ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1994
Seria: INFORMATYKA z. 26 Nr kol. 1255

Andrzej WILK

KONCEPCJA WYKORZYSTANIA SIECI NETWARE DO REALIZACJI PRZETWARZANIA RÓWNOLEGŁEGO*

Streszczenie. Artykuł prezentuje ideę realizacji algorytmów równoległych w środowisku sieci Netware firmy Novell. Do demonstracji został użyty algorytm równoległy rozwiązania uproszczonego problemu Hilberta. Przedstawiona koncepcja może być traktowana jako praktyczna realizacja systemu sterowanego przepływem argumentów.

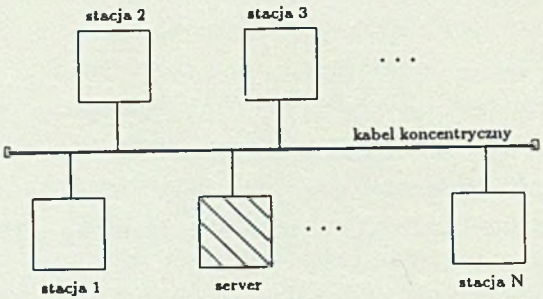
CONCEPTION OF THE NETWARE NETWORK UTILIZATION FOR PARALLEL PROCESSING REALIZATION

Summary. The article describes the idea of parallel algorithm realization in the Novell's Netware network environment. Parallel algorithm of simplified Hilbert's problem resolution is used for the demonstration. Presented conception may be treated as a practical realization of the system controlled by flow of arguments.

CONCEPTION D'UTILISATION DE RÉSEAU NETWARE POUR LA RÉALISATION DE TRAITEMENT PARALLÈLE

Résumé. L'article présente l'idée de réalisation des algorithmes parallèles dans le milieu de réseau Novell – Netware. Algorithme parallèle de résolution de problème simplifie d'Hilbert est utilisé pour la démonstration. La conception présentée peut être considerée comme la réalisation pratique des systèmes controlés par le flux des arguments.

*Praca zrealizowana w ramach Projektu Badawczego (GRANTU) nr KBN 3 P406 011 04



Rys. 1. Schemat sieci Netware
Fig. 1. Netware network scheme

1. Wprowadzenie

Istotą przetwarzania równoległego jest powierzenie wykonania zadania więcej niż jednemu procesorowi. Zadanie dzielone jest na fragmenty, które można wykonać równolegle na niezależnych procesorach. W efekcie osiąga się skrócenie czasu samego przetwarzania, dochodzi jednakże czas przesyłu fragmentów zadania pomiędzy procesorami. Wypadkowy efekt czasowy jest zatem zależny zarówno od organizacji przetwarzania, jak i szybkości przesyłu.

Przetwarzanie równoległe może być powierzone maszynie wieloprocesorowej. Częściej jednak, ze względu na powszechność lokalnych sieci komputerowych, będących de facto zbiorem komputerów – procesorów wykorzystujących w miarę szybkie medium transmisyjne, np. Ethernet, rozważa się możliwość wykorzystania tego środowiska do przetwarzania równoległego, rozproszonego. Niniejszy artykuł opisuje pomysł wykorzystania do tego celu popularnej sieci Netware firmy Novell. Zaproponowane zostały mechanizmy synchronizacji procesów poszczególnych komputerów, wynikające z cech tej sieci. Koordynacja ta jest sterowana wystąpieniami podzadań do wykonania lub uzyskaniem ich wyników. Przygotowane w ten sposób środowisko sieci Netware może być zatem traktowane jako praktyczna realizacja systemu sterowanego przepływem argumentów [1], w którym argumentami są podzadania oraz generowane przez nie wyniki.

2. Sieć Netware

Typowa sieć Netware obejmuje pewną liczbę mikrokomputerów, z których każdy podłączony jest do wspólnego medium transmisyjnego. Aktualnie dość powszechnie do łączenia komputerów stosuje się standard "cienkiego" Ethernetu, opartego na medium w postaci kabla koncentrycznego (rys. 1).

Sieć Netware funkcjonuje w sposób określany w literaturze mianem klient - serwer. Wyróżniony komputer - *serwer*, na którym uruchomiony jest system Netware, pełni rolę składnicy plików, które udostępnia pozostałym, przyłączonym do sieci komputerom -

klientom (stacjom roboczym). W rozwiązaniu tym komunikacja odbywa się jedynie pomiędzy stacją i serwerem, tzn. każda ze stacji "widzi" wspólną pamięć dyskową serwera, natomiast nie "widzi" innych stacji. Wynika stąd, że każda ze stacji może wykonać dopuszczalne przez system operacyjny operacje dyskowe na dysku serwera. Należą do nich:

- sprawdzenie istnienia pliku o określonej nazwie,
- otworzenie pliku o określonej nazwie,
- zmiana nazwy pliku,
- usunięcie pliku o określonej nazwie,
- zapisanie informacji do pliku o wskazanej nazwie,
- odczytanie informacji z pliku o wskazanej nazwie.

System Netware ma również zaimplementowane mechanizmy operacji na semaforach służących do koordynacji procesów. Semafory te są tworzone przez serwera i mają do nich dostęp stacje sieci. Semaforów może być wiele i są one rozróżnialne np. przez nazwę. Operacjami na semaforach są:

- utworzenie semafora,
- próba wykonania operacji *P(S)* na semaforze przez zadany czas. Po tym czasie program rezygnuje z dostępu do zasobu chronionego tym semaforem,
- operacja *V(S)*,
- usunięcie semafora.

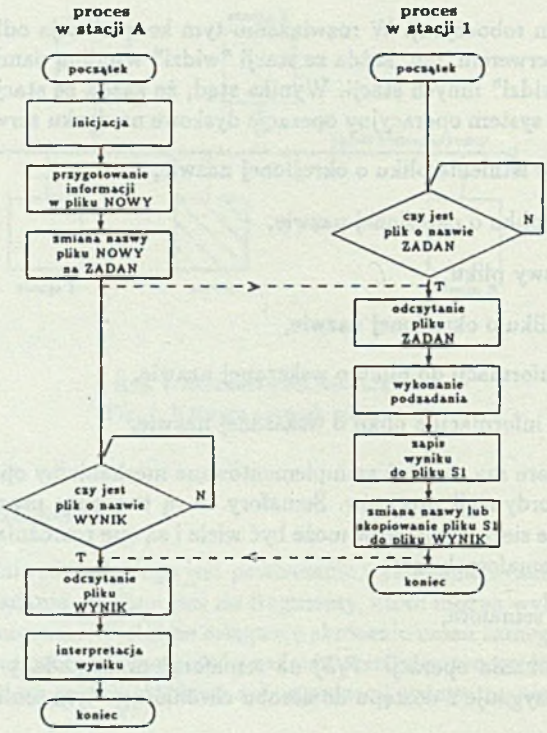
Pliki serwera mieszczące się w jego pamięci dyskowej oraz semafory są zatem elementami wspólnymi, do których mają dostęp stacje sieci. Za pomocą tych elementów można skoordynować procesy biegnące w poszczególnych stacjach sieci.

3. Koordynacja procesów

Współpraca dwóch lub więcej procesów (stacji sieci) polega na przygotowaniu informacji w jednej ze stacji i przesłaniu jej do innej lub innych stacji. Koordynacja polega tu na tym, że proces w stacji odbierającej informację musi być gotowy na jej przyjęcie, jak i musi poczekać, aż zostanie ona przygotowana w stacji wysyłającej. Z kolei stacja wysyłająca musi dać znać stacji odbierającej, że przygotowana informacja jest gotowa do odbioru.

Istnieje wiele wariantów mechanizmów koordynacji procesów różnych stacji, które można utworzyć na bazie dostępnych operacji dyskowych i semaforowych serwera. Dla naszych potrzeb należy rozgraniczyć dwa przypadki:

1. Odbiorcą informacji jest tylko jeden proces. Połączenie to nazywane jest połączeniem dwupunktowym (jeden nadaje – jeden odbiera).



Rys. 2. Koordynacja dwóch procesów
Fig. 2. Coordination of two processes

2. Odbiorcą lub nadawcą informacji jest jeden proces wybrany według pewnych reguł z wielu ubiegających się o nią procesów. Połączenie takie nazywane jest połączeniem wielopunktowym.

Ad 1. Gdy odbiorcą informacji jest tylko jeden proces, to koordynację można przeprowadzić np. przez kreowanie na dysku pliku o odpowiedniej nazwie, na który ten proces czeka (rys. 2).

Naturalnie, żeby pliki *ZADAN* oraz *WYNIK* z rysunku 2 były widoczne przez oba procesy, muszą się znajdować w pamięci dyskowej serwera widzianej przez wszystkie stacje.

Za pomocą połączeń dwupunktowych można połączyć proces stacji *A* z wieloma procesami stacji 1 do *N*. Dla każdego połączenia przewidziana jest indywidualna nazwa pliku, np. spośród *ZADAN*.1 do *ZADAN.N* oraz *WYNIK*.1 do *WYNIK.N*. Wadą rozwiązania jest to, że proces stacji *A* musi "znać" wszystkie procesy stacji 1 do *N*.

Ad 2. Gdy na przesyłaną informację oczekuje wiele procesów (stacji), lecz tylko jeden

może ją otrzymać, to dodatkowo konieczne staje się utworzenie mechanizmu rozstrzygającego konflikt dostępu.

Jednym ze sposobów jest utworzenie dodatkowego procesu, który odbiera przesyłaną informację i przekazuje do wybranego przez siebie jednego z procesów oczekujących. W rozwiązaniu tym z połączenia wielopunktowego tworzy się wiele połączeń dwupunktowych, gdyż w danym momencie dodatkowy proces komunikuje się tylko z jednym procesem oczekującym. Połączenia dwupunktowe realizuje się za pomocą mechanizmu opisanego w punkcie 1 przy użyciu różnych nazw przekazywanych plików indywidualnie dla każdego połączenia.

Innym sposobem na rozstrzygnięcie konfliktu dostępu jest utworzenie sekcji krytycznej, do której w danym momencie ma dostęp tylko jeden proces. Można to zrealizować za pomocą dostępnych w Netware operacji na semaforach, np. w sposób przedstawiony na rysunku 3. Należy zwrócić uwagę, że tylko jedna ze stacji 1 do N może wykonać z sukcesem operację $P(S_{WE})$ (względnie $P(S_{WY})$). Inne procesy nie będą mogły tego uczynić do czasu zwolnienia semafora operacją $V(S_{WE})$ (względnie $V(S_{WY})$).

Główną zaletą wielopunktowego połączenia jest to, że proces stacji A nie musi wiedzieć, jaka liczba N procesów równa liczbie stacji jest uruchomiona. Może ich być dowolna liczba.

Wadą przedstawionego rozwiązania jest np. to, że proces stacji A komunikuje się z procesami stacji 1 do N za pomocą tylko dwóch plików: *ZADAN* – przy wysyłaniu informacji i *WYNIK* – przy odbieraniu wyników. Powoduje to, że proces stacji A nie może wygenerować nowej informacji do pliku *ZADAN* dopóty, dopóki poprzednia (tzn. zawartość poprzedniego pliku *ZADAN*) nie zostanie odebrana, jak również, procesy stacji 1 do N nie mogą odesłać wyniku w pliku *WYNIK* dopóty, dopóki tenże nie zostanie odczytany przez proces stacji A. Jednym ze sposobów na zlikwidowanie tej wady jest utworzenie kilku połączeń pomiędzy procesami, tzn. wielu różnych plików *ZADAN* oraz *WYNIK* jak: *ZADAN*.1, *ZADAN*.2, ... *ZADAN.M*, jak również *WYNIK*.1, *WYNIK*.2, ... *WYNIK.M* w liczbie M równej lub większej od liczby procesów N. Zapis przesyłanej informacji dokonywany jest wtedy do pierwszego wolnego pliku. Proces odczytujący informację kontroluje kolejno poszczególne pliki do czasu, aż znajdzie plik do wykorzystania. W tym rozwiązaniu każdy z plików jest chroniony oddzielnym semaforem. Przesył odbywa się zatem wieloma kanałami, co przyśpiesza działanie całości systemu.

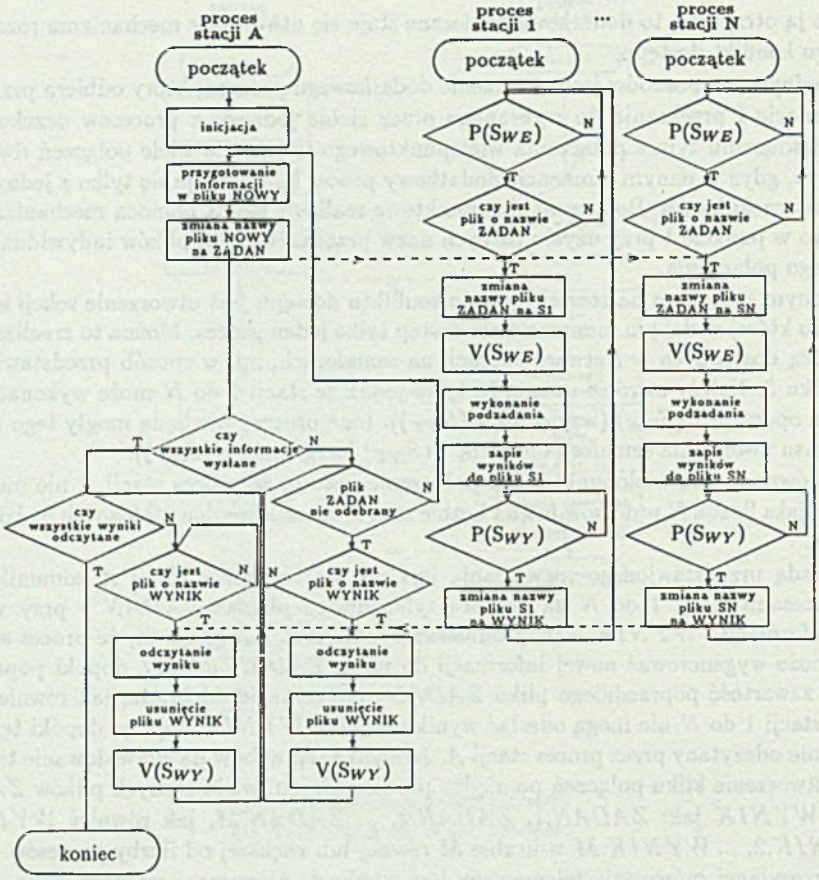
Tę ideę postępowania przyjęto przy implementacji przykładowego algorytmu równoległego poszukiwania rozwiązania uproszczonego, 10 problemu Hilberta.

4. Idea przetwarzania równoległego

Jeśli wielomian o całkowitych współczynnikach A_i: $i = 1, \ldots, n-1$ ma pierwiastki całkowite, to są one dzielnikami wyrazu wolnego A_0. Stąd algorytm równoległy szukania tych pierwiastków może wyglądać następująco:

W bloczku zatytułowanym *inicjacja* procesu stacji A z rysunku 3 szukane są podzielniki wyrazu wolnego A_0 wielomianu:

$$x^n + A_{n-1}x^{n-1} + \ldots + A_1x^1 + A_0$$



Rys. 3. Koordynacja wielu procesów stacji 1 do N z procesem stacji A

Fig. 3. Coordination of many processes of station 1 to N with the process of station A

Algorytm inicjacji jest następujący:

licz.dziel. := 0
for $u := |A_0|$ **to** $-|A_0|$ **step** -1 **do**
  begin
  if $u \neq 0$ **and** $\frac{A_0}{u}$ = *liczba całkowita* **then**
    begin
    licz.dziel. := *licz.dziel.* + 1
    tabl.dziel.[*licz.dziel.*] := u
    end
  end

Po wykonaniu powyższej procedury otrzymujemy liczbę dzielników *licz.dziel.* oraz wartości dzielników zgromadzone w tablicy *tabl.dziel.*

Krokiem następnym jest przygotowanie informacji dla procesów stacji 1 do N w pliku *NOWY* (rys. 3). W jednym pliku przesyłane są wszystkie współczynniki wielomianu oraz kolejna wartość podzielnika u odczytana z tablicy dzielników *tabl.dziel*.

Po przemianowaniu pliku *NOWY* na *ZADAN* zostaje on przyjęty przez jeden z procesów stacji 1 do N. Tu następuje wyliczenie wartości wielomianu. Ponieważ wielomian można zapisać w postaci:

$$(\ldots((x + A_{n-1})x + A_{n-2})x + \ldots + A_1)x + A_0,$$

zatem algorytm *wykonanie podzadania* z rysunku 3 jest następujący:

wielom. := 1
for $i := n - 1$ to 0 step -1 do
    wielom. := *wielom.* $* u + A_i$

Wynikiem zapisywanym do jednego z plików $S1$ do SN jest *wielom.*, będący wartością wielomianu oraz u, dla którego był on liczony.

Wynik ten jest następnie odczytany w procesie stacji A i interpretowany:

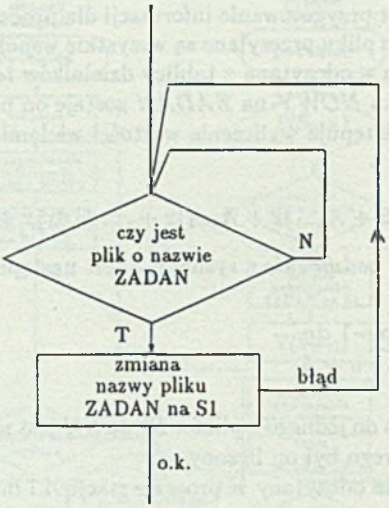
if *wielom.* = 0
    then *pisz* : *u jest pierwiastkiem calkowitym*
    else *pisz* : *u nie jest pierwiastkiem calkowitym*

5. Uwagi

W zrealizowanej implementacji algorytmu postać procesu stacji 1 do N została uproszczona i nie skorzystano z operacji semaforowych. Architektura systemu Netware spełnia funkcje szeregowania w dostępie do plików pamięci dyskowej serwera. Zatem zawsze znajdzie się proces, który wcześniej przed innymi wykona zmianę nazwy pliku *ZADAN* na własną, np. $S1$. Dla pozostałych procesów operacja ta jest już niewykonalna, gdyż plik o nazwie *ZADAN* nie będzie istniał. Brak możliwości zmiany nazwy nieistniejącego pliku sygnalizowany jest błędem, który odczytany w programie powoduje zaniechanie dalszych działań i powrót na jego początek (rys. 4). Rozwiązanie to eliminuje potrzebę stosowania semafora S_{WE}.

Semafor S_{WY} został wyeliminowany przez zastosowanie różnych nazw plików zawierających wyniki. Proces stacji k, $k \in [1, N]$ po przejęciu informacji z pliku *ZADAN.i*, $i \in [1, M]$ tworzy plik wynikowy o nazwie *WYNIK.i*. Żaden inny proces stacji 1 do N takiego pliku nie utworzy, gdyż proces stacji A jest tak skonstruowany, że pojawienie się nowej informacji w pliku *ZADAN.i* nastąpi dopiero po odczytaniu wyniku z pliku *WYNIK.i*.

W przedstawionym algorytmie najpierw wyliczane są wszystkie dzielniki wyrazu wolnego A_0, potem dopiero generowane są informacje – podzadania do wykonania za pomocą stacji 1 do N. Aby zyskać na czasie w zrealizowanej implementacji informacja jest generowana natychmiast po znalezieniu podzielnika. Dzięki temu w czasie, gdy szukany jest kolejny podzielnik, jeden z procesów stacji 1 do N zajmuje się już obliczaniem wartości wielomianu.



Rys. 4. Uproszczony mechanizm rozstrzygania konfliktu dostępu do pliku *ZADAN*
Fig. 4. Simplified mechanism of acces collision resolution to file *ZADAN*

Z obserwacji wynika, że czas liczenia wielomianu w stacjach 1 do N jest znacznie krótszy od czasu przesłania doń informacji. Stąd wprowadzenie równoległości na bazie sieci Netware wydłużyło czas otrzymania wyników w stosunku do realizacji jednoprocesorowej. Jednakże, jeśli zadania realizowane w stacjach będą czasochłonne, to wykorzystanie wielu stacji sieci powinno dać wymierne korzyści czasowe.

LITERATURA

[1] Węgrzyn S.: Systemy sterowane przepływem operacji i systemy sterowane przepływem argumentów. Zeszyty Naukowe Politechniki Śląskiej, seria Informatyka, z. 24, Gliwice 1993.

[2] Wilk A.: Wykorzystanie sieci Netware firmy Novell do tworzenia oprogramowania współbieżnego na przykładzie implementacji równoległego algorytmu rozwiązania uproszczonego problemu Hilberta (maszynopis nie publikowany).

[3] Novell, Dokumentacja systemu Netware v.3.11.

Recenzent: Prof. dr inż. Stanisław Kozielski

Wpłynęło do Redakcji 28 grudnia 1993 r.

Abstract

Parallel processing may be performed on the local area network being de facto a computer - processor set linking by the quick transmission medium e.g. Ethernet. The article describes the idea of popular Novell's Netware network utilisation for this purpose. Synchronization mechanisms of the individual computer processes are proposed due to features of this network. This coordination is controlled by the appearance of the subtask to execute and the results to interpret. Presented conception may be treated as a practical realization of the system controlled by flows of arguments.

Parallel algorithm of simplified Hilbert's problem resolution is used for the idea demonstration. It is to note that the resultant algorithm realization time depends both on the processing organisation and on the speed of transmission between computers.