ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1995
Seria: INFORMATYKA z. 28 Nr kol. 1270

Piotr BRUDŁO
Henryk KRAWCZYK

WYZNACZANIE BIEŻĄCYCH OBCIĄŻEŃ WĘZŁÓW LOKALNEJ SIECI KOMPUTEROWEJ *

Streszczenie. Zaproponowano metodę wyznaczania stanu obciążeń węzłów realizujących przetwarzanie rozproszone w sieci lokalnej. Dokonano jej praktycznej implementacji w postaci testu dla stacji roboczych pracujących pod systemem Unix. Przedstawiono wyniki eksperymentalne wykonania takiego testu w środowisku OpenWindows 3.3 i systemu operacyjnego SunOS Generic 5.3 pracujących na stacjach roboczych Sun Classic i Sun Sparc Station LX. Porównano otrzymane wyniki z obliczeniami teoretycznymi i zasygnalizowano realne możliwości wykorzystania tej metody przy implementacji przetwarzania rozproszonego.

DETERMINATION OF CURRENT PROCESSING LOAD FOR LOCAL AREA NETWORK NODES

Summary. A method of current load determination for local area network nodes realising distributed processing is proposed. Its practical implementation in the form of a test for Unix workstations is presented. Experimental results of the test execution in the environment of OpenWindows 3.3 and SunOS Generic 5.3 running on Sun Classic and Sun Sparc Station LX, respectively are given. The results are compared with theoretical estimations. Practical utilisation of the method for implementation of the distributed processing strategy is also highlighted.

* Praca sponsorowana przez grant KBN nr: 8 S503 017 07.

BESTIMMUNG DES LAUFENDEN LAST VON KNOTEN IM LOKALEN NETZE

Zusammenfassung. Die Arbeit enthält einen Vorschlag der Methode zur Bestimmung von Stand des Lasten der Knoten im lokalen Netze, das nach dem Prinzip verteiltes System funktioniert. Die Methode wurde als Test für Workstation unter Unix geprüft. Man zeigte die Ergebnisse für Systeme OpenWindows 3.3 und SunOS Generic 5.3, die arbeiten auf den Workstation Sun Classic und Sun Sparc Station LX. Die Ergebnisse wurden mit theoretischen Berechnungen vergleicht. Die gezeigte Methode kann bei Verarbeitung in verteilte Systemen verwendet werden.

1. Wstęp

Lokalne sieci komputerowe, oprócz tradycyjnych zastosowań, coraz częściej są adaptowane do wykonywania obliczeń równoległych i rozproszonych [2], [4], [12]. Wiele takich modeli przetwarzania wymaga znajomości aktualnych obciążeń węzłów w celu zapewnienia ich równomiernego obciążenia. W artykule skoncentrowano się na strategii wyznaczania aktualnego stanu obciążeń węzłów w sieci na podstawie testu czasowo-wydajnościowego. Celem działania tego testu jest określenie aktualnego obciążenia węzła sieci wyrażonego procentem przydziału czasu procesora dla pojedynczego zadania w stosunku do wszystkich przetwarzanych zadań. Dotychczasowe rozwiązania [1], [6], [7] nie uwzględniają dynamiki zmian rzeczywistych obciążeń poszczególnych węzłów. Zaprezentowano wyniki eksperymentalne proponowanego testu dla typowych konfiguracji sieciowych.

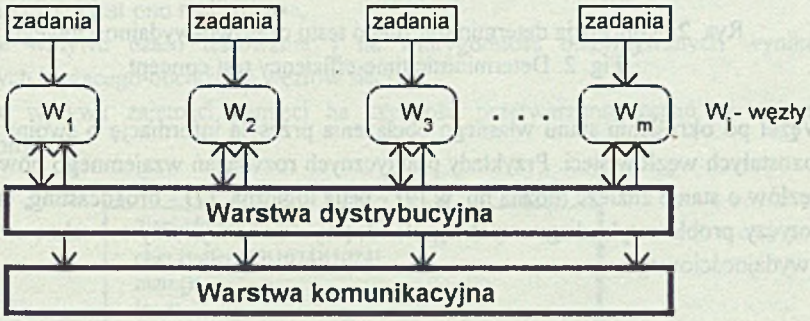
Przedstawiono analizę otrzymanych pomiarów, ich zgodność z estymatorami teoretycznymi oraz ich przydatność dla przetwarzania rozproszonego. Zasygnalizowano sposoby modyfikacji parametrów testów w celu zwiększenia dokładności oceny stanu bieżących obciążeń węzłów. Zwrócono uwagę na ograniczenia ilościowe dla obciążeń stacji roboczych Sun Classic i Sun Sparc Station LX pracujących w środowisku OpenWindows 3.3 pod systemem operacyjnym SunOS Generic 5.3.

2. Koncepcja przetwarzania rozproszonego

W sieci lokalnej każda ze stacji roboczych pracuje praktycznie niezależnie z możliwością dostępu do współdzielonych zasobów dyskowych (NFS, NIS, Yellow Pages etc. [8]), a także z możliwością jawnego dostępu (telnet, rsh, rlogin, etc. [10]) do zasobów obliczeniowych innych stacji. Istnienie wspólnego systemu plików powoduje, że czas dostępu do fizycznego nośnika (dysku, taśmy) dla dowolnej stacji roboczej jest praktycznie porównywalny. Z punktu widzenia użytkownika pracującego na danej stacji roboczej najefektywniejsze jest wykonanie

własnego programu na stacji o najmniejszym obciążeniu. Przy czym nie ma znaczenia, czy jest ona stacją lokalną czy zdalną. W celu optymalnego wykorzystania mocy obliczeniowej sieci istotne jest równomierne rozłożenie obciążenia obliczeniowego na wszystkie stacje LAN [5].

Przetwarzania rozproszonego dokonuje się poprzez rozdzielenie zadań w sieci na różne węzły. Może to być dokonane przez użytkownika bądź automatycznie. W modelu automatycznego przetwarzania rozproszonego wyróżnić można dwie podstawowe warstwy zarządzania: warstwę komunikacyjną i warstwę dystrybucyjną [3]. Model takiego przetwarzania jest przedstawiony na rys. 1. Każda ze stacji roboczych otrzymuje zadania do wykonania od użytkowników z konsoli. Jednocześnie stacje mogą inicjować zdalne wykonanie własnych zadań na pozostałych stacjach będących w sieci. W warstwie komunikacyjnej określane są bieżące obciążenia stacji roboczych oraz dokonywane jest powiadamianie wszystkich węzłów o bieżących obciążeniach. W warstwie dystrybucyjnej na podstawie stanu obciążeń dokonywany jest rozdział zadań pomiędzy stacje robocze zgodnie z określonym kryterium np. największej globalnej efektywności [5]. W założonym modelu przetwarzania rozproszonego stacje pracują całkowicie niezależnie. Stacja podejmuje decyzje o lokalnym bądź zdalnym wykonaniu danego zadania jedynie na podstawie aktualnie posiadanych informacji o obciążeniach innych stacji w sieci. Stąd wniosek, że od aktualności i dokładności informacji o obciążeniach, jako danych będących podstawą do dystrybucji zadań, zależy globalny stopień uzyskania założonego kryterium przetwarzania rozproszonego. W artykule skoncentrowano się na sposobach określania bieżącego stanu obciążeń, zaznaczając jednocześnie możliwości wzajemnego powiadamiania się stacji o tym stanie.



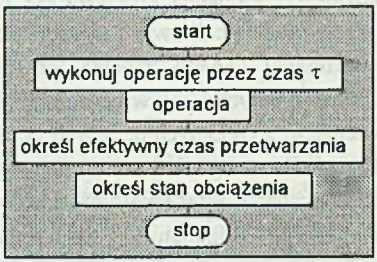
Rys. 1. Model sieciowego przetwarzania rozproszonego
Fig. 1. Distributed network processing model

3. Strategia określania stanu obciążeń

W założonym modelu przetwarzania stacje robocze pracują w trybie podziału czasu. Każde z zadań dotyczące poszczególnego węzła jest więc efektywnie przetwarzane przez przedział czasu proporcjonalny do jego priorytetu. Dla przykładu, jeżeli rozważymy węzeł, na

którym przetwarzanych jest jednocześnie m zadań o priorytecie jednostkowym, to zgodnie z zasadą przydziału procesora każde z zadań otrzymuje 1/m efektywnego czasu przetwarzania. Przy przetwarzaniu zadań o różnych priorytetach, przy sumie priorytetów wynoszących m, na jednostkę priorytetu przypada 1/m efektywnego czasu procesora. Przy czym uwzględnione być musi dodatkowo obciążenie wprowadzane przez system operacyjny.

Do określania bieżącego obciążenia węzłów sieci zaproponowano koncepcję deterministycznego testu czasowo-wydajnościowego. Test ten został zdefiniowany jako zadanie o jednostkowym priorytecie, a podstawową jego cechą jest to, że rzeczywisty czas τ potrzebny do określenia obciążenia jednostki jest stały (całkowicie niezależny od tego obciążenia). Oznacza to, że po upłynięciu zadanego kwantu czasowego τ wykonanie testu jest przerywane. Określany jest efektywny czas przetwarzania testu przez procesor. Bieżący stan obciążenia jednostki określany jest jako procent efektywnego czasu przetwarzania testu w zadanym kwancie czasowym τ. Koncepcja takiego testu jest przedstawiona na rys. 2.



Rys. 2. Koncepcja deterministycznego testu czasowo-wydajnościowego
Fig. 2. Deterministic time-efficiency test concept

Węzeł po określeniu stanu własnego obciążenia przesyła informację o swoim obciążeniu do pozostałych węzłów sieci. Przykłady praktycznych rozwiązań wzajemnego powiadamiania się węzłów o stanie znaleźć można np. w [9] - pętla logiczna, [7] - broadcasting, etc. Pozycja [9] dotyczy problematyki diagnostycznej, ale ideę tę można wykorzystać także w przypadku testu wydajnościowego.

4. Implementacja testu wydajnościowego w środowisku Unix

Test wydajnościowy zaimplementowany został jako jednoprocesowe (bez rozwidleń funkcją *fork*) zadanie z wykorzystaniem systemowych timerów. Wykorzystano następujące timery systemowe:

1. ITIMER_REAL - timer czasu rzeczywistego,
2. ITIMER_PROF - timer efektywnego czasu wykonania zadania,

3. ITIMER_VIRTUAL - timer efektywnego wykonania zadania bez uwzględnienia wywołań systemowych.

Zasadą korzystania z timerów jest inicjalne wpisanie do nich zadanych wielkości czasowych oraz implementacja procedur, do których następuje przejście w przypadku upłynięcia zadanego kwantu czasowego. Jak łatwo zauważyć, procent czasu procesora przydzielonego do wykonania testu może być obliczony jako iloraz czasu efektywnego wykonania zadania (timer 2) i czasu rzeczywistego (timer 1). Przy czym timer 1 jest ustawiany na zadaną wartość czasu, a w procedurze obsługi dla tego timera odczytywany jest timer 2. Wielkość otrzymana jest bezpośrednią miarą obciążenia danego węzła. Przy czym pamiętać należy, że tak określona wartość musi być jeszcze znormalizowana względem sprzętowej szybkości przetwarzania.

5. Eksperymenty praktyczne

Przeprowadzone eksperymenty miały na celu określenie stopnia przydatności zaproponowanego testu do dynamicznego określania stanu obciążeń stacji roboczych. Cele przeprowadzonych eksperymentów są następujące:

- przedstawienie czasu efektywnego przetwarzania zadania w funkcji zadanego obciążenia węzła, na którym jest ono realizowane,
- określenie wpływu czasu testowania τ na wiarygodność otrzymywanych wyników dotyczących bieżącego obciążenia węzłów sieci,
- określenie wpływu zajętości pamięci na szybkość przetwarzania zadań w systemie rozproszonym.

```
#include <stdio.h>
char Buffer [X][1024][1024]
main(){
int x;
    while(1){
    x = x-3;
    x = x+3;
        }
}
```

Rys. 3. Zadanie emulujące obciążenie
Fig. 3. Load emulation task

W celu emulacji obciążeń węzła uruchamiano proste zadania składające się z nieskończonej pętli obliczeniowej. Tekst przykładowego zadania przedstawiono na rys. 3, przy czym wielkość X jest rozmiarem tablicy w MB (dla zbadania wpływu zajętości pamięci na szybkość przetwarzania). Zadanie to jest przetwarzane ciągle (nie czeka na żadne zdarzenia

zewnętrzne) i jego wpływ na system jest taki sam jak każdego innego aktywnego zadania. Przeprowadzono eksperymenty dla konfiguracji przedstawionych w tabeli 1.

Tabela 1

Analizowane konfiguracje sprzętowe

Nazwa komputera	Typ stacji	Pamięć fizyczna	Pamięć wirtualna	System operacyjny	Środowisko
loqi05.elka.pg.gda.pl	Sparc Classic sparc (sun 4m)	24M	75M	SunOS Release 5.3 Generic	OpenWindows Version 3.3
wicio.elka.pg.gda.pl	Sparc Classic sparc (sun 4m)	16M	109M	SunOS Release 5.3 Generic	OpenWindows Version 3.3
roger06.elka.pg.gda.pl	Sparc Classic sparc (sun 4m)	32M	123M	SunOS Release 5.3 Generic	OpenWindows Version 3.3
howell.elka.pg.gda.pl	Sparc Station LX sparc (sun 4m)	40M	125M	SunOS Release 5.3 Generic Patch	OpenWindows Version 3.3

Dla tych konfiguracji otrzymano podobne wyniki eksperymentów. Jest to zgodne z przewidywaniami, gdyż o szybkości przetwarzania decyduje typ procesora. Dlatego też przedstawione eksperymentalne wyniki obejmują globalnie wszystkie wymienione konfiguracje.

Przeprowadzono eksperymenty w zakresie zmienności parametrów:

- czasu testowania od 1 do 20 sekund co 1 sekundę,
- ilości zadań obciążających od 0 do 35 co 1.

Na podstawie otrzymanych wyników można sformułować następujące wnioski:

- Dla około 23-24 procesów efektywny czas wykonywania testu stabilizuje się. Jest to efekt konstrukcji systemu operacyjnego, który jest zdolny aktywnie przetwarzać tę liczbę procesów (pozostałe procesy czekają). Stąd wniosek, że granicą, dla której można stosować test przy badaniu obciążenia, jest liczba 23-24 procesów aktywnych dla każdego węzła sieci.
- W zależności od czasu testowania zmienia się dokładność wyników. Dla czasów testowania τ = 10-20s istnieje możliwość określenia obciążenia (mierzonego ilością procesów aktywnych) w zakresie 1-23 procesów z dokładnością do 1-2 procesów. Dla czasów testowania τ = 5-9s można określić obciążenie z dokładnością do 1-2 procesów tylko w zakresie 1-15 procesów, natomiast w zakresie 15-23 procesów dokładność ta spada do 3-4 procesów. Dla czasów testowania τ = 1-5s tylko w zakresie 1-5 procesów można dokładnie określić obciążenie.
- Dla ponad 30 procesów interakcyjna praca z systemem staje się praktycznie niemożliwa (czas reakcji systemu powyżej 10s).

Proponowany test z jednej strony powinien wprowadzać możliwie najmniejsze obciążenie (krótki czas wykonania), z drugiej zaś pozwalać na dokładne określenie obciążenia (im dłuższy czas wykonania tym dokładniej). Stąd konieczne wydaje się osiągnięcie pewnego

kompromisu. Dla pełnego zakresu obciążeń czas testowania 10s wydaje się być w pełni zadowalający. Dla spodziewanych obciążeń do 15 procesów można zastosować test krótszy (np. 5s). Wybrane wyniki przedstawiające procent efektywnego czasu przetwarzania testu w funkcji czasu testowania τ zestawiono w tabeli 2.

W celu analizy wzrostu czasu przetwarzania zadania w zależności od obciążenia uruchamiano zadanie wymagające efektywnego czasu przetwarzania 1s przy obciążeniu w zakresie 1-30 procesów. Wyniki w formie wykresu przedstawione są na rys. 4. Teoretycznie czas wykonania powinien być liniowy. Uzyskane wyniki w zasadzie są zgodne z przewidywaniami (przy 9 procesach pewne anomalie) w zakresie 0-23 aktywnych procesów. Powyżej zakresu czas wykonania stabilizuje się, co potwierdzają wyniki wcześniejszych eksperymentów. Ponadto czas wykonania procesu jest nieco mniejszy, niż wynikałoby to z liczby zadań. Spowodowane jest to tym, że system operacyjny przydziela więcej efektywnego czasu procesora zadaniom, które zostały uruchomione później (w stosunku do zadań już przetwarzanych).

Tabela 2

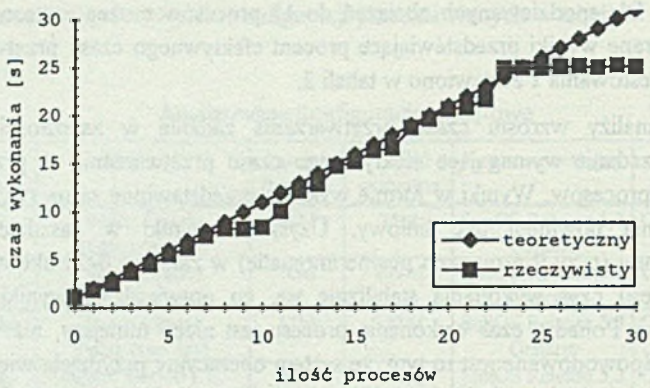
Procent efektywnego czasu przetwarzania testu w funkcji czasu testowania τ i obciążenia stacji wyrażonego liczbą aktywnych procesów

Ilość procesów / czas testowania τ [s]	20	15	10	5	1
0	98.0%	98.0%	98.0%	97.0%	99.0%
5	18.5%	18.0%	19.0%	20.0%	20.0%
10	12.0%	12.0%	12.0%	12.3%	12.0%
15	6.5%	6.6%	6.6%	6.4%	8.0%
20	5.0%	5.1%	5.2%	5.6%	8.0%
25	4.2%	4.1%	4.0%	4.0%	4.0%
30	4.2%	4.1%	4.0%	4.0%	4.0%

W przeprowadzonych badaniach nie stwierdzono wpływu zajętości pamięci na szybkość przetwarzania.

6. Wnioski końcowe

Test wydajnościowy może być wykorzystywany do konfiguracji sprzętowych z tabeli 1 (istnieje możliwość dokładnego określenia stanu obciążenia) w zakresie 1-23 procesów. Stopień dokładności zależy od czasu testowania τ, przy czym okres 10s wydaje się spełniać z jednej strony oczekiwania efektywnościowe (małe obciążenie wnoszone przez test), jak i zapewniać wystarczającą z praktycznego punktu widzenia dokładność określenia stanu.



Rys. 4. Porównanie teoretycznego i rzeczywistego czasu przetwarzania zadania w funkcji obciążenia stacji roboczej

Fig. 4. Theoretical and real task processing time in relation with workstation load

Niedogodnością przyjętego rozwiązania jest wprowadzanie obciążenia przez test. Podejściem alternatywnym jest analiza informacji o procesach otrzymywana od jądra systemu operacyjnego. Jednakże z punktu dystrybucji zadań istotny jest procent efektywnego czasu przydzielonego dla zadania na określonej stacji roboczej, nie zaś faktyczna liczba zadań będących w systemie. Ponadto podejście testowania ujmuje globalnie wszystkie klasy procesów (czasu rzeczywistego, systemowe, z podziałem czasu) pracujących w systemie Unix.

Istotne jest badanie strategii wzajemnego powiadamiania się węzłów o bieżącym obciążeniu określonym za pomocą proponowanego testu. Przewiduje się opracowanie i analizę dynamicznej strategii doboru czasu testowania wydajnościowego w zależności od przewidywanego stanu obciążeń.

LITERATURA

[1] Beguelin A., Dongarra J., Geist A., Jiang W., Manchek R.: HeNCE: A User's Guide Version 1.2, Oak Ridge National Lab., Univ. of Tennessee, grudzień 1992.

[2] Brudło P., Krawczyk H.: A Concept of a Fully Distributed Processing Strategy oriented for Unix Local Area Networks, Euromicro Journal, przyjęte do publikacji.

[3] Brudło P.: Dwuwarstwowa strategia rozproszonego przetwarzania zadań w lokalnej sieci komputerowej, Materiały konferencji Informatyka na Wyższych Uczelniach dla Gospodarki Narodowej, Gdańsk, listopad 1994, tom 1, s. 135-139.

[4] Brzeziński J.: Problemy przetwarzania rozproszonego, Materiały konferencyjne POLMAN'94, Poznań, maj 1994, s. 173-187.

[5] Burdorf C., Marti J.: Load Balancing Strategies for Time Warp on Multi-User Workstations, The Computer Journal, vol. 36, no. 2, 1993, s. 168-176.

[6] Geist A., Beguelin A., Dongarra J., Jiang W., Manchek R., Sunderam V.: PVM 3 User's Guide and Reference Manual, Oak Ridge National Lab., Univ. of Tennessee, maj 1993.

[7] Graf T. P., Assini R. G., Lewis J. M.: HP Task Broker: A Tool for Distributing Computational Tasks, Hewlett-Packard Journal, sierpień 1993, s. 15-22.

[8] Rose M.: The Simple Book: An Introduction to Internet Management, Prentice Hall, Eaglewood Cliffs, New Jersey, 1994.

[9] Silva L., Silva J.: DIP: Distributed Diagnosis Protocol, Microprocessing and Microprogramming, vol. 38, 1993, s. 171-177.

[10] Stevens W. R.: Unix Network Programming, Prentice Hall, Eaglewood Cliffs, New Jersey, 1990.

[11] Uszok A., Król A., Zieliński K.: Cluster Management Software for Open Object Oriented Systems, Materiały konferencyjne HPCN'94, Monachium, 1994, s.140-141.

[12] Zieliński K., Czajkowski G., Gajęcki M.: Parallel Programming Systems for LAN Distributed Computing, Materiały konferencyjne ICDCS-14, Poznań, czerwiec 1994, s. 600-607.

Recenzent: Prof. dr hab. inż. Andrzej Grzywak

Wpłynęło do Redakcji 21 listopada 1994 r.

Abstract

Integration tendencies of local area network active processing resources are highlighted. The two level concept of a fully distributed processing strategy is presented (see Fig. 1). The strategy is fully distributed in the sense that there is no controlling or managing unit in the network. Then the two management levels can be distinguished. On the communication level workload distribution in the network is monitored and the load distribution information is delivered to workstations. The distribution level is responsible for task distribution according to the normalised load balancing criterion. A method of current load determination for local area network nodes realising distributed processing is proposed. Its practical implementation in the form of a test for Unix workstations is presented. The logical structure of the test is shown in Fig. 2. A workstation load is determined as a percent of time assigned to a single task (represented by the test) in the time sharing processing. If the tasks' priorities are the same for all of the tasks the workload may be expressed as a number of active tasks according to the time sharing principle. In order to determine effective processing time of the test Unix system timers are utilised. The test is not dependent on a workstation's load. The Unix real time timer provides constant execution time of the test. Experimental results of the test execution in the

environment of OpenWindows 3.3 and SunOS Generic 5.3 running on Sun Classic and Sun Sparc Station LX for various memory configurations are given (Table 2). The results are compared with theoretical expectations (Fig. 4). It is shown that the results are close to theoretical expectations in the range of 1-23 active tasks. If the number of active tasks exceeds 23 the congestion state is observed. Possibilities of the test utilisation for current workload estimation in the permissible range are discussed. Application of the method for distributed processing is suggested and future development works are highlighted.