ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1994
Seria: ELEKTRONIKA z. 3 Nr kol. 1254

Jerzy KARDASZEWICZ
Instytut Elektroniki
Politechnika Śląska

ANALIZA EFEKTYWNOŚCI PRZETWARZANIA RÓWNOLEGŁEGO W SYSTEMIE WIELOPROCESOROWYM

Streszczenie. W artykule przedstawiono najważniejsze problemy pojawiające się przy oprogramowywaniu systemu wieloprocesorowego oraz zależności pomiędzy uzyskiwaną efektywnością a stosowanym algorytmem rozdziału zadania. Omówiono także wyniki przeprowadzonych eksperymentów.

The analysis of parallel processing effectiveness of a multiprocessor system

Summary. This paper presents the most important problems that appear while writing programs for a multiprocessor system and relations between the effectiveness and the algorithm of distribution of the tasks. The results of experiments are discussed too.

Анализ производительности параллельной обработки информации мультипроцессорных систем

Резюме. Предлагаемая статья относится к наиболее существенным вопросам возникающим во время разработки программ для мультипроцессорных систем, а также к зависимостьям между производительностью и распределением задач. Результаты проведеных опытов оговорены в конечной части статьи

1. WPROWADZENIE

Od początku lat siedemdziesiątych, gdy pojawiły się pierwsze mikroprocesory, do dziś osiągnięto ogromny postęp w tej dziedzinie techniki. Tak dynamiczny rozwój jest odpowiedzią na stale rosnące wymagania użytkowników. Dotyczą one wzrostu mocy przetwarzania danych oraz większej niezawodności systemów. Można wymienić wiele gałęzi nauki, dla których obecne prędkości przetwarzania danych są o wiele za małe w stosunku do potrzeb. Do najważniejszych z nich należą:

- inżynieria genetyczna,
- prognozowanie pogody i globalnych zmian klimatu,
- projektowanie półprzewodników,
- nadprzewodnictwo,

- astronomia,
- transport.

Mówiąc o historycznych motywacjach rozwoju systemów wieloprocesorowych należy wyróżnić dwie linie ewolucji tych struktur. Pierwsza powstawała na skutek potrzeby uwolnienia głównego procesora, z reguły o bardzo dużej mocy przetwarzania, od realizacji zadań we/wy. Zadania te przekazywano mniejszym i lepiej do tego celu dostosowanym jednostkom. W obecnie produkowanych minikomputerach i dużych komputerach znajduje się kilkadziesiąt lub nawet kilkaset procesorów dedykowanych do konkretnych zastosowań (zarządzanie pamięcią, operacjami we/wy itp.). Doszło więc do tego, że trudno jest jednoznacznie odpowiedzić na pytanie o rodzaj i funkcje procesora głównego w takich systemach.

Drugi rodzaj ewolucji był wywołany przez fakt prowadzenia badań nad systemami wieloprocesorowymi przez wojsko. Początkowo głównym celem było uzyskanie większej gotowości operacyjnej dzięki istnieniu zespołu identycznych jednostek. Każda z nich mogła wykonywać te same zadania nawet wtedy, gdy działała tylko część zespołu. Później zdano sobie sprawę, że struktury wieloprocesorowe mogą również poprawić efektywność systemu. Obecnie wszystkie liczące się firmy światowe produkujące duże komputery stosują w swoich produktach kilka do kilkunastu procesorów głównych.

Zagadnienie systemów wieloprocesorowych jest ściśle związane z tzw. problemem wielowątkowości. Wielowątkowość polega na przydzielaniu jednemu przetwarzanemu zadaniu kilku procesorów. Wpływa to w decydujący sposób na uzyskiwaną efektywność tego systemu. Ponieważ otrzymanie zadowalających rezultatów przy rozdziale zadania na poszczególne procesory nie jest zagadnieniem trywialnym, dlatego nie notujemy szybkiego rozwoju tej dziedziny nauki.

W niniejszej pracy przedstawiono wyniki badań efektywności systemu wieloprocesorowego rozwiązującego konkretne zadania obliczeniowe z wykorzystaniem wielowątkowości.

2. PRZEDSTAWIENIE WYKORZYSTYWANEGO SYSTEMU WIELOPROCESOROWEGO

Wieloprocesor, który był wykorzystywany w eksperymentach, można nazwać procesorem typu MIMD (Multiple Instruction stream Multiple Data stream) [5]. W skład systemu może wchodzić do ośmiu procesorów - jeden Master i siedem procesorów Slave. Jednostki centralne, zarówno Master, jak i Slave, stanowią ośmiobitowe CPU z mikroprocesorami 8085. Każdy z procesorów jest wyposażony we własną pamięć lokalną i własny układ dostępu do magistrali globalnej (pamięci globalnej). Komunikacja z pamięcią globalną wymaga uzyskania zezwolenia od, wspólnego dla wszystkich procesorów, układu arbitrażu. Procesor Master może komunikować się z procesorami Slave tylko za pośrednictwem pamięci globalnej. Tam też umieszcza informacje dotyczące rodzaju zadania do wykonania oraz adresy argumentów. Następnie, wykorzystując mechanizm przerwań, uruchamia podłączone do systemu procesory Slave, które rozpoczynają realizację przydzielonych im zadań.

Szczegółowy opis wykorzystywanego systemu wieloprocesorowego można znaleźć w [6].

Przy tworzeniu oprogramowania przyjęta została koncepcja swoistego "zdalnego sterowania" systemu wieloprocesorowego przez system nadzorczy. Jako system nadzorczy został

wybrany komputer klasy IBM PC. Przyjęto założenie, że tam właśnie przygotowywane będą dane i zadania do wykonania. Łącze między wieloprocesorem a systemem nadzorczym zrealizowano wykorzystując interface szeregowy RS232C.

Całe oprogramowanie systemu składa się z dwóch części:

- oprogramowania systemu nadzorczego,
- oprogramowanie systemu wieloprocesorowego.

Zadaniem pierwszego z nich jest stworzenie środowiska pozwalającego na przygotowanie danych i uruchamianie wcześniej napisanych aplikacji w wieloprocesorze. Natomiast oprogramowanie systemu wieloprocesorowego ma na celu realizację konkretnych zadań obliczeniowych. Napisane oprogramowanie zostało dokładnie omówione w [1]. Potrzeby niniejszej pracy wymagały jednak jego znacznej modyfikacji - szczególnie oprogramowanie systemu wieloprocesorowego.

3. BADANIE EFEKTYWNOŚCI SYSTEMU WIELOPROCESOROWEGO

Dla zapoznania się z problemem efektywności wieloprocesora należało wybrać jakieś zadanie, a następnie opracować algorytm jego rozwiązania na danym systemie. Oczywiście już sam wybór rodzaju zadania miał wpływ na później uzyskiwane rezultaty. Przyjęto, że będzie to rozwiązywanie układu równań liniowych metodą eliminacji Gaussa. Trzeba zaznaczyć, że decyzja o wyborze tego zadania ma dwie poważne zalety. Po pierwsze, z problemem rozwiązywania równań liniowych często spotykamy się w praktyce. Wiele zjawisk fizycznych opisuje się układami równań różniczkowych. Jedna z częściej stosowanych metod ich rozwiązywania polega na sprowadzeniu układu do postaci kilku równań różniczkowych rzędu pierwszego. Dla obliczenia całki szczególnej w tej metodzie konieczne jest rozwiązanie układu równań liniowych. Innym przykładem mogą być symulatory do analizy układów elektronicznych (np. program SPICE), które także muszą rozwiązywać układy równań liniowych. Druga zaleta - to stosunkowa łatwość opracowania algorytmu wykorzystującego równoczesną pracę kilku procesorów.

Założeniem było, aby stworzony program potrafił rozwiązywać układy równań liniowych z kilkunastoma niewiadomymi. Taka liczba daje już wymierne rezultaty czasowe. Rozwiązywanie większych układów równań było o tyle bezcelowe, że nie posiadano konkretnych przykładów, a w pracy chodziło o stworzenie samej idei algorytmu rozwiązania zadania. Argumenty równań powinny być zapisywane w standarcie zmiennoprzecinkowym, przy czym ich długość miała wynosić 16 bajtów. Pierwszy bit odpowiada za znak argumentu (Z), 11 kolejnych to cecha (C), a reszta - 14.5 bajtów przypada na mantysę (M). Format stosowanych liczb zmiennoprzecinkowych jest pokazany poniżej:

$$L = (-1)^Z \times 1.M \times 2^{C-1023}$$

Odpowiada on rozszerzonemu formatowi zgodnemu ze standardem IEEE. Przyjmując, że maksymalną liczbą niewiadomych będzie 15, co daje 240 elementów w układzie równań, można obliczyć, iż same argumenty zajmą prawie 4kB obszaru pamięci (3840 bajtów). Liczba 15 została

przyjęta dlatego, że, jak się później okazało, dla niej jeszcze wystarczał jeden ośmiobitowy rejestr przy obliczaniu adresów.

Program musiał uwzględniać ponadto możliwość zmiany konfiguracji i potrafić odpowiednio rozdzielić zadanie. Najważniejszym problemem było opracowanie algorytmu rozdziału i wykonania zadania. Biorąc pod uwagę założony sposób rozwiązywania układów równań, należało zastanowić się, jak optymalnie dzielić zadanie na części przydzielane poszczególnym procesorom. Można tu przedstawić trzy następujące koncepcje:

1) Pierwsza opiera się na pomyśle wykorzystywania procesora Master tylko jako administratora systemu. Znaczy to, że miałby on jedynie obliczać adresy kolejnych elementów macierzy, nad którymi operacje arytmetyczne wykonywałyby Slave'y. Dzięki temu można by zlecać procesorom Slave niemalże elementarne operacje, przykładowo: podzielenie przez siebie dwóch argumentów i dodanie trzeciego. Taki rozdział zadania dałby częste, ale stosunkowo krótkie odwołania do pamięci globalnej. Wtedy prawdopodobnie obciążenie magistrali globalnej, która jest wąskim gardłem tego systemu, byłoby mniejsze, a co za tym idzie, wzrosłaby efektywność całego systemu. Metoda ta ma jednak także poważne wady. Po pierwsze, obliczenie adresu zajmuje mniej czasu niż wykonanie operacji arytmetycznej. W takim razie zysk z zastosowania jednego Slave'a byłby mały. Stosowanie większej liczby procesorów Slave powoduje natomiast znaczne skomplikowanie algorytmu obliczania kolejnych adresów przez Master'a. Dodatkowe wady takiego rozumowania wynikają ze specyfiki założonego zadania. Mianowicie, dla eliminacji kolejnych elementów danego wiersza trzeba najpierw obliczyć iloraz: pierwszego argumentu z rozpatrywanego wiersza przez pierwszą liczbę z aktualnie pierwszego wiersza. Jest zrozumiałe, że byłoby bardzo nieekonomiczne, aby iloraz ten liczył oddzielnie wszystkie procesory Slave wielokrotnie w przypadku długiego wiersza. Podobny problem wystąpiłby przy obliczaniu wartości niewiadomych z macierzy trójkątnej. Rozpatrując wszystkie powyższe uwagi odrzucono tę koncepcję rozdziału zadania.

2) Drugi pomysł był taki, aby zlecać każdemu z procesorów Slave wykonywanie operacji nad jednym wierszem. Okazuje się jednak, że także i to nie jest dobrym rozwiązaniem. Jeżeli w tym przypadku Master spełniałby tylko funkcje administratora systemu, to byłby on praktycznie bezczynny. Należałoby więc przyjąć koncepcję, że także Master będzie wykonywał obliczenia arytmetyczne. Trzeba by wtedy pogodzić się z faktem, że procesory Slave zawsze oczekiwałyby na rozdział zadania i obliczenie nowych adresów przez Master'a. Ostatecznie można by przyjąć takie rozwiązanie, gdyby nie jeszcze jeden problem. Szczególnie gdy aktywnych jest kilka procesorów Slave, to nie można przewidzieć, czy po jednorazowym przydziale wierszy wszystkie elementy kolumny pod przekątną główną zostały wyzerowane. Trzeba by więc za każdym razem to sprawdzać. Niestety wtedy algorytm rozdziału zadania poważnie by się skomplikował. Przez to wzrósłby także czas oczekiwania procesorów Slave. Dlatego rozpatrzono jeszcze trzecie rozwiązanie.

3) Trzecia koncepcja zakłada, że Master jest zarówno administratorem systemu, jak i wykonawcą zadań. Najpierw sprawdza się, ile układ równań ma wierszy, następnie ile jest dostępnych procesorów. Na koniec każdemu z procesorów przydziela się taką

liczbę wierszy, aby po zakończeniu obliczeń przejść do eliminacji elementów z następnej kolumny. Procesor Master otrzymuje do przetworzenia zawsze przynajmniej o jeden wiersz mniej niż Slave'y. Eliminuje to okresy oczekiwania procesorów Slave na przydział zadania. Poważną zaletą jest także stosunkowo prosty algorytm. Wadą tej metody jest natomiast to, że pewien ciężar obliczania kolejnych adresów został przerzucony na procesory Slave. Czas tracony jest jednak znikomo mały w porównaniu z czasem potrzebnym na wykonanie operacji arytmetycznych w całym wierszu.

Analizując wszystkie powyżej przedstawione możliwości zdecydowano się ostatecznie właśnie na to trzecie rozwiązanie.

4. PRZEDSTAWIENIE WYNIKÓW I ICH ANALIZA

Po napisaniu i uruchomieniu wszystkich algorytmów należało sprawdzić efektywność wieloprocesora. Zrobiono to w ten sposób, że na początku stworzono kilka układów równań z różną liczbą niewiadomych. Niewiadome, w tym przypadku, były znanymi liczbami całkowitymi, co umożliwiło jednocześnie kontrolę poprawności obliczeń. Następnie dokonano pomiaru czasu zużytego na rozwiązanie poszczególnych układów równań przez system wieloprocesorowy, występujący w różnych konfiguracjach. Dla zobrazowania uzyskanych wyników zostaną przedstawione rezultaty otrzymane dla pięciu układów równań. Są to układy o: 6, 7, 10, 11 i 13 niewiadomych. W tabeli poniżej umieszczono wartości zmierzonych czasów. Są one podane w sekundach.

Tabela 1

Czasy [sek] rozwiązywania układów równań przez wieloprocesor znajdujący się w różnych konfiguracjach							
liczba niewiadomych	liczba aktywnych procesorów						
	1	2	3	4	5	6	7
6	**20.49**	13.00	11.59	10.27	9.25	9.07	8.96
7	**30.11**	19.34	14.07	13.90	12.53	11.23	11.21
10	**74.01**	43.25	31.92	28.63	25.27	22.09	21.98
11	**96.21**	56.59	42.64	35.60	30.44	28.68	25.00
13	**149.9**	85.44	60.93	51.26	44.89	39.73	36.87

Jak widać, czas potrzebny na rozwiązanie zadania przez pojedynczy procesor jest dosyć długi (w tabeli wartości pogrubione). Zwiększa się on systematycznie przy dodaniu każdej kolejnej niewiadomej. Wprowadzenie natomiast dodatkowego procesora zmniejsza ten czas w sposób znaczący (około 40%). Przy podłączaniu kolejnych procesorów polepszenie szybkości jest coraz mniejsze.

Dla oszacowania wzrostu efektywności systemu wieloprocesorowego należy wprowadzić pojęcie: współczynnika zwiększania szybkości obliczeń. Jest to iloraz: czasu potrzebnego do

wykonania zadania przez jeden procesor do czasu obliczenia tego samego zadania przez p-procesorów:

$$S_p = \frac{T_1}{T_p}$$

W granicznym przypadku wartość tego współczynnika równałaby się p. Byłoby tak wtedy, gdy system z p-procesorami wykonywałby obliczenia p razy krócej.

Wykorzystując powyższy wzór oraz dane z tabeli, można wyliczyć efektywność wieloprocesora przy zmieniającej się jego konfiguracji i różnej liczbie niewiadomych (rys.1). Jak widać, dla trzynastu niewiadomych obliczenia prowadzone przez 7 procesorów dają ponadczterokrotny wzrost efektywności w porównaniu z czasem samego Master'a. Jednak dla 6 niewiadomych poprawa ta jest niewiele ponaddwukrotna. W idealnym przypadku wzrost powinien być siedmiokrotny.

Przy 6 niewiadomych i 7 pracujących procesorach Master nie wykonuje działań arytmetycznych ani razu. Jego rola ogranicza się do przygotowywania nowych adresów i rozdziału zadania. Wzrost szybkości w porównaniu z sześcioma aktywnymi procesorami jest jednak niewielki. Można więc wyciągnąć wniosek, że Master jest w tym przypadku słabo wykorzystywany. Czyli główny zysk czasowy, jaki można uzyskać w tym algorytmie, wiąże się z równoczesnością obliczeń arytmetycznych.

Dodatkowym potwierdzeniem tego rozumowania jest występowanie, dla każdego z wykresów, nieregularności linii. Zastanawiające jest, że we wszystkich przypadkach już po zmaleniu przyspieszenia obliczeń potem jeszcze raz następuje jego wzrost. Szczególnie jaskrawo widać to w przykładzie z siedmioma i jedenastoma niewiadomymi. Zostało to przedstawione osobno (rys.2), aby nie zaciemniać poprzedniego rysunku. Można zauważyć, że gdy liczba niewiadomych jest niecałe dwa razy większa od liczby aktywnych procesorów:

- siedem niewiadomych i cztery procesory,
- jedenaście niewiadomych, sześć procesorów,

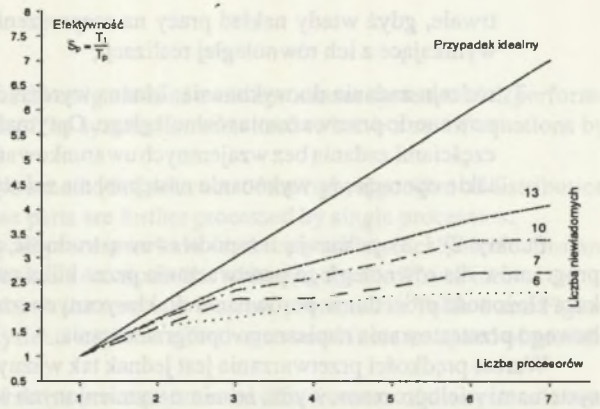
to wzrost efektywności jest mniejszy, niż gdy ten stosunek jest równy około półtora. Wytłumaczenie tego zjawiska jest następujące: gdy powyższy iloraz jest bliski, lecz mniejszy od dwóch, to na początku otrzymywana przez procesor Master liczba wierszy do przetworzenia jest prawie połową tego, co przydzielane jest procesorom Slave. Czas trwania takiej sytuacji jest dłuższy niż w przypadku, gdy ten stosunek jest bliski półtora. Jeżeli przyjąć założenie, że czas obliczania nowych adresów i rozdziału zadania jest znacznie krótszy od skomplikowanych obliczeń arytmetycznych, to powstające nieregularności można wytłumaczyć dłuższym okresem bezczynności procesora Master.

Trzeba także zauważyć, iż wahania wzrostu efektywności dla zadania z jedenastoma niewiadomymi są mniejsze w porównaniu do układu o siedmiu wierszach. Można to tłumaczyć w ten sposób, że im większy jest układ równań, tym więcej potrzeba aktywnych procesorów do uzyskania krytycznego stosunku. Większa liczba procesorów Slave powoduje zajęcie się nimi przez Master'a dłuższy okres czasu, a co za tym idzie, jego krótszą bezczynność. Potwierdzeniem takiego rozumowania może być także wykres efektywności dla zadania o trzynastu niewiadomych (rys.2). Mimo że włączonych jest 7 procesorów, to praktycznie żadna nieregularność nie jest zauważalna.

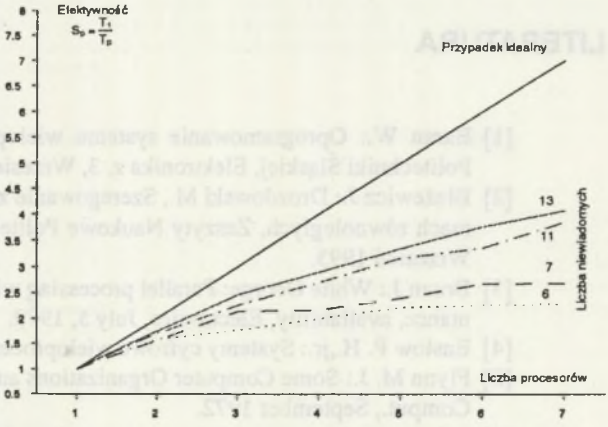
Podsumowując trzeba powiedzieć, że niemożność uzyskania idealnego (liniowego) wzrostu efektywności wieloprocesora jest związana z:

1) występowaniem konfliktów na magistrali systemowej,
2) zastosowanym algorytmem.

Jest właściwie jeszcze trzeci powód, a mianowicie wykorzystywanie układów równań o niewielkiej liczbie niewiadomych. Jak wynika z wykresów (rys.1 i 2), ze wzrostem liczby niewiadomych wzrasta efektywność systemu. Jest to związane z lepszym wykorzystaniem poszczególnych procesorów. Można więc przypuszczać, że w praktycznym zastosowaniu (dla stukilkudziesięciu niewiadomych), uzyskiwane rezultaty byłyby znacznie lepsze.



Rys. 1. Efektywność systemu wieloprocesorowego
Fig. 1. Efeectiveness of the multiprocessor system



Rys. 2. Nieregularność wzrostu efektywności
Fig. 2. Irregularity of the increase in effectivness

5. PODSUMOWANIE

W pracy przedstawiono wyniki uzyskane przy testowaniu konkretnego systemu wieloprocesorowego wykonującego złożone obliczenia arytmetyczne. Pokazują one, że wykorzystanie wieloprocesora pozwala na zwiększenie efektywności i szybkości przetwarzania zadań w porównaniu z klasyczną architekturą von Neumana. Uzyskiwana efektywność zależy głównie od trzech czynników:

1) sprzętu, którym dysponujemy, a więc od szerokości i szybkości magistral systemowych, rodzaju i sposobu działania układu arbitrażu udostępniającego zasoby globalne itp.;
2) algorytmu dzielącego zadanie na części. Optymalną metodą wydaje się tutaj podział zadania na jednostkowe operacje. Operacje te nie powinny być jednak zbyt krótko-

trwałe, gdyż wtedy nakład pracy na rozproszenie obliczeń przewyższyłby korzyści wynikające z ich równoległej realizacji;

3) rodzaju zadania do wykonania. Można wyróżnić zadania bardziej i mniej predysponowane do przetwarzania równoległego. Optymalnie byłoby prowadzić obliczenia nad częściami zadania bez wzajemnych uwarunkowań. Niestety jest bardzo trudno znaleźć takie operacje, że wykonanie następnej nie zależy od wyniku poprzedniej.

Punkty 2) i 3) pokazują też podstawową trudność, z jaką spotykamy się przy pisaniu programów dla równoległego przetwarzania przez kilka procesorów. Jest nią mianowicie większa złożoność procedur w porównaniu do klasycznych rozwiązań. Brak jest także możliwości łatwego przetestowania napisanego oprogramowania.

Wzrost prędkości przetwarzania jest jednak tak ważnym argumentem przemawiającym za systemami wieloprocesorowymi, że mimo wymienionych wad trzeba spodziewać się ich dalszego rozwoju.

LITERATURA

[1] Baran W.: Oprogramowanie systemu wieloprocesorowego, Zeszyty Naukowe Politechniki Śląskiej, Elektronika z. 3, Wrzesień 1994.

[2] Błażewicz J.: Drozdowski M., Szeregowanie zadań wieloprocesorowych w systemach równoległych, Zeszyty Naukowe Politechniki Śląskiej, Informatyka z. 24, Wrzesień 1993.

[3] Braun J.: White George: Parallel processing with minicomputers increases performance, availability, Electronics, July 5, 1979.

[4] Enslow P. H.,jr.: Systemy cyfrowe wieloprocesorowe, WNT, Warszawa 1978.

[5] Flynn M. J.: Some Computer Organizations and Their Effectiveness, IEEE Trans. Comput., September 1972.

[6] Taborek K.: System wieloprocesorowy dla mikroprocesorów ośmiobitowych -rozwiązanie sprzętowe, Zeszyty Naukowe Politechniki Śląskiej, Elektronika z. 3, Wrzesień 1994.

Wpłynęło do Redakcji w czerwcu 1994 r.

Abstract

The paper presents the results of investigations of a multiprocessor system which performs complicated arithmetical instructions. The system has been used to solve a set of equations by Gauss method.

The first part of the article shows a method of selection of the right algorithm of distribution of the tasks into separate parts. These parts are further processed by single processors.

The arguments explaining why all processors have to execute arithmetical instructions are also discussed. The Master processor has to compute the addresses for the Slaves too.

The second part of the article shows the results, which are presented in Table 1 and Figure 1. They prove that multiprocessor systems improve effectiveness and increase speed of processing of tasks.

The influence of the number of active processors on the effectiveness of the system is also analyzed (Fig. 2).

At the end, a list of problems and difficulties that appear while writing programs for multiprocessor systems.