Jan SADECKI
POLITECHNIKA OPOLSKA, WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI I INFORMATYKI,
INSTYTUT AUTOMATYKI I INFORMATYKI

Analiza efektywności klastrowych aplikacji algorytmów komunikacji

Dr hab. inż. Jan SADECKI

Jest długoletnim pracownikiem Politechniki Opolskiej. Stopień doktora nauk technicznych (1988) oraz doktora habilitowanego (2004) uzyskał w zakresie automatyki i robotyki na Wydziale Elektroniki i Technik Informacyjnych Politechniki Warszawskiej. Zajmuje się przetwarzaniem równoległym i rozproszonym, systemami równoległymi, klastrami obliczeniowymi.

e-mail: jsad@po.opole.pl

Streszczenie

W artykule przedstawiono przykładowe rezultaty analizy efektywności pewnych zadań komunikacyjnych dla systemów klastrowych o strukturze dwupoziomowej. Rozważana metoda dekompozycji systemu wieloprocesorowego może prowadzić do znacznej poprawy efektywności realizacji bardzo czasochłonnych zadań komunikacji typu „each-to-each".

Słowa kluczowe: klastry, algorytmy komunikacji, dekompozycja.

An analysis of efficiency of cluster implementation of communication algorithms

Abstract

The paper presents the results of the efficiency analysis of some communication algorithms for two-level cluster architecture. The applied decomposition of parallel multiprocessor system into e few groups of processors is a key step for effective realization of the time consuming each-to-each communication tasks.

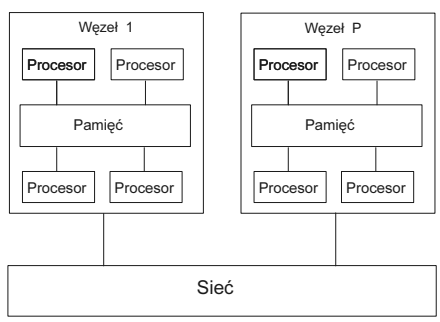
Keywords: clusters, communication algorithms, decomposition.

1. Wstęp

Artykuł dotyczy wybranych problemów związanych z badaniem możliwości, sposobów oraz efektywności zastosowania systemów klastrowych do rozwiązywania złożonych czasowo zagadnień obliczeniowych. Klaster jest rodzajem systemu rozproszonego składającego się z pewnej liczby połączonych ze sobą, najczęściej typową siecią, niezależnych jednostek komputerowych, stanowiących z punktu widzenia użytkownika pojedynczy, jednorodny zasób obliczeniowy. W chwili obecnej klastry obliczeniowe, głównie ze względu na szeroką dostępność komputerów osobistych jak też specjalistycznego oprogramowania oraz relatywnie niskie koszty stają się jednym z najpopularniejszych narzędzi przetwarzania równoległego [1, 3]. Obserwując tendencje rozwojowe w dziedzinie architektury systemów klastrowych, można zauważyć stopniowe odchodzenie od struktury jednopoziomowej (węzły jednoprocesorowe) na rzecz struktury dwupoziomowej, w której węzły klastra stanowią 2, 4, 8... elementowe zespoły procesorów typu SMP komunikujących się ze sobą za pośrednictwem pamięci współdzielonej. Dodatkowo węzły klastra mogą być wspomagane poprzez zastosowanie technologii HT (*Hyper Threading*), co, jak pokazują badania praktyczne, może zwiększyć efektywność wykorzystania poszczególnych węzłów o dalsze 20–25%. Specyfika rozwiązań klastrowych wymaga jednak szczególnie starannego podejścia do problemu projektowania aplikacji rozproszonych, głównie odnośnie konfiguracji zadań komunikacyjnych (przez zadanie komunikacyjne rozumie się tutaj przesłanie danych pomiędzy procesorami, niezbędne dla prawidłowej realizacji algorytmu równoległego). Wynika to głównie ze stosunkowo dużych kosztów czasowych procesów komunikacyjnych, szczególnie dla zadań o niedużej ziarnistości, Krytycznym problemem są tutaj aplikacje wymagające częstych komunikacji pomiędzy każdą parą wykorzystywanych w klastrze węzłów, z czym mamy do czynienia w przypadku, gdy na pewnym etapie realizacji algorytmu występuje konieczność utworzenia w pamięci lokalnej poszczególnych procesorów kopii rezultatów obliczeń otrzymanych we wszystkich węzłach klastra [5]. Zadanie to nabiera szczególnej wagi w odniesieniu do struktury dwupoziomowej, w której mamy do czynienie z dwoma rodzajami komunikacji, a mianowicie z komunikacją w obrębie jednego węzła oraz praktycznie o rząd wolniejszą komunikacją pomiędzy węzłami klastra. Implikuje to celowość takiego projektowania aplikacji równoległych aby zadania często się ze sobą komunikujące były lokowane w obrębie tego samego węzła, natomiast zadania wymagające rzadszej wymiany danych – na różnych węzłach klastra. Obniżenie wymagań komunikacyjnych można uzyskać poprzez odpowiednie zaprojektowanie (dekompozycję) struktury zadań komunikacyjnych. Jednym z celów prezentowanej pracy jest przebadanie efektywności przykładowych realizacji zadań komunikacyjnych, a w dalszej perspektywie optymalizacja aplikacji równoległych polegająca na dostosowaniu tych algorytmów (rozdział zadań obliczeniowych, organizacja komunikacji) do specyfiki wymagań klastrów obliczeniowych

2. Klastry obliczeniowe

Klaster to technologia łączenia komputerów, mająca na celu utworzenia wirtualnego superkomputera dedykowanego do przetwarzania równoległego. Głównym celem zastosowania klastrów w praktyce jest zwiększenie mocy obliczeniowej [1]. Współczesne klastry budowane są najczęściej przy użyciu węzłów dwu- lub czteroprocesorowych, połączonych ze sobą szybką siecią wymiany danych (rys. 1). Atrakcyjność klastrów wynika głównie ze stosukowo niskiej ceny, jak i szerokiej dostępności sprzętu oraz oprogramowania stanowiącego podstawę ich budowy. Jednak parametry stosowanych obecnie w klastrach nawet najbardziej zaawansowanych sieci nie gwarantują efektywnej równoległej realizacji algorytmów obliczeniowych wymagających wykonania szczególnie dużej ilości zadań komunikacyjnych. Chodzi tu nie tyle o szybkość transmisji danych, która dzisiaj jest już bardzo wysoka, lecz o opóźnienia transmisyjne (*latency*), których wartość jest ciągle zbyt duża. Jest to szczególnie uciążliwe w przepadku realizacji zadań wymagających częstych transmisji danych [6].



Rys. 1. Architektura klastra
Fig. 1. Cluster architecture

Każdy komputer będący elementem składowym klastra nosi miano węzła *(node)*. Węzłem klastra może być pojedyncza maszyna (np. typowy komputer klasy PC), lub też maszyna typu SMP (*Symmetric Multi Processing*), składająca się kilku procesorów komunikujących się poprzez wspólną pamięć (rys. 1).

3. Wymagania komunikacyjne

W procesie tworzenia efektywnych aplikacji klastrowych dużego znaczenia nabiera faza ich projektowania, w której należy dokonać takiej alokacji procesów oraz dystrybucji danych z których one korzystają w trakcie realizacji obliczeń, aby zminimalizować - na ile jest to możliwe – ilość zadań komunikacyjnych. Poniżej przedstawiono wyniki analizy teoretycznej, zmierzającej do pokazania możliwości obniżenia liczby komunikatów, bazującej na idei dekompozycji (grupowania procesorów), oraz na założonej możliwości jednoczesnego wykonywania tych zadań przez różne grupy procesorów. Typowym, bardzo czasochłonnym a jednocześnie często występującym w aplikacjach równoległych wielu algorytmów problemem jest sytuacja, którą można określić mianem pełnej wymiany, polegająca na konieczność utworzenia w pamięci lokalnej poszczególnych procesorów kopii rezultatów obliczeń otrzymanych we wszystkich węzłach klastra (*each-to-each communication problem*).

Całkowitą ilość komunikatów L_s jakie należy przesłać realizując takie zadanie określa zależność:

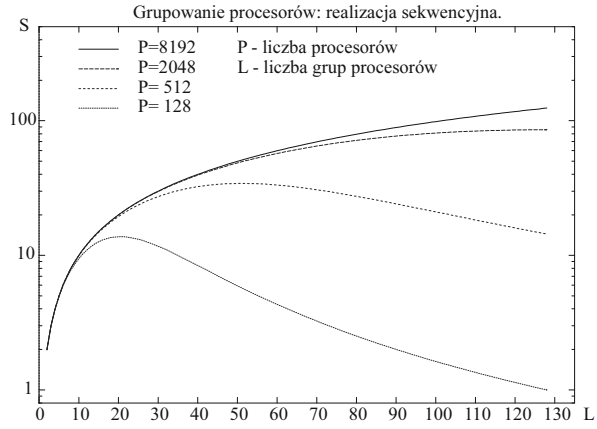
$$L_s = P(P-1), \tag{1}$$

gdzie P oznacza liczbę procesorów.

Dzieląc wszystkie procesory na L grup, z których każda będzie zawierała w przybliżeniu P/L procesorów, całkowita liczba komunikatów jakie należy przesłać realizując to samo zadanie będzie określona przez zależność:

$$L_d = \frac{P}{L}\left(\frac{P}{L}-1\right)L + L(L-1) + \left(\frac{P}{L}-1\right)L, \tag{2}$$

gdzie $(P/L)(P/L\text{-}1)L$ oznacza sumaryczną liczbę zadań komunikacyjnych realizowanych w obrębie wszystkich grup procesorów, $L(L\text{-}1)$ oznacza liczbę komunikatów przesyłanych pomiędzy wszystkimi grupami, natomiast $(P/L\text{-}1)L$ jest ilością komunikatów przesyłanych w obrębie grup procesorów, związanych z rozesłaniem rezultatów otrzymanych od innych grup procesorów.



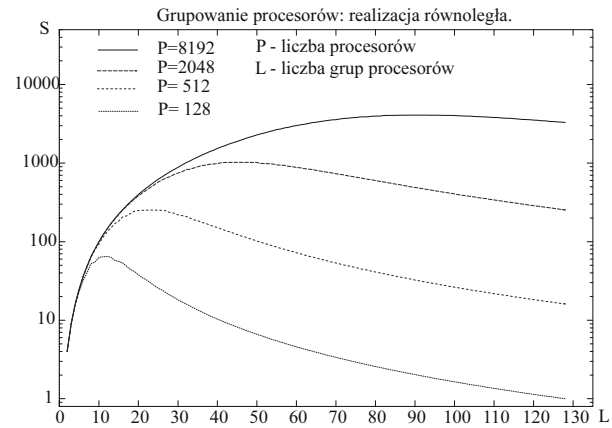
Rys. 2. Wpływ liczby grup procesorów na wartość współczynnika S
Fig. 2. Influence of the number of processor's groups on the value of S

Na rys. 2 przedstawiono wykres zależności wartości współczynnika określonego zależnością:

$$S = L_s / L_d \tag{3}$$

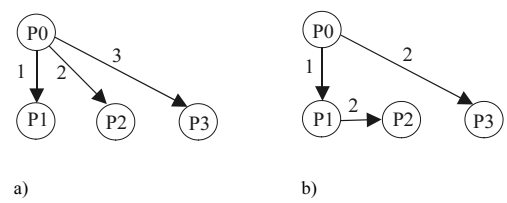
w funkcji liczy grup na jakie zostały podzielone wszystkie procesory dla kilku przykładowych wartości P. Rysunek ten pokazuje, jak znacząco mogą zostać zmniejszone wymagania komunikacyjne związane z realizacją zadania typu each-to-each w konsekwencji zastosowanej dekompozycji wirtualnej struktury procesorów. Wskazuje on również iż dla danego P można określić optymalną, tzn. gwarantującą najmniejsze obciążenie komunikacyjne klastra liczbę grup, na jakie należałoby podzielić procesory.

Rysunek ten obrazuje korzyści wynikające z grupowania procesorów (procesów), przy założeniu sekwencyjnego wykonywania wszystkich zadań komunikacyjnych. Jednakże teoretycznie, jak również w wielu sytuacjach praktycznie (zależy to m.in. od struktury sieci) możliwe jest jednoczesne wykonywanie zadań komunikacyjnych realizowanych w poszczególnych grupach. Korzyści wynikające z takiego sposobu realizacji komunikacji będą jeszcze bardziej widoczne w odniesieniu do liczby komunikatów określonej zależnością (1) (rys. 3).



Rys. 3. Wpływ liczby grup procesorów na wartość współczynnika S dla niezależnej komunikacji w grupach
Fig. 3. Influence of the number of processor's groups on the value of S factor for independent communications in all groups

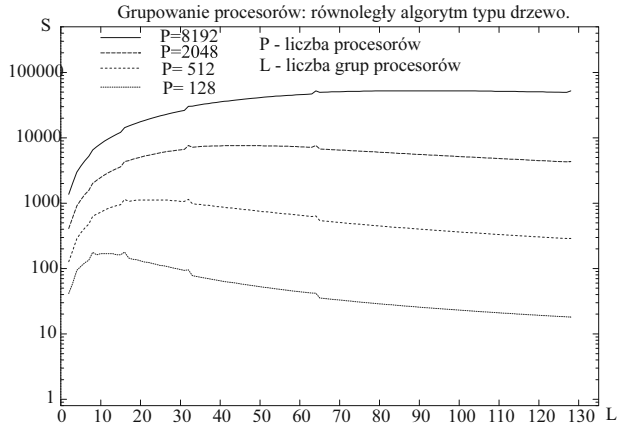
Ważnym problemem jest również sposób konkretnej realizacji złożonych zadań komunikacyjnych. W praktycznych aplikacjach (np. MPI) zadania takie, jak np. rozesłanie przez dany procesor tego samego komunikatu do wielu lub wszystkich pozostałych procesorów optymalizowane jest w oparciu strukturę drzewa (binominal tree) (rys. 4). Taka realizacja nie zmniejsza bezwzględnej ilości komunikatów, lecz specyfika ich rozsyłania przy założeniu że jest możliwe rzeczywiście jednoczesne przesyłanie komunikatów pomiędzy różnymi parami procesorów powoduje dalszą poprawę implementacji zadania each-to-each (rys. 5).



Rys. 4. Algorytmy wymiany: a) sekwencyjny, b) drzewiasty
Fig. 4. Broadcast algorithms: a) sequential, b) binominal tree

Reasumując, rysunki 2, 3 oraz 5 pokazują jak znaczące obniżenie w zakresie ilości komunikatów niezbędnych do wykonania rozważanego problemu wymiany danych można otrzymać drogą grupowania procesorów, czyli tworzenia wirtualnej, dwupoziomowej ich struktury, odpowiadającej zresztą rzeczywistej strukturze wielu systemów klastrowych. Można ponadto dokonać optymalizacji procesu dekompozycji pod kątem minimalizacji wymagań komunikacyjnych – krzywe na omawianych wykresach posiadają niejednokrotnie wyraźne maksimum. Przykładowo, dla przypadku analizowanego na rys. 2 (komunikaty pomiędzy procesorami przesyłane sekwencyjnie), największe korzyści można

otrzymać dzieląc wszystkie procesory na 22 grupy – dla P=128 (po 5-6 procesorów w grupie), oraz odpowiednio na 51 grup – dla P=512 (po 10 procesorów w grupie). Jeżeli natomiast dopuści się możliwość jednoczesnego (niezależnego) przesyłania komunikatów w obrębie grup (rys. 3), wówczas najlepsze wyniki otrzymuje się dla węzłów o w przybliżeniu dwukrotnie większej liczbie procesorów w poszczególnych grupach (dla P=128: 11-12 grup, tzn. po 11-12 procesorów w grupie).



Rys. 5. Wpływ liczby grup procesorów na wartość współczynnika S dla drzewiastej komunikacji w grupach. jak i pomiędzy grupami
Fig. 5. Influence of the number of processor's groups on the value of S factor for tree communications in groups and between them

4. Modelowanie zadań komunikacyjnych

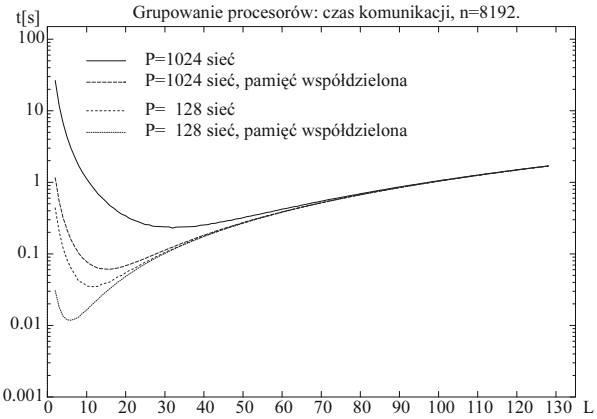
Prezentowane powyżej rezultaty pokazują iż praktycznie to samo zadanie wymiany danych może być zrealizowane na wiele różnych sposobów, różniących się nawet znacznie pod względem efektywności. Wskazane jest zatem przeprowadzenie analizy pozwalającej na wybór najlepszego z możliwych rozwiązań, uwzględniającego zarówno specyfikę algorytmu, jak i strukturę klastra. Analizę taką można przeprowadzić opierając się na prostym, lecz często stosowanym modelu, pozwalającym na estymację czasu transmisji komunikatu, który można w przybliżeniu określić na podstawie znajomości dwóch parametrów: czasu opóźnienia (latency) α oraz szybkości transmisji danych β. Przez czas opóźnienia rozumie się, czas jaki zachodzi pomiędzy momentem, gdy pojawi się konieczność wysłania komunikatu a momentem gdy ta transmisja będzie się mogła rozpocząć. Natomiast szybkość transmisji określona jest przez średni czas transmisji jednego bajtu [2, 5].

$$t_{com} = \alpha + m\beta, \tag{4}$$

Przez m oznaczono w zależności (4) długość komunikatu. Analizę taką przeprowadzono dla przypadku rozważanego na rys. 3. Dla parametrów α oraz β przyjęto przykładowe konkretne wartości: dla komunikacji wewnątrz grupy (pamięć współdzielona) – dla procesora P III ($\alpha_{\text{sh-mem}} \approx 4*10^{-6}$, $\beta_{\text{sh-mem}} \approx 6.6*10^{-9}$), natomiast dla komunikacji międzywęzłowej – dla sieci Gigabit Ethernet ($\alpha_{\text{network}} \approx 10^{-4}$, $\beta_{\text{network}} \approx 1.4*10^{-8}$) [2].

Przykładowe wyniki analizy przedstawiono na rys. 6. Przez n na rysunku oznaczono długość przetwarzanego równolegle przez wszystkie procesory wektora, którego kopia powinna się znaleźć po fazie wymiany w każdej lokalnej pamięci procesora.

Rysunek ten pokazuje bardzo wyraźnie, iż możliwe jest przy przyjętych założeniach optymalne określenie hipotetycznej struktury systemu, gwarantującej przy pewnych parametrach (jak np. liczba dostępnych procesorów P, wartości parametrów α oraz β, ilość przesyłanych danych n) uzyskanie najmniejszego czasu wymaganego do realizacji zadania komunikacji typu each-to-each. Na rysunku tym przeanalizowano dwa warianty, a mianowicie przypadek gdy wszystkie komunikaty pomiędzy procesorami przesyłane były poprzez sieć, oraz przypadek gdy komunikacja pomiędzy grupami realizowana była poprzez sieć, natomiast wymiana danych pomiędzy procesorami w grupie – poprzez pamięć współdzieloną



Rys. 6. Efektywność algorytmów komunikacji
Fig. 6. Performance of broadcast algorithms

5. Podsumowanie

Prezentowany artykuł dotyczy zagadnień związanych z badaniem możliwości oraz sposobów implementacji wybranej grupy równoległych algorytmów komunikacji, możliwych do zaimplementowania w systemach klastrowych. Pokazano, że właściwy dobór sposobu organizacji komunikacji międzyprocesorowej może mieć znaczący wpływ na efektywność obliczeń równoległych, świadcząc też o celowości badań zmierzających do efektywnego ograniczania ilości zadań komunikacyjnych przy projektowaniu równoległych aplikacji klastrowych dla różnych algorytmów obliczeniowych.

6. Literatura

[1] Baker M. (ed.): Cluster computing white paper, University of Portsmouth, UK, 2000, http://www.dcs.port.ac.uk/~mab/tfcc/ White Paper.
[2] Gergel V.: Optimizing Performance of MPI open-source implementations for Linux on POWER processor clusters, 2005, http://www.spscicomp.org/ScicomP11/Presentations/User/gergel.pdf.
[3] Karbowski A., Niewiadomska-Szynkiewicz E. (eds): Obliczenia równoległe i rozproszone. Oficyna Wydawnicza Politechniki Warszawskiej, 2001.
[4] Kumar V., Grama A., Gupta A., Karypis G.: Introduction to Parallel Computing, Addison Wesley, 2002.
[5] Sadecki J.: Algorytmy równoległe optymalizacji i badanie ich efektywności; systemy równoległe z rozproszoną pamięcią, Studia i monografie, Politechnika Opolska, Opole 2001.
[6] Sadecki J., Saduniowski Ł., Szychowski M., Tworuszka A.: Badanie efektywności klastrowych aplikacji równoległych algorytmów programowania dynamicznego", Zeszyty Naukowe PO, seria Informatyka, z. 2, 2005, str.:169-92.

Artykuł recenzowany