ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1993

Seria: INFORMATYKA z. 24 Nr kol. 1222

Leszek BORZEMSKI
Robert BŁAŻKÓW
Jacek WRONKA

WIELOETAPOWE PODEJMOWANIE DECYZJI W SIECI KOMPUTEROWEJ Z RÓWNOWAŻENIEM OBCIĄŻENIA*

Streszczenie. W pracy przedstawiono system wieloetapowego podejmowania decyzji zastosowany do rozpoznawania wieloetapowego. System pracuje w sieci TCP/IP typu Ethernet z komputerami PC wyposażonymi w system operacyjny Unix z dobudowanymi funkcjami równoważenia obciążenia.

THE MULTISTAGE DECISION-MAKING IN THE LOAD BALANCED LOCAL AREA NETWORK

Summary. In this paper we present a distributed load balanced computer system for supporting the multistage decision-making, namely for the multistage recognition. The system is running under Unix operating systems in the Ethernet TCP/IP-based local area network.

MULTIPLEXES ENTSCHEIDUNGSSYSTEM FÜR EIN LOKALES NETZWERK

Zusammenfassung. In diesem Artikel präsentieren wir ein Entscheidungssystem für die Unterstützung der verschiedenen Schritte des Entscheidungsprozesses. Dieses System läuft unter dem UNIX Betriebssystem in einem TCP/IP - basierendem lokalen Netzwerk.

* Pracę zrealizowano w ramach projektu Nr 3 0762 91 01 finansowanego w latach 1991-1994 przez Komitet Badań Naukowych.

1. Wprowadzenie

Wieloetapowe podejmowanie decyzji [1] charakteryzuje się realizacją sekwencji decyzjnych, które są organizowane w ramach określonego wspólnego schematu decyzyjnego opisanego drzewem decyzyjnym. Przestrzeń możliwych decyzji podlega zawężeniu z etapu na etap, aż do osiągnięcia pewnej pojedynczej decyzji, która jest ostatecznym wynikiem całego procesu decyzyjnego. Przykładem takiego postępowania decyzyjnego jest rozpoznawanie wieloetapowe RW [4].

W pracy opisano rozproszony system wieloetapowego podejmowania decyzji zbudowany z wykorzystaniem rozproszonego systemu równoważenia obciążenia [2]. Opracowany system wykorzystano do przeprowadzenia eksperymentów rozpoznawania wieloetapowego. System wykorzystuje lokalną sieć Ethernet komputerów PC z systemami operacyjnymi typu Unix.

2. Rozpoznawanie wieloetapowe

Rozpoznawanie wieloetapowe realizowane jest w następujący sposób. Na pierwszym etapie rozpoznawania następuje pomiar określonych cech x_0 opisujących rozpoznawany obiekt, które stanowią podstawę do podjęcia decyzji i_1. Decyzja ta, będąca wynikiem rozpoznawania na pierwszym etapie, określa pewien podzbiór w zbiorze wszystkich klas i jednocześnie wskazuje cechy x_{i_1} (spośród x), jakie należy zmierzyć, aby podjąć decyzję na etapie następnym. Na drugim etapie następuje pomiar cech x_{i_1}, które wraz z i_1 są podstawą do podjęcia kolejnej decyzji i_2. Decyzja ta, podobnie jak i_1, wskazuje cechy x_{i_2} potrzebne do podjęcia decyzji następnej (na trzecim etapie) oraz również podobnie, określa pewien podzbiór klas, z tym, że nie w zbiorze wszystkich klas, lecz w podzbiorze wskazanym przez decyzję i_1. Ogólnie, na etapie n-tym decyzja i_n, podjęta na podstawie zmierzonych cech x_{n-1} wyznaczonych przez poprzednią decyzję i_{n-1} określa podzbiór w zbiorze klas wskazanych przez poprzednią decyzję i_{n-1} oraz wyznacza cechy x_n potrzebne do podjęcia decyzji na etapie (n+1). Tak więc opisany proces decyzyjny ma zbieżny charakter, bowiem kolejno podejmowane decyzje określają coraz mniej liczny zbiór klas, a cała procedura kończy się etapem ostatnim (N-tym), na którym podjęta decyzja i_N wskazuje na pojedynczą klasę obiektów stanowiącą ostateczny wynik rozpoznawania. Reprezentacją schematu decyzyjnego

dla RW może być drzewo decyzyjne z węzłami wewnętrznymi typu OR, w których podejmowane są decyzje na danym etapie decyzyjnym. Jeśli węzłom terminalnym takiego drzewa decyzyjnego przypiszemy decyzje ostateczne, to otrzymamy pełen schemat wieloetapowego podejmowania decyzji. Każda pojedyncza decyzja związana jest z rozwiązaniem przez lokalny algorytm decyzyjny pewnego lokalnego problemu decyzyjnego, skojarzonego przez projektanta systemu z danym węzłem. Wieloetapowy proces decyzyjny rozpoczyna się zawsze od decyzji lokalnej w korzeniu drzewa. Algorytmy lokalne są implementowane w postaci odpowiednich niezależnych programów. Dla systemu przetwarzania programy te są zadaniami wymagającymi alokacji do procesorów, celem ich wykonania.

Przetwarzanie na potrzeby pojedynczej decyzji wieloetapowej polega na wykonaniu pracy będącej wielozadaniowym łańcuchem zadań, w którym pierwszym zadaniem jest zawsze program związany z węzłem - korzeniem drzewa decyzyjnego. Następne zadania w łańcuchu generowane są na bieżąco. Przechodzenie do następnego etapu przetwarzania dokonuje się zgodnie z regułą OR (tj. wybór jednej i tylko jednej z kilku możliwych dróg w drzewie decyzyjnym, począwszy od węzła bieżącego) - wówczas wykonywany jest program związany z pierwszym węzłem na tej drodze, tj. węzłem następnikiem węzła bieżącego (tak się dzieje, kiedy zgodnie z definicją drzewa taki węzeł następnik istnieje, w przeciwnym razie bieżący program jest ostatnim w łańcuchu tak przetwarzanych programów). Decyzja o wyborze drogi w drzewie znana jest dopiero po wykonaniu programu - poza sytuacjami przetwarzania ostatnich programów z łańcuchów. Pojedynczy łańcuch zadań tworzący pracę może mieć różną długość. Globalnym wynikiem przetwarzania pracy jest informacja o decyzji, która zapada przy przetwarzaniu ostatniego z programów w łańcuchu.

W praktycznych sytuacjach rozpoznawania stosujemy rozpoznawanie z uczeniem, wykorzystujące informację zawartą w ciągu uczącym $S_m = (x_1, j_1),(x_2, j_2),...,(x_m, j_m)$. Ciąg uczący składa się z m par. Każda para zawiera wektor wartości zmierzonych cech obiektu x_l oraz poprawną klasyfikację j_l, $l = 1,2,...,m$. W omawianym systemie zaimplementowany został algorytm k-najbliższych sąsiadów do rozpoznawania wieloetapowego przedstawiony w pracy [4], aczkolwiek należy zauważyć, że przy zastosowanej modułowej konstrukcji systemu algorytm ten może być w prosty sposób wymieniony na dowolny inny.

Lokalne algorytmy decyzyjne wykorzystują dwie podstawowe kategorie danych, a mianowicie informację o wartościach cech charakteryzujących obiekt (w postaci tzw. *wektora obiektu*) oraz dane zawarte we wspólnej dla wszystkich lokalnych decydentów bazie danych z ciagiem uczącym. Lokalny algorytm decyzyjny w węźle i sięga po podciąg S_{m_i} o długości m_i z ciągu uczącego S_m, a mianowicie ciąg $S_{m_i} = (x_{i,j_1}, j_1),(x_{i,j_2}, j_2),...,(x_{i,j_{m_i}}, j_{m_i})$ z obiektami

uczącymi zawierającymi tylko klasy osiągalne z węzła i. Wektor obiektu jest przekazywany pomiędzy poszczególnymi programami obliczeniowymi. Dostęp do bazy danych jest niezależny z każdego z programów. Ciąg uczący może być przechowywany w specjalnie do tego celu zaprojektowanej bazie danych [3]. Przy rozpoznawaniu wieloetapowym z uczeniem dzięki dekompozycji zadania rozpoznawania następuje zmniejszenie złożoności obliczeniowej pojedynczych zadań obliczeniowych (mniejsze ciągi uczące oraz mniejsze wektory cech w węzłach).

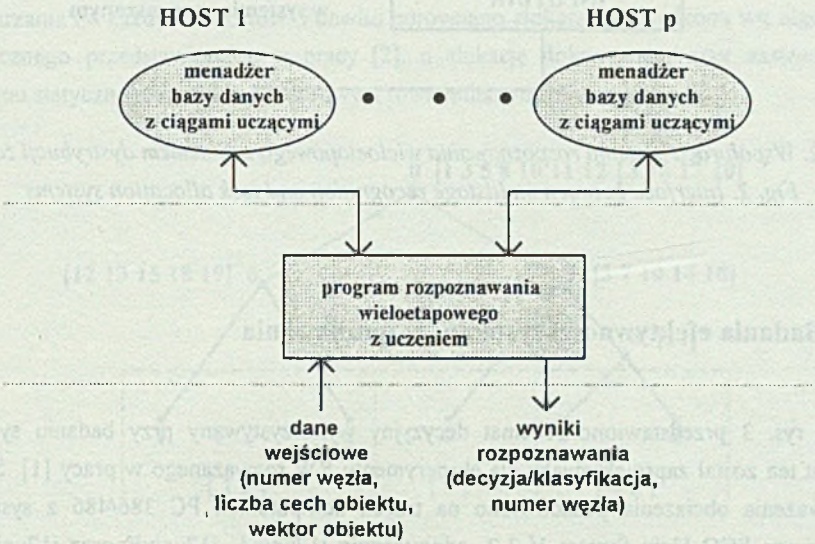
Przy komputerowej realizacji systemu rozpoznawania wieloetapowego istotne jest systematyczne przechowywanie danych i zapewnienie efektywnego dostępu do nich. W tym celu zaprojektowano i zaimplementowano bazę danych, która zawiera następujące informacje: opis struktury drzewa decyzyjnego (identyfikatory węzłów wewnętrznych i terminalnych, zależności pomiędzy węzłami, wektory nazw cech dla każdego węzła wewnętrznego), ciąg uczący oraz inne informacje potrzebne do podjęcia decyzji w każdym węźle wewnętrznym. Problemowo zorientowana baza danych znakomicie ułatwia komputerową realizację systemu wieloetapowego rozpoznawania z uczeniem w praktycznych zastosowaniach [3].

W szczególności, dla bezpośrednich celów obliczeniowych wykorzystywana jest ta część bazy, która zawiera ciąg uczący. Ciąg uczący zawarty w bazie danych jest podzielony na rozłączne podciągi fizycznie przechowywane w osobnych plikach. Poszczególne pliki, na żądanie programów uruchamianych w węzłach sieci, biorących udział w przetwarzaniu, mogą być transmitowane w sposób niezależny od siebie. W dalszej części przyjmiemy, że pliki te zostały uprzednio skonstruowane według metody zaproponowanej w pracy [1].

3. Rozpoznawanie wieloetapowego w rozproszonym systemie komputerowym z równoważeniem obciążenia

Opracowany system rozpoznawania wieloetapowego jest aplikacją umiejscowioną w warstwie aplikacji systemu rozproszonego z równoważeniem obciążenia w sieci komputerowej przedstawionego w pracy [2]. System rozproszonego rozpoznawania wieloetapowego składa się z tylu identycznie zorganizowanych programów, ile jest węzłów systemu rozproszonego (tzw. hostów). W każdym hoście program ten jest wykonywany dla zadanego węzła drzewa decyzyjnego z charakterystycznymi dla niego danymi wejściowymi. Rozpoznanie jednego obiektu to ciąg wywołań tego programu na tym samym lub na różnych hostach dla odpowiednich węzłów decyzyjnych, począwszy od węzła zerowego - korzenia, a skończywszy na węźle, w którym bezpośrednim następnikiem jest węzeł reprezentujący rozpoznaną klasę. Każde wykonanie programu generuje dwie dane wyjściowe określające następny krok decyzyjny. Pierwszą z nich jest numer węzła następnika, natomiast druga określa decyzję - jest

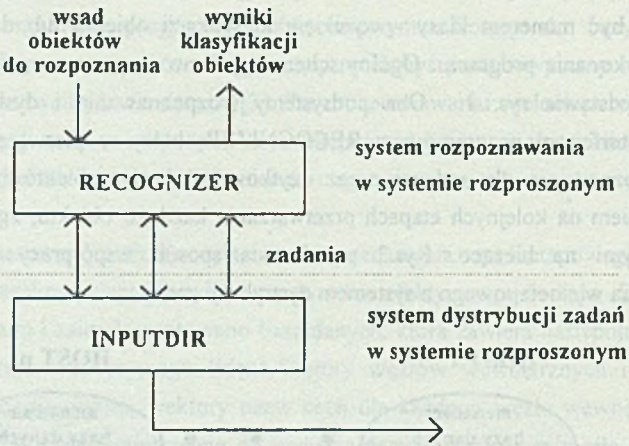
nią albo kolejny etap rozpoznawania albo klasyfikacja. W zależności od rodzaju decyzji numer węzła może być numerem klasy - wynikiem klasyfikacji obiektu lub daną wejściową do kolejnego wykonania programu. Ogólny schemat systemu rozpoznawania wieloetapowego z uczenim przedstawia rys. 1. Oba podsystemy (rozpoznawania i dystrybucji zadań) są połączone interfejsem programowym RECOGNIZER, który w pierwszej fazie uruchamia system rozpoznawania dla zadanej przez użytkownika liczby obiektów, a dalej zarządza rozpoznawaniem na kolejnych etapach przetwarzania każdego obiektu, zgodnie z decyzjami podejmowanymi na bieżąco. Rys.2 przedstawia sposób współpracy między systemem rozpoznawania wieloetapowego a systemem dystrybucji zadań.



Rys. 1. Schemat systemu rozpoznawania wieloetapowego z uczeniem

Fig. 1. General organization of the multistage recognition system

Wszystkie dane potrzebne do obliczeń (począwszy od ciągów uczących, poprzez strukturę drzewa decyzyjnego, aż obiektów do sklasyfikowania) są umieszczone w plikach. Stąd aby zmienić strukturę drzewa decyzyjnego, rozmieszczenie plików z podciągami uczącymi czy prawdopodobieństwa poprawnej klasyfikacji klas (dla celów wygenerowania odpowiedniego ciągu uczącego, gdy brak jest danych rzeczywistych), należy zmienić tylko zawartość odpowiednich plików konfiguracyjnych za pomocą dostarczonego interfejsu. Wśród tych plików czołową rolę odgrywają pliki z ciągiem uczącym. Zarządza nimi menedżer bazy danych. Jest to program (daemon) działający w tle, który oczekuje na żądania połączenia i transmisji zbiorów z bazy danych, pochodzące od programów - klasyfikatorów.



Rys. 2. Współpraca systemu rozpoznawania wieloetapowego z systemem dystrybucji zadań
Fig. 2. Interface between multistage recognition and task allocation systems

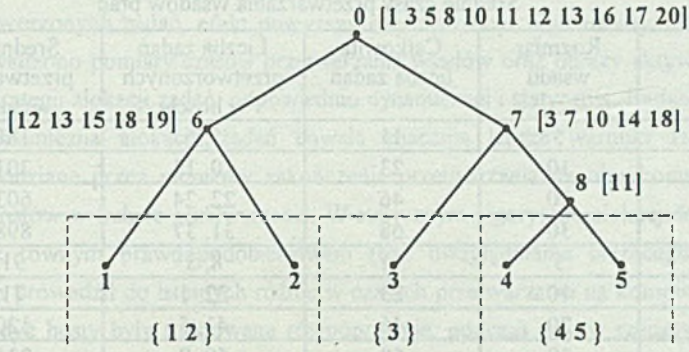
3. Badania efektywności systemu rozpoznawania

Na rys. 3 przedstawiono schemat decyzyjny wykorzystywany przy badaniu systemu. Schemat ten został zaprojektowany dla eksperymentu RW rozważanego w pracy [1]. System równoważenia obciążenia posadowiono na trzech komputerach PC 386/486 z systemem operacyjnym SCO Unix System V 3.2, odpowiednio i17unixA, i17unixB oraz i17unixC (o aliasach ua, ub, uc) w lokalnej sieci komputerowej w Instytucie Sterowania i Techniki Systemów Politechniki Wrocławskiej. Dwa komputery ub i uc miały zbliżoną wydajność, która była 6-7 mniejsza od mocy hosta i17unixA [2].

W każdym węźle drzewa decyzyjnego rozpatruje się zadany podzbiór cech ze zbioru 20 cech obiektów (numery wykorzystywanych cech w węźle podane są obok numeru węzła). Ciąg uczący o wielkości 200 obiektów (dla klas 1,2,3,4,5 odpowiednio po 20, 42, 46, 46, 46 elementów) zapamiętano w bazie danych składającej się z trzech plików z podciągami odpowiednio dla klas {1,2}, {3}, {4,5}. Wszystkie trzy pliki umieszczone zostały na jednym wybranym hoście i17unixA jako najszybszym i zawierającym najwięcej przestrzeni w pamięci dyskowej. Przy prowadzonych eksperymentach ciąg uczący był generowany sztucznie dla zadanej wielkości wektora cech oraz liczby obiektów uczących dla każdej klasy. Cechy generowano wg rozkladu normalnego N~(nr klasy, 0,6). Prawdopodobieństwa poprawnych

rozpoznań klas wyznaczane były automatycznie dla danego ciągu uczącego. Ciąg uczący można było też wprowadzić z zewnątrz. Klasyfikacja przebiegała według algorytmu k-najbliższych sąsiadów. Przyjęto, że k=5. Obiekty do sklasyfikowania były generowane według podobnego schematu, jak obiekty uczące.

Efektywność działania systemu oceniano pod kątem całkowitego czasu wykonania wsadu prac (czas liczony od momentu wygenerowania poczatkowego zbioru zadań do momentu powrotu wyników klasyfikacji wieloetapowych dla wszystkich obiektów, których dotyczył wsad). Poza tym oceniano współczynnik przyspieszenia obliczeń liczony jako stosunek czasu wykonania wsadu prac w systemie jednokomputerowym do czasu obliczeń w systemie sieciowym. Obserwowano liczbę zadań alokowanych do poszczególnych hostów oraz przebieg przetwarzania na każdym z hostów. Ponadto porównano alokację prowadzoną wg algorytmu dynamicznego przedstawionego w pracy [2], a alokację dokonywaną przy zastosowaniu algorytmu statycznego o rozkładzie losowym równomiernym.



Rys. 3. Przykładowa struktura drzewa decyzyjnego z opisem numerów cech w węzłach oraz plików bazy danych z podciągami uczącymi

Fig. 3. Sample decision tree with nodes labeled by features used at the nodes and the files of the learning sequence data base

W dalszej części pracy przedstawimy wybrane wyniki pomiarów efektywności pracy systemu. Czas wykonania wsadu z jedną pracą (dla wybranego obiektu - klasyfikacja w 3 etapach) wynosił dla hostów ua, ub i uc, odpowiednio 18,9; 105,3; 103,3 sek. Dla wsadu prac składającego się z szeregu współbieżnie wykonywanych zadań w systemie operacyjnym Unix następuje zmniejszenie czasu potrzebnego średnio na wykonanie zadania. Jest to wyniki nakładania się czasów wykonywania zadań, co eliminuje częściowo narzuty związane z uruchamianiem kolejnych zadań.

Tabela 1 przedstawia wyniki przetwarzania wsadów prac o różnych rozmiarach dla konfiguracji dwu- i trzykomputerowych. W przypadku komputerów identycznych (lub prawie identycznych) można oczekiwać równego podziału zadań między komputerami. Współczynnik przyspieszenia czasu wykonania wsadu zadań, powinien dążyć do swojej wartości idealnej osiągalnej dla zbioru niezależnych zadań przy podziale symetrycznym, równej liczbie komputerów w sieci (przy tzw. symmetric load balancing). W eksperymencie z dwoma komputerami i17unixB i i17unixC o porównywalnej mocy należało oczekiwać ok. dwukrotnego przyspieszenia czasu wykonania wsadu w porównaniu z czasami uzyskanymi dla jednego z tych dwóch komputerów. Wynosiło ono dla badanych hostów od 1,51 do 1,96, przy rozmiarze wsadu prac wynoszącym od 5 do 30 prac. Oba komputery biorące udział w przetwarzaniu były obciążone mniej więcej równą liczbą zadań (wszystkie zadania były porównywalnych rozmiarów), uzyskano więc zrównoważenie obciążenia. Uzyskane wyniki potwierdziły skuteczność opracowanego mechanizmu równoważenia obciążenia.

Tabela 1

Średnie czasy przetwarzania wsadów prac

Hosty	Rozmiar wsadu	Całkowita liczba zadań	Liczba zadań przetworzonych w hoście	Średni czas przetwarzania wsadu [s]
ub, uc	5	11	5, 6	188,5
ub, uc	10	23	10, 13	301,5
ub, uc	20	46	22, 24	603,2
ub, uc	30	68	31, 37	898,4
ua, ub	5	11	8, 3	91,8
ua, ub	10	23	22, 1	71,8
ua, ub	20	46	41, 5	120,1
ua, ub	30	68	59, 9	231,5
ua, ub, uc	5	11	6, 4, 1	152,7
ua, ub, uc	10	23	17, 2, 4	105,1
ua, ub, uc	20	46	32, 3, 11	485,2
ua, ub, uc	30	68	45, 7, 16	615,3

Wyniki dają podstawę do następujących wniosków dla przypadków systemów niejednorodnych. Widać, że system nie osiągał maksymalnej efektywności, bowiem z punktu widzenia hosta szybszego (i17unixA) czas przetwarzania wsadu albo się nie zmienił, albo wręcz wzastał, pomimo, że teoretyczna moc obliczeniowa całego systemu (równa sumie mocy obliczeniowych poszczególnych komputerów) była większa od mocy tego komputera. Wynika to z istoty funkcjonowania systemu rozproszonego, w którym zadania są niepodzielnymi jednostkami, w całości przetwarzanymi w pojedynczym węźle systemu. W przypadku dużych dysproporcji między potencjałem obliczeniowym węzłów systemu, jak w rozpatrywanym eksperymencie, widać, że teoretyczny potencjał hostów wolniejszych może być praktycznie

bez znaczenia, i lepiej byłoby, z punktu widzenia całego systemu, dociążać zadaniami tylko najszybszy węzeł, a hosty wolniejsze wykorzystywać w sytuacjach, gdy niezależne od systemu dystrybucji zadań obciążenie najmocniejszego hosta wzrośnie, przez co początkowe dysproporcje pomiędzy hostami ulegną zmniejszeniu (w granicznym przypadku ulegną zniwelowaniu). Porównując liczbę zadań wykonanych przez hosty widać, że proporcja między nimi odzwierciedla stosunek czasów wykonania na nich pojedynczych zadań. System nie gwarantuje konkretnej kolejności alokacji zadań do węzłów sieci, co może prowadzić do nierówności czasów zakończenia przetwarzania przez poszczególne węzły. Zaobserwowano bowiem przypadki, w których system alokował jedno z ostatnich zadań, wygenerowanych w procesie przetwarzania, do wolniejszego hosta, przez co całkowity czas przetwarzania wydłużał się. Wynikało to z tego, że konstrukcja algorytmu alokacji zadań nie zakładała znajomości środowiska systemu zadań, a w szczególności znajomości czasów wkonywania zadań ani liczby zadań pozostających jeszcze do wykonania. Było to szczególnie widoczne dla niewielkich wsadów zadań. Gdy system działał w dłuższym odcinku czasu i przy dużej liczbie przetworzonych zadań, efekt powyższy częściowo był niwelowany.

Przeprowadzono pomiary czasów przetwarzania wsadów oraz obrazy aktywności hostów dla dwóch strategii alokacji zadań, odpowiednio dynamicznej i statycznej. Badano wsady o 20 pracach. Dynamiczna alokacja zadań dawała znacznie lepsze warunki zrównoważenia obciążenia widziane przez momenty zakończenia przetwarzania na obu komputerach oraz aktywność hostów w trakcie przetwarzania. W statycznym algorytmie alokacji każdy host był wybierany z równym prawdopodobieństwem (bez uwzględniania bieżącego obciążenia). Algorytm ten prowadził do istotnych różnic w czasach przetwarzania na komputerach (jest to oczywiste, gdyż hosty były traktowane równoprawnie, podczas gdy w rzeczywistości miały one różną moc obliczeniową) oraz dużo większych odcinków czasu, w których hosty były nieaktywne.

5. Uwagi końcowe

W pracy przedstawiono rozproszony system wieloetapowego podejmowania decyzji. Przykładową aplikację zrealizowano dla rozpoznawania wieloetapowego. System wspiera dowolny algorytm rozpoznawania - w wersji testowanej zaimplementowano algorytm 5 najbliższych sąsiadów. System zrealizowano w środowisku lokalnej sieci komputerów Unix z równoważeniem obciążenia. Opracowany system przetwarza zadania na potrzeby aplikacji dążąc do zrównoważenia obciążenia hostów w sieci. Najlepsze zrównoważenie obciążenia

uzyskuje się dla systemu homogenicznego. W przypadku systemu niehomogenicznego, przy dużych dysproporcjach w mocach obliczeniowych poszczególnych hostów, obciążenie hostów może znacznie odbiegać od stanu zrównoważenia. Z punktu widzenia słabszych komputerów w sieci, system daje znaczne korzyści, gdyż pozwala na zmniejszenie czasu przetwarzania wsadu prac, dzięki przeniesieniu obciążenia na inne hosty w sieci. Korzyść ta nie zawsze jest obserwowana z punktu widzenia mocnych hostów. Poprawę wskaźnika wykorzystania zasobów systemu można by oczekiwać przy zastosowaniu strategii alokacji zadań z dopuszczeniem ich dalszej migracji (tzn. gdyby host mógł być wywłaszczony z wcześniej alokowanego do niego zadania, w sytuacji gdy w pewnej chwili znalazł się w systemie host o zerowej kolejce wejściowej).

LITERATURA

[1] Borzemski L., File allocation in a distributed knowledge-based multistage decision making system, *Syst. Sci*, vol. 18, no. 1-2, 1992.

[2] Borzemski L., Głowacz S., Wojtczak K., System równoważenia obciążenia w sieci komputerów Unix, Zeszyty Naukowe Politechniki Śląskiej, Seria: Informatyka z. 24, 1993, 193-204

[3] Borzemski L., Lebiediewa S., Baza danych dla wieloetapowego podejmowania decyzji w procesie rozpoznawania obiektów, Materiały II Krajowej Konferencji Inżynieria Wiedzy i Systemy Ekspertowe, Wrocław 1993.

[4] Kurzyński M., Algorytmy rozpoznawania wieloetapowego oraz ich zastosowania medyczne i techniczne. Prace Nauk. Inst. Ster. i Techn. Systemów Polit. Wrocł., Nr 5, Seria Monogr. nr 3, 1987.

Recenzent: Dr inż. Krzysztof Nałęcki

Wpłynęło do redakcji 20 września 1993 r.

Abstract

Here we considered the multistage decision-making (MDM) that is a widely used approach for supporting complex decision problems. Then instead of making a single and usually computationally complex decision, we deal with a series of simpler decisions that are organized on the basis of the OR tree decomposition. Each MDM request is executed by dynamically created multiple task job. Typically, MDM requests are processed in batches of requests. In batch processing of MDM requests the set of different in length and resource requirements multiple task jobs have to be processed. The multistage decision-making can be naturally organized in distributed computer system, as for example for the needs of distributed computer-aided testing and diagnosis. Such systems are especially attractive in microcomputer based local area networks with Unix workstations. In distributed computer systems there is a need for load control to improve the system performance. The load balancing approach is investigated in the environment of the multistage decision-making, namely for the multistage recognition. The system is implemented in the Ethernet based local area network with Unix systems. In the paper the results of testbeds are presented. The results show the advantages in performance when applying the distributed load balanced approach to the MDM processing.