Dariusz CHAŁADYNIAK[1]

PODSTAWY PRZETWARZANIA RÓWNOLEGŁEGO INFORMACJI

Streszczenie

W artykule poruszono wybrane podstawowe zagadnienia związane z przetwarzaniem równoległym. Przedstawiono główne obszary zastosowania przetwarzania równoległego oraz dokonano podziału systemów równoległych za pomocą klasyfikacji zaproponowanej przez M. Flynna. Scharakteryzowano cztery rodzaje przetwarzania: SISD (Single Instruction Stream – Single Data Stream), SIMD (Single Instruction Stream – Multiple Data Stream), MISD (Multiple Instruction Stream – Single Data Stream) oraz MIMD (Multiple Instruction Stream – Multiple Data Stream). Opisano również trzy architektury pamięciowe: MIMD-SM (Shared Memory), MIMD-DM (Distributed Memory) oraz MIMD-HDSM (Hybrid Distributed-Shared Memory). W poniższym opracowaniu przedstawiono ponadto ogólny podział architektury komputerów równoległych. Artykuł kończą informacje na temat rodzajów równoległości przetwarzania danych na przykładzie równoległości homogenicznej i heterogenicznej.

The paper considers selected basic issues related with parallel computing. Main areas of parallel computing application and parallel systems classification according to M. Flynn are presented. Four types of processing are characterized: SISD (Single Instruction Stream – Single Data Stream), SIMD (Single Instruction Stream – Multiple Data Stream), MISD (Multiple Instruction Stream – Single Data Stream) and MIMD (Multiple Instruction Stream – Multiple Data Stream). Three memory architectures are described: MIMD-SM (Shared Memory), MIMD-DM (Distributed Memory) and MIMD-HDSM (Hybrid Distributed-Shared Memory). A general classification of parallel computers architecture is additionally presented. The paper is completed by information concerning types of parallelism of data processing with the example of homogenic and heterogenic parallelism.

1. WPROWADZENIE

Współczesne zapotrzebowanie na moc obliczeniową komputerów rośnie niemal w postępie wykładniczym. Dlatego inżynierowie informatycy projektują różne architektury systemów komputerowych, aby sprostać bieżącym i najbliższym wyzwaniom odnośnie szybkości przetwarzania informacji.

[1] Dr inż. Dariusz Chaładyniak jest wykładowcą w Warszawskiej Wyższej Szkole Informatyki.

Istnieje wiele alternatywnych rozwiązań zwiększania wydajności współczesnych komputerów. Należy do nich również idea przetwarzania równoległego polegająca na podziale programu na fragmenty, z których każdy wykonywany jest przez inny procesor (lub węzeł). Kod podzielony na „kawałki" wykonywane przez *n* procesorów (węzłów), będzie się wykonywał *n*–razy szybciej niż analogiczny kod, realizowany na maszynie jednoprocesorowej.

Idea obliczeń równoległych narodziła się pod koniec lat sześćdziesiątych w okresie wzmożonego zapotrzebowania na moc obliczeniową, a także powstawania coraz bardziej zaawansowanych architektur komputerów. Jednak dopiero w latach osiemdziesiątych gwałtowny rozwój techniki komputerowej umożliwił praktyczną realizację tego sposobu przetwarzania danych.

Główne obszary zastosowania przetwarzania równoległego to:

1. Przemysł – skrócenie czasu projektowania i symulacji, związanych z wprowadzeniem nowych produktów;
2. Nauka – dostarczanie mocy obliczeniowej, pozostającej do tej pory w domenie specjalistów od science fiction;
3. „Grand Challenge Problems" – pogoda i klimat, aktywność geologiczna i sejsmiczna, genom ludzki, reakcje jądrowe;
4. Deep Blue – „pokaz siły" maszyny równoległej nad arcymistrzem Garry Kasparovem.

2. PODZIAŁ SYSTEMÓW RÓWNOLEGŁYCH

Ze względu na architekturę komputery do przetwarzania równoległego można podzielić na poniższe grupy [1]:

- komputery o przetwarzaniu potokowym;
- komputery wektorowe;
- komputery wieloprocesorowe;
- klastry.

Istnieje wiele różnych podziałów przetwarzania równoległego, ale chyba najbardziej znanym jest klasyfikacja zaproponowana w 1966 roku przez M. Flynna. Według niego podstawą klasyfikacji jest liczba strumieni danych i liczba strumieni rozkazów przesyłanych do procesora.

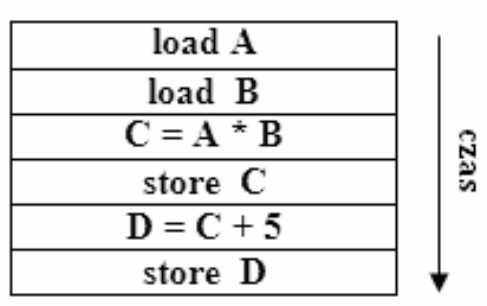
Zgodnie z tą ideą można wyróżnić następujące warianty:

- SISD (Single Instruction Stream – Single Data Stream);
- SIMD (Single Instruction Stream – Multiple Data Stream);
- MISD (Multiple Instruction Stream – Single Data Stream);
- MIMD (Multiple Instruction Stream – Multiple Data Stream).

2.1. SISD (Single Instruction Stream – Single Data Stream)

Cechy charakterystyczne tego typu przetwarzania danych to:

1. Tradycyjna (szeregowa) metoda przetwarzania danych (krok po kroku);
2. Pojedynczy zestaw instrukcji jest wykonywany sekwencyjnie w pojedynczym strumieniu danych;
3. Wykorzystanie komputerów jednoprocesorowych.

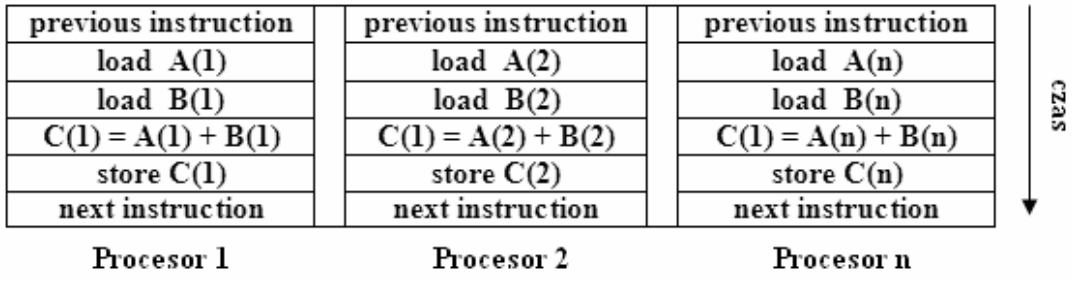


Rys. 1.1. Przykład przetwarzania typu SISD [3].

2.2. SIMD (Single Instruction Stream – Multiple Data Stream)

Cechy charakterystyczne tego typu przetwarzania danych to:

1. Wczesne implementacje systemów równoległych;
2. Każdy z procesorów przetwarza ten sam zestaw instrukcji na własnym zestawie danych;
3. Do układów SIMD zaliczają się procesory macierzowe, wektorowe.



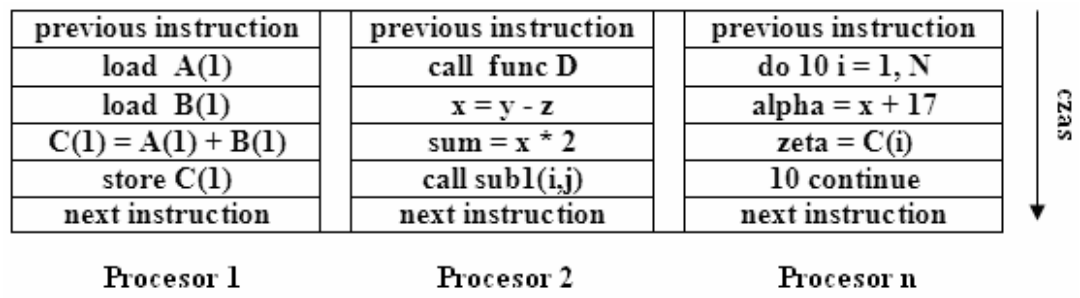
Rys. 1.2. Przykład przetwarzania typu SIMD [3].

2.3. MISD (Multiple Instruction Stream – Single Data Stream)

Cechy charakterystyczne tego typu przetwarzania danych to:
1. Model niedostępny w systemach komercyjnych;
2. Prace nad nim są prowadzone jedynie w ośrodkach naukowych;
3. Może występować w procesorach peryferyjnych systemów CDC–6000.

2.4. MIMD (Multiple Instruction Stream – Multiple Data Stream)

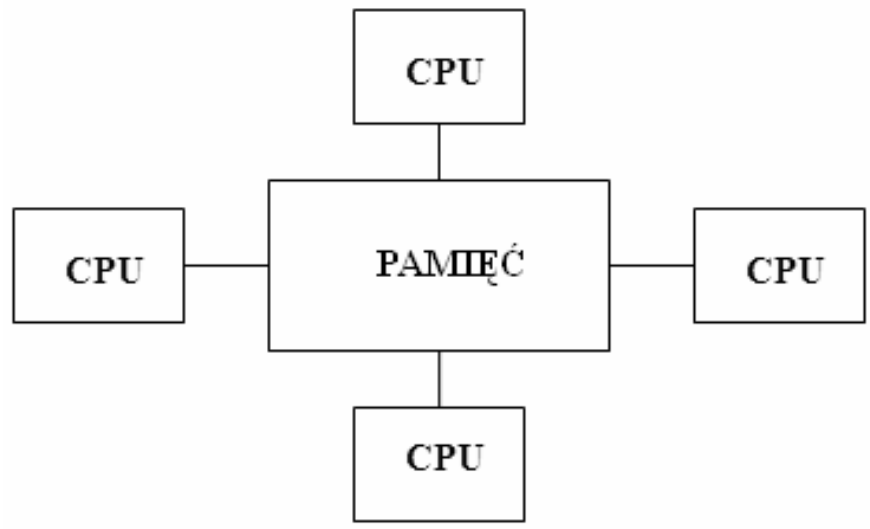
Cechy charakterystyczne tego typu przetwarzania danych to:
1. Najpopularniejsza ze wszystkich implementacji;
2. Każdy procesor pracuje z własnym zestawem instrukcji, operując na własnym zestawie danych;
3. W ramach MIMD wyróżnia się trzy kategorie:
   – MIMD–SM (Shared Memory),
   – MIMD–DM (Distributed Memory),
   – MIMD–HDSM (Hybrid Distributed–Shared Memory).



Rys. 1.3. Przykład przetwarzania typu MIMD [3].

MIMD–SM (Shared Memory)

Cechy charakterystyczne tego typu przetwarzania danych to:
1. Procesory są połączone specjalizowaną siecią interconnect, poprzez którą komunikują się ze wspólnym obszarem pamięci;
2. W ramach systemów SM można wyróżnić dwie kategorie, różniące się podejściem do problemu współdzielenia zasobów:
   – shared everything – komunikacja między procesorami odbywa się poprzez operacje zapisu i odczytu pamięci dzielonej, przy czym wymagane jest, aby wszystkie dane programu były przechowywane we wspólnym obszarze pamięci;
   – shared something – komunikacja między procesorami odbywa się poprzez operacje zapisu i odczytu pamięci dzielonej, przy czym wymaga się od użytkownika wyraźnego zdefiniowania, które dane powinny się w tym obszarze znaleźć.

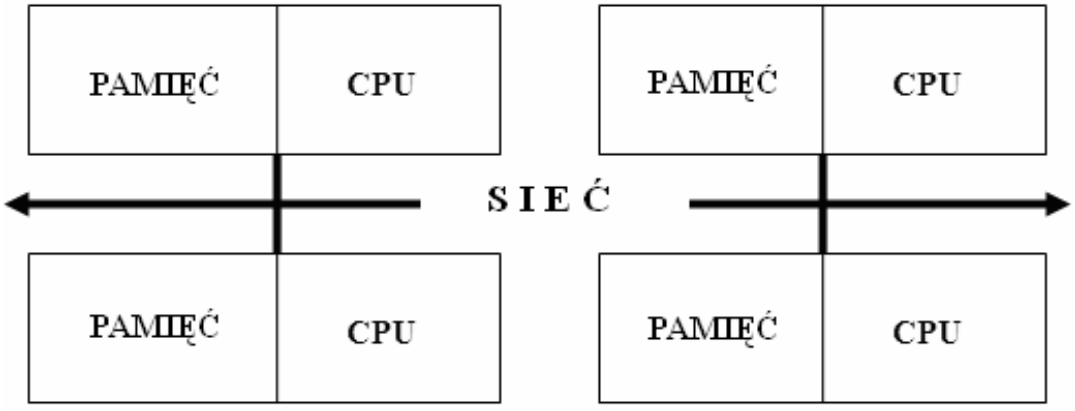


Rys. 1.4. Przykład architektury z pamięcią współdzieloną [3].

MIMD–DM (Distributed Memory)

Cechy charakterystyczne tego typu przetwarzania danych to:

1. W odróżnieniu od systemów SM, pamięć w tym modelu nie jest współdzielona między procesorami;
2. Każdy procesor posiada własny obszar pamięci;
3. Sieć interconnect służy do wymiany komunikatów pomiędzy procesorami.

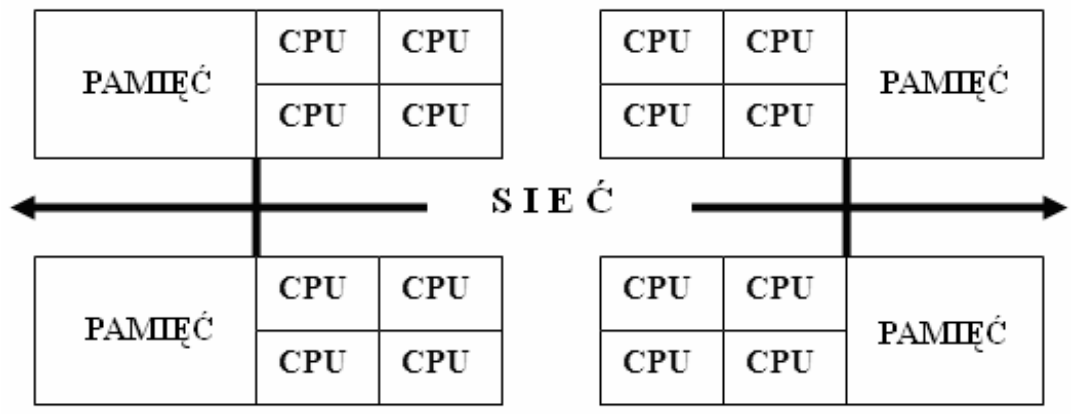


Rys. 1.5. Przykład architektury z pamięcią rozproszoną [3].

MIMD–HDSM (Hybrid Distributed–Shared Memory)

Cechy charakterystyczne tego typu przetwarzania danych to:

1. Pamięć w tym modelu jest zarówno rozproszona jak i współdzielona między procesorami;
2. Wiele procesorów posiada własny obszar pamięci;
3. Sieć interconnect służy do wymiany komunikatów pomiędzy procesorami i pamięcią.

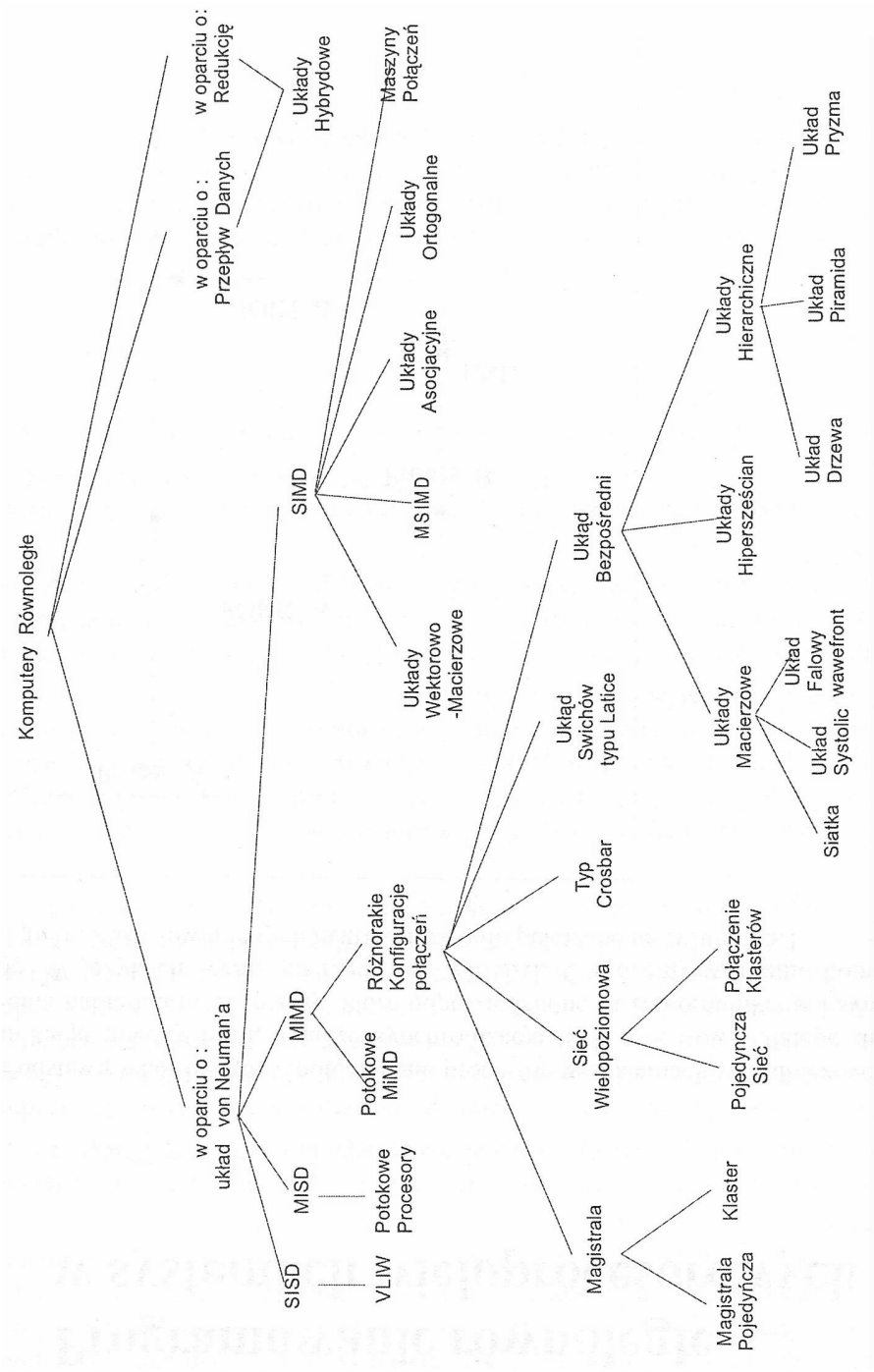


Rys. 1.6. Przykład architektury z rozproszoną pamięcią współdzieloną [3].

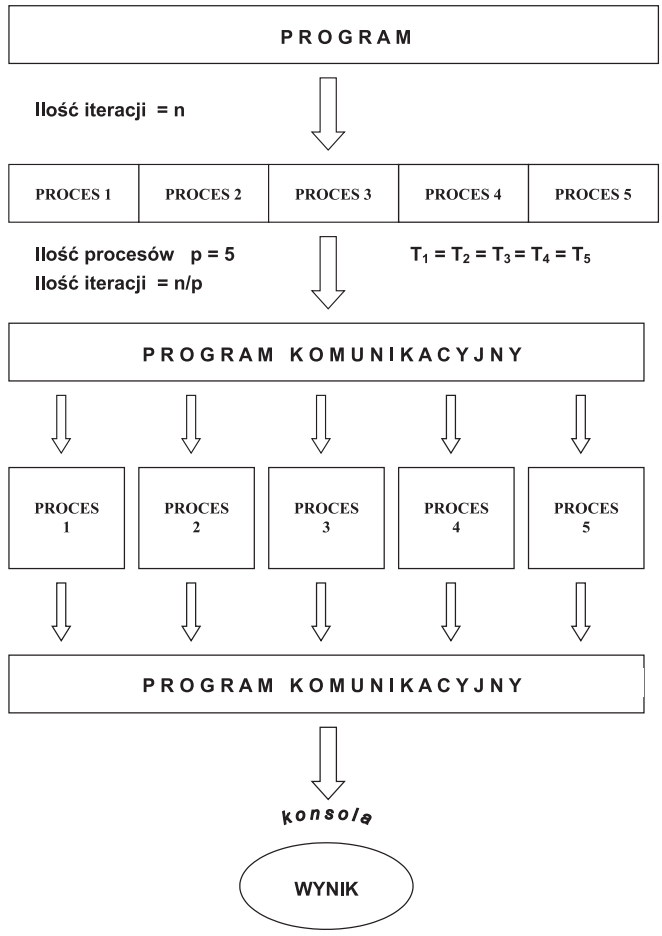
3. RODZAJE RÓWNOLEGŁOŚCI PRZETWARZANIA DANYCH

3.1. Równoległość homogeniczna

Równoległość homogeniczna zachodzi wówczas, gdy zadanie, które ma być wykonane przez dany algorytm programu komputerowego może zostać podzielone na takie same co do wielkości (homogeniczne) podzadania. Każde takie podzadanie stanowi pewną niewielką część całego zadania. Na rysunku 2.1 został przedstawiony schemat blokowy równoległości homogenicznej.



Rys. 1.7. Schemat architektury komputerów równoległych [1].

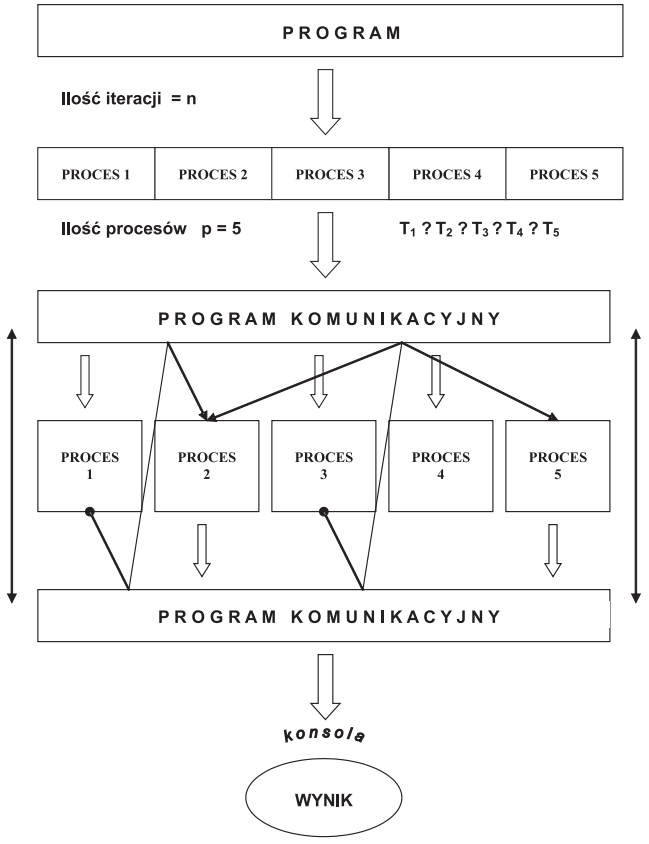


Rys. 2.1. Przykład równoległości homogenicznej [1].

3.2. Równoległość heterogeniczna

Równoległość heterogeniczna zachodzi wówczas, gdy zadanie, które ma być wykonane przez dany algorytm programu komputerowego musi zostać podzielona na różne, co do wielkości (heterogeniczne) zadania. Każde takie podzadanie jest niewielką porcją całego zadania, ale inaczej aniżeli w przypadku równoległości homogenicznej, podzadania mogą być uruchamiane niezależnie od siebie. Będą to najczęściej podprogramy, które od chwili załadowania danych wsadowych aż do instrukcji zakończenia i powrotu, mogą się wykonywać samodzielnie, przetwarzając konkretnymi funkcjami swoje dane. Przy równoległości heterogenicznej może zachodzić zależność danych między poszczególnymi procesami. Oznacza to, że pewne procesy

korzystają z danych przetworzonych przez uprzednio ukończone procesy współbieżne. Na rysunku 2.2 proces 2 czeka na dane z procesów 1 i 3, z kolei proces 5 nie rozpocznie działania, zanim nie zostaną mu dostarczone dane z procesu 3.



Rys. 2.2. Przykład równoległości heterogenicznej [1].

4. PODSUMOWANIE

W artykule poruszono jedynie podstawowe aspekty równoległego przetwarzania informacji. Wydaje się, że właśnie ten rodzaj obróbki danych będzie odgrywał coraz większą rolę w nowoczesnych zastosowaniach. Ponadto należy pamiętać, że na ogólną wydajność systemu komputerowego mają wpływ nie tylko procesory i pamięci, ale również coraz wydajniejsze systemy wymiany danych. Dzięki ciągle dynamicznie rozwijającej się technice światłowodowej oraz technologii Ethernet można projektować rozwiązania mogące zaspokoić obecne i przyszłe zapotrzebowanie na dużą moc obliczeniową. Również odpowiednio zoptymalizowane oprogramowanie systemowe wpływa znacząco na poprawę wydajności przetwarzania dowolnej informacji.

Literatura

[1] Borowik B. E., *Programowanie równoległe w zastosowaniach*, MIKOM, Warszawa, 2001
[2] Buyya R., *High Performance Cluster Computing: Architectures & Systems,* Published by Prentice Hall PTR, 1999
[3] Chaładyniak D., *The role of the cluster servers in parallel computing of hydrometeo–rological data*, XIV–th International Scientific and Technical Conference, The Part of Navigation in Support of Human Activity on the Sea, Gdynia, 2004
[4] Culler D., Singh J. P., Gupta A., *Parallel Computer Architecture: A Hardware/Software Approach,* Published by Morgan Kaufmann, 1998
[5] Dongarra J., Foster I., Fox G., Kennedy K., White A., Torczon L., Gropp W., *The Sourcebook of Parallel Computing*, Published by Morgan Kaufmann, 2002
[6] Komorowski W., *Krótki kurs architektury i organizacji komputerów*, MIKOM, Warszawa, 2004
[7] Vrenios A., *Linux Claster Architecture*, Published by SAMS, 2002