porcja danych a_k przetwarzana na jednym procesorze w warstwie k może być wyznaczona z następującego układu równań:

$$A a_{k+1} = dS + A a_k + C a_k \left(1 + \sum_{i=0}^{d-k-1} 2^i \right)$$

$$Aa_{k+1} = kS + Aa_k + Ca_k\left(1 + \sum_{i=0}^{k-2} 2^i\right) \quad ; \quad k = d-1, d-2, \ldots, 1$$

$$V = a_0 + \sum_{k=1}^{d} a_k 2^{k-1} a_0 \quad ; \quad a_1, \ldots, a_d > 0. \tag{2}$$

Układ równań (2) jest identyczny z opisem innego podziału zadania jednorodnego w p-portowej kracie 3-wymiarowej, podanym w pracy [5], przy $p = 1$. Układ równań (2) może nie mieć rozwiązania dla każdej wartości parametru d. W takiej sytuacji należy przyjąć mniejszą wartość tego parametru, stosując metodę opisaną w poprzednim rozdziale.

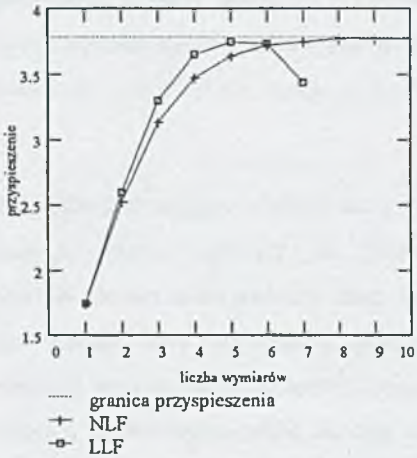
6. Analiza efektywności przetwarzania

Obecnie dokonamy oceny efektywności przetwarzania zadania jednorodnego w hiperkostce według obu proponowanych metod. Za kryterium oceny przyjmiemy *przyspieszenie* obliczeń zdefiniowane jako iloraz czasu przetwarzania zadania na jednym procesorze do czasu przetwarzania w hiperkostce procesorów. Aby osadzić wyniki modelowania na bazie obecnie dostępnych rozwiązań technologicznych, nadamy parametrom komunikacyjnym modelu wartości zmierzone dla systemu wieloprocesorowego o topologii hiperkostki Intel iPSC/2: $S = 700\mu s$ oraz $C = 0.36\mu s$/bajt [6]. Ponadto przyjmiemy $A = 1\mu s$/bajt oraz $V = 10^3$ bajtów. Proces modelowania realizowany jest za pomocą pakietu obliczeniowego MATHCAD.

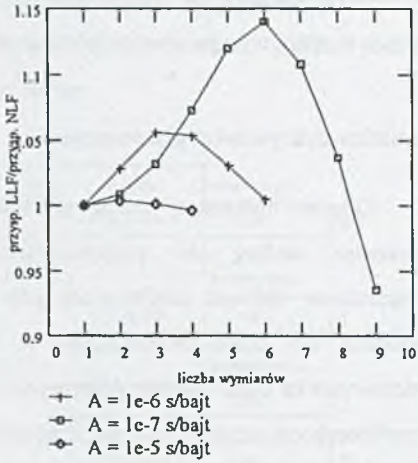
Na rysunku 4 przestawiona jest zależność przyspieszenia od liczby wymiarów d hiperkostki dla algorytmów NLF oraz LLF. Z wykresu wynika, że dla każdego z badanych algorytmów przyspieszenie rośnie wykładniczo względem tej wielkości aż do osiągnięcia poziomu nasycenia. Dalsze zwiększanie liczby wymiarów hiperkostki nie wpływa na wielkość przyspieszenia (algorytm NLF) lub nawet powoduje jego spadek (algorytm LLF). Można zaobserwować, że przyspieszenie algorytmu LLF rośnie szybciej i wcześniej osiąga poziom nasycenia. Z drugiej strony algorytm NLF jest bardziej stabilny w tym sensie, że jego przyspieszenie jest opisane niemalejącą funkcją liczby wymiarów, podczas gdy przyspieszenie algorytmu LLF osiąga swoje maksimum, po czym zaczyna gwałtownie maleć. Ten efekt jest związany z większym kosztem inicjalizacji komunikacji algorytmu LLF.

Na rysunku 5 pokazany jest stosunek przyspieszenia LLF do przyspieszenia NLF w funkcji liczby wymiarów dla trzech różnych prędkości przetwarzania. Pierwsza krzywa ilustruje przebieg badanej funkcji dla prędkości $A^{-1} = 10^6$ bajtów/s, natomiast druga oraz

trzecia krzywa odnoszą się odpowiednio do systemów z 10-krotnie szybszą oraz 10-krotnie wolniejszą prędkością przetwarzania. Można zaobserwować, że algorytm LLF jest tym lepszy, im większa jest prędkość przetwarzania procesorów w stosunku do prędkości komunikacji. Natomiast algorytm NLF ma przewagę w systemach, w których komunikacja jest znacząco wolniejsza od przetwarzania.



Rys. 4. Zależność przyśpieszenia od d
Fig. 4. Dependence of speedup on d



Rys. 5. Przyśpieszenie względne w funkcji A
Fig. 5. Speedup ratio vs. processing rate A

7. Podsumowanie

W pracy przedstawione zostały dwa algorytmy szeregowania dużych zadań jednorodnych w modelu systemu wieloprocesorowego, w którym przesyłanie danych pomiędzy procesorami wprowadza opóźnienie komunikacyjne. Opisano sposób modelowania podziału zadań w systemie za pomocą układu równań rekurencyjnych. Dokonano oceny efektywności proponowanych rozwiązań.

LITERATURA

1. Bharadwaj V., Ghose D., Mani V.: An Efficient Load Distribution Strategy for a Distributed Linear Network of Processors with Communication Delays. Computer and Mathematics with Applications, vol. 29, no. 9, 1995, pp. 95-112.

2. Błażewicz J., Drozdowski M.: Scheduling Divisible Jobs on Hypercubes. Parallel Computing, vol. 21, 1995, pp. 1945-1956.
3. Cheng Y.C., Robertazzi T.G.: Distributed Computation with Communication Delay. IEEE Transactions on Aerospace and Electronic Systems, vol. 24, no. 6, 1988, pp. 700-712.
4. Drozdowski M.: Selected Problems of Scheduling Tasks in Multiprocessor Computer Systems, Wydawnictwo Politechniki Poznańskiej, Poznań 1997.
5. Głazek W.: Distributed Computation in a Three-Dimensional Mesh with Communication Delays, Proceedings 6th Euromicro Workshop on Parallel and Distributed Processing, IEEE Computer Society, January 1998, pp. 38-42.
6. Ho C.-T., Raghunath M.T.: Efficient Communication Primitives on Hypercubes. Journal of Concurrency: Practice and Experience, vol. 4, no. 6, 1992, pp. 427-457.
7. Johnson S.L., Ho C.-T.: Optimum Broadcasting and Personalized Communication in Hypercubes. IEEE Transactions on Computers, vol. 35, no. 9, 1989, pp. 1249-1268.

Recenzent: Prof.dr hab.inż. Andrzej Grzywak

Abstract

In this paper we study scheduling divisible loads in multiprocessor system with hypercube interconnection topology and 1-port communication model. A *divisible* load is characterized by fine granularity, uniform structure and large volume. There are no precedence constraints among data elements. Such a load may be partitioned among processors in the system in an arbitrary way and each part may be then processed independently and separately of other parts. The objective is to find a partition of a load into shares and a distribution of load shares among processors which minimize the processing time of a load subject to data transfer delays. We present two algorithms which differ in the order in which load fractions are distributed to idle processors. Both algorithms rely on the classical method of scattering unit size messages in a hypercube. The scattering algorithm partitions processors in the system into groups, called *layers*, according to the cube dimension along which they are activated. The first presented algorithm distributes load shares to layers in the order of increasing distance from the data source (NLF), the second algorithm scatters load shares in the order of decreasing layer cardinality (LLF). In each case we model the process of load distribution with a set of algebraic equations. Linear modeling of processing time and communication delays leads to a tractable, continuous model of computation which may be analyzed with standard methods of recursive analysis. Finally, we evaluate performance of our algorithms in a series of computational experiments.