ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1996

Seria: INFORMATYKA z. 30 Nr kol. 1315

Leszek BORZEMSKI
Arkadiusz KIEDA
Krzysztof NIEKRASZ
Jerzy ŚWISTEK

SYSTEM RÓWNOWAŻENIA OBCIĄŻĘŃ W SIECI WINDOWS NT

Streszczenie. W pracy przedstawiono opracowany w Instytucie Sterowania i Techniki Systemów Politechniki Wrocławskiej system równoważenia obciążenia komputerów w sieci Windows NT. System rozprasza wykonanie dowolnych zarejestrowanych w systemie zadań wsadowych, dążąc do uzyskania zrównoważenia obciążeń komputerów w sieci, korzystając z jednej z zaimplementowanych strategii alokacji zadań. Zadania obliczeniowe mogą być wprowadzone do systemu na dowolnym komputerze w sieci. System zastosowano jako sieciową platformę obliczeniową metakomputera do rozproszonego, wieloetapowego rozpoznawania obiektów.

WINDOWS NT BASED LOAD BALANCING SYSTEM

Summary. In the paper we present the system for load balancing in the network of Windows NT based computers. The system allocates batch tasks that have been previously registered and uses one of implemented task allocation strategies. Tasks can arrive at any computer in the network. The system has been used as a metacomputer platform for the needs of the multistage recognition.

SYSTÈME D'ÉQUILIBRAGE DE CHARGES DANS LE RÉSEAU WINDOWS NT.

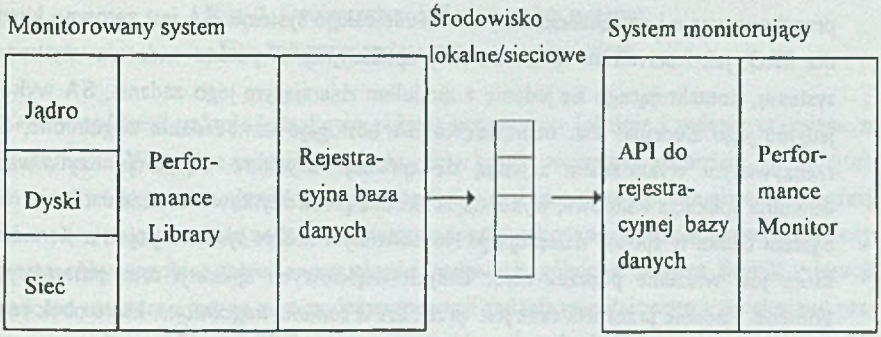
Résumé. Le système assure la dispersion de l'exécution de tâches arbitraires par lots, en tendant d'obtenir une mise en équilibre des charges des ordinateurs dans le réseau, et en utilisant dans ce but une parmi plusieurs stratégies implantées d'allocation des tâches. L'introduction des tâches de calcul dans le systéme peut avoir lieu dans un ordinateur quelconque. Le système a été utilisé dans la reconnaissance multi-étapes d'objets en tant que platforme de calcul, en forme d'un méta-ordinateur avec plusieurs unités d'exécution fonctionnant dans un réseau d'ordinateurs.

1. Wprowadzenie

W Instytucie Sterowania i Techniki Systemów Politechniki Wrocławskiej od kilku lat prowadzone są prace badawczo-rozwojowe w zakresie systemów równoważenia obciążenia w lokalnych sieciach komputerowych. W ramach tych prac powstał m.in. system równoważenia obciążenia dla komputerów wyposażonych w system operacyjny SCO UNIX [2, 3, 4]. Niniejsza praca prezentuje nowe rozwiązanie, którym jest system równoważenia dla Windows NT. System ten powstał z myślą zbudowania sieciowej platformy obliczeniowej o własnościach metakomputera [9]. W Instytucie funkcjonuje sieć komputerowa Windows NT, złożona z 12 stacji roboczych Windows NT 3.1 klasy PC 386/486, pracująca we wspólnej domenie pod kontrolą systemu Windows NT Advanced Server. System zarządza alokacją zadań obliczeniowych w sieci udostępniając każdemu z użytkowników stacji roboczych Windows NT wielokomputerową maszynę wirtualną, pozwalającą na wykorzystanie idei metaprzetwarzania. Zadania alokowane są pod kątem uzyskania zrównoważenia obciążenia systemu komputerów uczestniczących w przetwarzaniu. System pracuje „w tle", pozwalając użytkownikowi na wykorzystanie jego stacji roboczej do przeźroczystego wykonywania zadań innych użytkowników. System przystosowany jest do pracy w sieci zbudowanej z maszyn o różnej mocy. Maszyny o większej mocy preferowane są przy przydziale zadań. Możliwe jest dedykowanie maszyn do realizacji jedynie niektórych zadań. Rozpraszaniu po zarejestrowaniu podlegają dowolne zadania wsadowe. System napisany został w *Microsoft Visual C++ 1.5* dla Windows NT i wykorzystuje mechanizmy komunikacyjne i systemowe Windows NT. Szczególnie godny uwagi jest unikatowy w skali systemów operacyjnych mechanizm *Performance Library*, pozwalający na ocenę obciążenia dowolnego komputera w sieci Windows NT.

2. Pomiar obciążenia w środowisku Windows NT

Nasz system wykorzystuje mechanizm monitorowania zasobów w identyczny sposób jak to robi aplikacja *Performance Monitor*, będąca elementem systemu Windows NT (rys. 1). Pomiar obciążenia realizujemy z wykorzystaniem funkcji biblioteki *Performance Library*, za pomoccą których można monitorować działanie systemu Windows NT, w tym stopień wykorzystania jego zasobów, takich np. jak: procesory, pamięć RAM, urządzenie we/wy [7]. Informacje o zasobach systemu zapamiętywane są w *rejestracyjnej bazie danych* (ang. Registry). Informacje te nie są tam zapisane na stałe, ale są tam umieszczane w momencie aktywacji odpowiednich funkcji z biblioteki Performance Library, które uruchamiają mechanizm testowania wykorzystania zasobów. Możliwy jest dostęp do baz rejestracyjnych komputera zarówno lokalnego, jak i zdalnego (o ile nie kłóci się to z zasadami bezpieczeństwa). Stąd z jednego komputera możemy monitorować sytuację całej sieci.



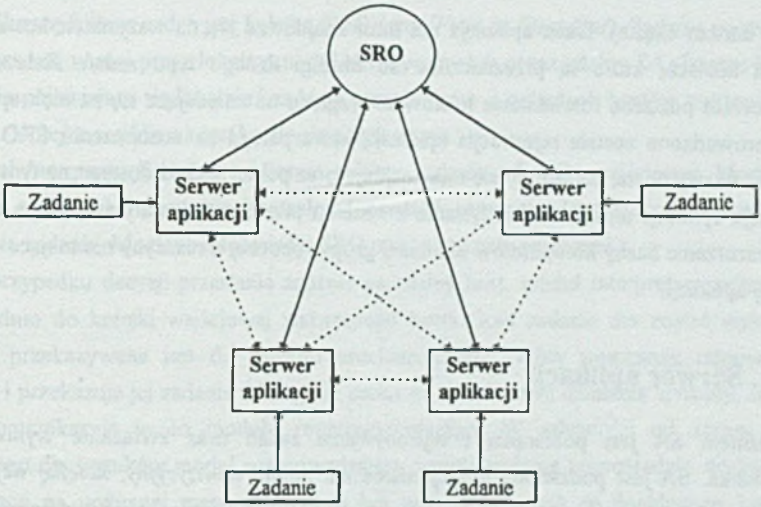
Rys. 1. Monitorowanie zasobów w systemie Windows NT [7]
Fig. 1. Performance monitoring in Windows NT [7]

3. Architektura i działanie systemu

Przyjęto następujące założenia dotyczące środowiska sprzętowego i systemowego:

- System działa w sieci lokalnej na komputerach połączonych szybkim medium transmisyjnym (wdrożenię dotyczyło sieci Ethernet z szybkością transmisji do 10 Mb/s).
- Wszystkie komputery są klasy IBM PC i działają pod kontrolą systemu Windows NT (wdrożenie dotyczyło wersji 3.1).
- Każdy komputer dysponuje własnym systemem plików.

- Budowę systemu oparto na architekturze klient/serwer. Architektura ta zapewnia optymalne wykorzystanie komputerów osobistych współpracujących z serwerami aplikacji oraz z serwerami baz danych. W naszym systemie klient i serwer rozumiane są nie jako maszyny, ale jako aplikacje.
- Decyzje o alokacji zadań podejmowane są centralnie, tzn. wybór procesora dla zadań pojawiających się w systemie dokonywany jest przez jeden program zarządzający, nazwany *serwerem równoważenia obciążenia (SRO)*. Przepływ zadań pomiędzy hostami odbywa się bezpośrednio, bez udziału SRO. W systemie znajduje się jeden SRO oraz wiele *serwerów aplikacji (SA)*. SA to program zbierający zadania i zarządzający ich wykonaniem. Zadania zgłaszane są do lokalnego SA, a pojawiają się na wskazanej przez SRO maszynie pracującej pod kontrolą systemu i zgłaszane są do wykonania do właściwego SA (rys. 2).
- Odnosząc konstrukcję systemu do architektury klient/serwer, serwerem jest SRO. Wykonuje on usługi zlecone przez klientów, tj. SA. SRO odpowiedzialny jest również za przechowywanie i zarządzanie danymi o stanie całego systemu. SA jest zarówno klientem dla SRO, jak i serwerem zadań warstwy aplikacyjnej. Z punktu widzenia użytkownika systemu, kontaktującego się jedynie z modułem zbierającym jego zadania, SA wykonuje jedynie jego zlecenia. Tak naprawdę to SA obsługuje tzw. zadania uogólnione, a ich rzeczywistym wykonaniem zajmują się *aplikacje użytkowe*. Aplikacją użytkową jest dowolna aplikacja wsadowa, wykonująca zlecone przez użytkownika zadania.
- System działa w sposób następujący. Na dowolnym hoście systemu pojawia się *zadanie*, które jest widziane poprzez zbiór danych wejściowych aplikacji oraz parametry linii komend. Zadanie przekształcane jest przez SA w *zadanie uogólnione*, które obok samego zadania zawiera informację o aplikacji użytkowej, której zlecane jest wykonanie zadania oraz informację,gdzie mają być przesłane wyniki zadania. SA wysyła następnie zapytanie do SRO o miejsce wykonania zadania. SA może uzyskać jedną z czterech odpowiedzi:
  - zadanie należy wykonać lokalnie. W takim przypadku SA uruchamia odpowiednią aplikację użytkową przekazując jej zadanie;
  - zadanie należy wykonać zdalnie. W takim przypadku SRO zwraca w odpowiedzi nazwę hosta, który wybrany został do wykonania zadania. SA wysyła zadanie do wskazanego hosta;
  - zadanie musi poczekać na wykonanie z powodu zbyt wysokiego obciążenia systemu. Zadanie pozostawiane jest na hoście, zaś SA przechodzi do obsługi następnego zadania;
  - nie został znaleziony host, który mógłby wykonać zadanie. Zadanie kasowane jest z kolejki wejściowej SA, zaś do katalogu docelowego wysyłany jest plik o nazwie „error.log", który zawiera informację o braku możliwości wykonania zadania.



Rys. 2. Ogólny schemat funkcjonalny systemu
Fig. 2. System architecture - a general view

Serwer aplikacji traktuje jednakowo zadania generowane lokalnie i zadania przysłane ze zdalnego hosta. Wszystkie zadania trafiają do jednej kolejki wejściowej na hoście. W podejściu takim możliwe jest zatem zjawisko migracji zadań w systemie. Wystąpienie zjawiska migracji zadań może powodować, że zadanie nie zostanie nigdy wykonane, wędrując po sieci od hosta do hosta. Aby zapobiec takiej ewentualności zadanie uogólnione zawiera licznik migracji. Licznik ten ustawiany jest na zero podczas generacji zadania uogólnionego i zwiększany przy każdym przesłaniu zadania. SRO porównuje ten licznik z limitem migracji. W przypadku, kiedy licznik migracji przekroczy limit migracji, zadanie zostaje wykonane lokalnie.

Wyniki działania aplikacji przetwarzającej zadanie odsyłane są do katalogu docelowego zdefiniowanego przez użytkownika na dowolnym komputerze w sieci.

Jak już wspomniano wcześniej, system obsługuje wyłącznie aplikacje wsadowe, tzn. aplikacje, które rozpoczynają swoje działanie pobierając parametry wywołania z linii komend, zaś kończą wykonanie po zakończeniu części obliczeniowej. System nie obsługuje aplikacji interakcyjnych. Z tego względu system nadaje się do przetwarzania zadań kompilacji, przetwarzania plików graficznych oraz zadań obliczeniowych.

System przesyła wyłącznie zadania dla aplikacji, nie przesyła zaś samych aplikacji rozumianych jako pliki binarne. Przesyłanie aplikacji nie miałoby sensu ze względu np. na specyfikę instalacji aplikacji, jej wielkość czy też wielkość wolnego miejsca na dysku docelowym. Aby zwiększyć elastyczność systemu przyjęto założenie, że aplikacje wykonujące zadania nie muszą znajdować się na wszystkich hostach systemu (w rzeczywistości jest to

sytuacja bardzo częsta). Dana aplikacja nie musi znajdować się na wszystkich hostach, ale tylko na hostach, które są przeznaczone do obsługi danego typu zadań. Założenie to spowodowało potrzebę rozróżnienia hostów ze względu na znajdujące się na nich aplikacje. Stąd wprowadzona została rejestracja aplikacji, która polega na dostarczeniu SRO nazwy aplikacji i nazwy hosta, na którym się ona znajduje oraz pełnej ścieżki dostępu na tym hoście. Rejestracja aplikacji umożliwia korzystanie z systemu przez więcej niż jedną grupę roboczą oraz rozszerzenie liczby komputerów dla danej grupy roboczej o maszyny nie mające dostępu do danej aplikacji.

3.1. Serwer aplikacji

Zadaniem SA jest pobieranie i wykonywanie zadań oraz zwracanie wyników do użytkownika. SA jest podzielony funkcjonalnie na: *moduł akwizycyjny, kolejkę wejściową, moduł interpretacji zadań* i *moduł rozprowadzający*. *Moduł akwizycyjny* odpowiedzialny jest za przyjmowanie zadań zgłaszanych do systemu. Przekształca on zadanie w *zadanie uogólnione* i umieszcza je w *kolejce wejściowej*, która jest katalogiem dyskowym. Moduł akwizycyjny potrzebuje następujących danych do stworzenia zadania uogólnionego (w nawiasach podano przyjęte terminy anglojęzyczne):

- lista plików, z których składa się zadanie oraz katalog, w którym one znajdują się (*Files, SourceDir*),
- typ zadania, czyli identyfikator aplikacji użytkowej, która ma zadanie wykonać (*AppId*),
- parametry aplikacji użytkowej, czyli linia komend, z którą aplikacja zostanie wywołana (*Options*),
- maszyna i katalog, gdzie mają być dostarczone wyniki (*DestHost, DestDir*),
- informacja czy razem z wynikami zwracać pliki zadania dla aplikacji, które modyfikują pliki zadania nie zmieniając ich nazwy (*SendBack*).

Moduł akwizycyjny pracuje w dwóch trybach: interakcyjnym i bezpośrednim (automatycznym). W trybie interakcyjnym, aby wykonać zadanie, użytkownik ręcznie wpisuje niezbędne dane w odpowiednim okienku dialogowym do jednorazowego uruchomienia zadania lub definiuje tzw. plik kontrolny wsadu z rozszerzeniem *BCF* (*ang. batch control file*), który zawiera informacje niezbędne do uruchomienia wsadu zadań.

W trybie bezpośrednim SA oczekuje na połączenie z aplikacją, która zażąda rozproszenia wsadu swoich zadań. Aplikacja taka musi dostarczyć pełnej informacji potrzebnej do stworzenia zadania uogólnionego. Metoda łączenia aplikacji z serwerem splikacji oparta została na mechanizmie nazwanych potoków (*ang. Named Pipe*). Rozwiązanie takie wymaga przystosowania aplikacji do współpracy z systemem poprzez dołączenie funkcji komunikacyjnej definiującej zadanie o nazwie AutoGenJob.

Kolejka wejściowa zadań jest kolejką FIFO (*ang. First-In-First-Out*). Zadania uogólnione umieszczane są w niej przez lokalny moduł akwizycyjny lub przez zdalny SA. Istotne jest to, że zadania pojawiające się lokalnie i zadania przychodzące z odległych hostów traktowane są tak samo. Kolejka wejściowa jest katalogiem dyskowym.

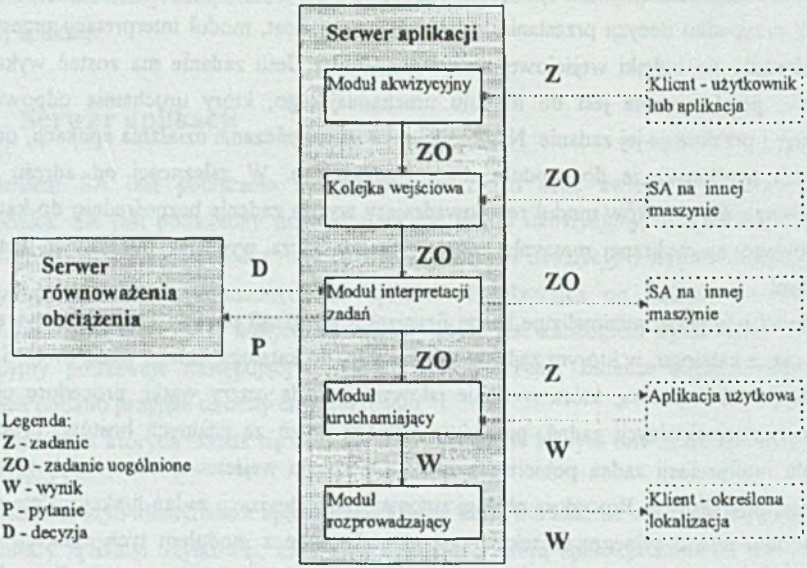
Moduł interpretacji zadań cyklicznie pobiera zadania z kolejki wejściowej. Moduł ten odczytuje niezbędne informacje kontrolne i przesyła zapytanie do SRO o miejsce (host) wykonania zadania. Możliwe odpowiedzi SRO opisane zostały wcześniej.

W przypadku decyzji przesłania zadania na zdalny host, moduł interpretacji przesyła je bezpośrednio do kolejki wejściowej wskazanego hosta. Jeśli zadanie ma zostać wykonane lokalnie, przekazywane jest do modułu uruchamiającego, który uruchamia odpowiednią aplikację i przekazuje jej zadanie. Następnie czeka na zakończenie działania aplikacji, odbiera wyniki i przekazuje je do modułu rozprowadzającego. W zależności od adresu hosta docelowego dla wyników moduł rozprowadzający wysyła zadanie bezpośrednio do katalogu docelowego na wybranej maszynie zdalnej lub umieszcza wynik w docelowym katalogu lokalnym.

Model powyższy minimalizuje liczbę fizycznych przepisań plików. Pliki przepisywane są tylko raz, z katalogu, w którym zadanie się pojawiło, do katalogu kolejki wejściowej. SA jest aplikacją wielowątkową, która w stanie jałowym posiada cztery wątki: procedurę obsługi automatycznej akwizycji zadań, procedurę odbioru zadań ze zdalnych hostów, procedurę modułu interpretacji zadań pobierającą zadania z kolejki wejściowej oraz wątek główny - aplikację interakcyjną. Procedura obsługi automatycznej akwizycji zadań funkcjonalnie należy do modułu akwizycyjnego. Oczekuje ona na połączenie z modułem tych aplikacji, które przystosowane są do bezpośredniego przesyłu wsadu swoich zadań do systemu. Procedura ta współpracuje z funkcją AutoGenJob. Działanie jej można opisać w sposób następujący: utworzony zostaje nazwany potok, po czym procedura wchodzi w stan „uśpienia" w oczekiwaniu na połączenie. Jeśli na tym samym komputerze wywołana zostanie funkcja AutoGenJob, procedura wychodzi ze stanu „uśpienia", tworzy nowy wątek oczekujący na kolejne połączenie po czym odbiera informacje przesłane przez funkcję AutoGenJob. Po zakończeniu nadawania procedura wywołuje procedurę tworzenia zadania uogólnionego, a po jej zakończeniu kończy swoje działanie zamykając jednocześnie swój wątek.

Opiszemy teraz, jak wygląda wykonanie zadania. Moduł interpretacji zadań rozpakowuje zadanie uogólnione do tymczasowego katalogu roboczego i tworzy wątek obsługujący zadanie, po czym wraca do monitorowania kolejki wejściowej. Rozpakowanie jest czynnością odwrotną do generacji zadania uogólnionego i polega na utworzeniu z jednego pliku zadania uogólnionego plików wejściowych aplikacji użytkowej, czyli zadania. Następnie wywoływana jest aplikacja użytkowa z odpowiednimi opcjami przez funkcję CreateProcess. Procedura wykonująca zadania wchodzi w stan „uśpienia" w oczekiwaniu na zakończenie wykonywania

aplikacji. Następnie zwraca wyniki do katalogu docelowego na docelowym hoście. Tymczasowe katalogi robocze są usuwane, po czym procedura kończy swoje działanie i zamyka swój wątek. SA przesyła do SRO komunikat przed i po zakończeniu wykonania zadania. Historia działania systemu jest zapamiętywana w specjalnym pliku kontrolnym. Każdy SA zgłasza też do SRO zarówno swój start, jak i zakończenie pracy. SRO posiada zawsze aktualne informacje o wszystkich działających w danej chwili SA.



Rys. 3. Schemat działania Serwera Aplikacji
Fig. 3. The operation of the SA module

3.2. Serwer równoważenia obciążeń

Serwer równoważenia obciążeń (SRO) jest aplikacją sterującą i kontrolującą pracę całego systemu. Od tej aplikacji zależy, gdzie zostaną skierowane zadania obsługiwane przez system i czy w ogóle możliwe będzie ich wykonanie. Dla tego celu SRO na bieżąco monitoruje i przeprowadza zgrubną analizę statystyczną obciążenia sieci Windows NT.

SRO ma budowę modularną i składa się z modułów: *pomiaru obciążenia, decyzyjnego, kontrolnego, zbierania danych i statystyki*. Oprócz wskazanych modułów SRO zawiera jeszcze dwa zasoby. Są to: dane organizacyjne, spisy aplikacji i maszyn (hostów) pracujących w systemie, wartości obciążenia systemu dla każdej z maszyn (hostów).

Operator systemu komunikuje się tylko z modułem organizacyjno-kontrolnym, posiadającym interfejs użytkownika. Ingerencje operatora w działanie systemu prawie zawsze mają natychmiastowy skutek. Pozwala to dynamicznie zmieniać parametry systemu. SRO może też pracować bez ingerencji operatora (poza włączeniem programu, oczywiście). System działa automatycznie i samodzielnie podejmuje decyzje.

Moduł kontrolny to część *SRO* odpowiedzialna za zainicjowanie pracy programu oraz za interakcyjną współpracę z operatorem. Wprowadzane parametry podzielić można na dwie grupy: parametry dotyczące strategii działania i parametry dotyczące rejestracji. Parametry dotyczące strategii działania można zmieniać dynamicznie i zmiany te odnoszą natychmiastowy skutek. Strategie są omówione przy opisie modułu decyzyjnego. Druga grupa parametrów ma charakter statyczny. Oznacza to, że chociaż można je zmieniać w czasie pracy programu, to zmiany zaczną obowiązywać po ponownym starcie SRO.

Moduł pomiaru obciążenia wykonuje cyklicznie pomiary obciążenia wszystkich maszyn, na których pracują SA. Częstość zbierania danych (takt pomiarowy Δt) jest parametrem SRO. Zbierane dane zapisywane są w tablicy obciążeń.

Spośród wielu obiektów Windows NT do oceny obciążenia maszyny wykorzystano obiekt *System*. Pozwala on ocenić obciążenie całej maszyny. Uwzględnione są wszystkie procesory dla maszyn wieloprocesorowych. Obiekt *System* ma kilkanaście mierników w oryginalnej literaturze dotyczącej Windows NT, nazywanych licznikami (*ang. counter*) [7]. W SRO zaimplementowano przede wszystkim liczniki charakteryzujące obciążenia procesora, a także dwa charakteryzujące obciążenie całego systemu:

- procentowy całkowity czas uprzywilejowany (*ang. Total Privleged Time*). Jest to średni procent czasu, jaki zużywają wszystkie procesory w trybie uprzywilejowanym (*ang. Privileged Mode*);
- procentowy całkowity czas procesora (*ang. Total Processor Time*). Jest to średni procent czasu, jaki wszystkie procesory zużywają na wykonywanie wątków, które nie są jałowe (*ang. idle*). Jest to część czasu procesora zużywana na użyteczną z punktu widzenia systemu i aplikacji pracę. Pozostały czas zużywany jest na obsługę wątku jałowego procesu jałowego, który z założenia jest tworzony dla każdego zainstalowanego w systemie procesora;
- procentowy całkowity czas użytkownika (*ang. Total User Time*). Jest to średni procent czasu, jaki zużywają wszystkie procesory pracując w trybie użytkownika (*ang. User Mode*);
- przełączenia kontekstu na sekundę (*ang. Context Switches/sec*). Jest to ilość przełączeń pomiędzy wątkami pracującymi w całym systemie. Z każdym wątkiem skojarzony jest kontekst. Im więcej wątków, tym więcej przełączeń między kontekstami. Każdy procesor

systemu w danej chwili może obsługiwać tylko jeden wątek, musi więc następować ciągłe przełączanie pomiędzy wszystkimi wątkami pracującymi na danym procesorze;

- odwołania systemowe na sekundę (*ang. System Calls/sec*). Częstotliwość odwołań do funkcji systemowych Windows NT dotyczących synchronizacji, zarządzania pamięcią, dostępu do urządzeń (poza urządzeniami graficznymi);
- całkowita ilość przerwań na sekundę (*Total Interrupts/sec*). Częstotliwość przerwań sprzętowych generowanych przez zegar systemu, karty sieciowe, urządzenia peryferyjne, linie komunikacyjne itp.

Wartości wszystkich tych mierników są na bieżąco zapamiętywane w tablicach obciążeń, w trzech postaciach: *surowej* (*ang. raw*), *uśrednionej* (*ang. average*), *zmodyfikowanej* (*ang. modified*),

Postać surowa licznika v^r to po prostu wartość licznika, jaką otrzymano z pomiaru po przeliczeniu wartości 32-bitowej, odczytanej z rejestracyjnej bazy danych systemu Windows NT na wartość specyficznego typu, np. wartość procentową. Taki licznik może być bardzo czuły na chwilowe zmiany obciążenia maszyny. Aby zmniejszyć tę czułość wprowadzono liczniki średnie i zmodyfikowane. Dodatkowo, aby rozróżnić maszyny o różnej mocy, wprowadzono współczynnik mocy maszyny pracującej w systemie.

Zamiast wartości surowej licznika v^r rozpatrywać będziemy wartość znormalizowaną $v^n = f_p\, v^r$. Normalizacja mocy maszyn wprowadzona została z myślą o odciążeniu maszyn słabszych i preferowaniu przy wyborze maszyn o większej mocy. Współczynnik f_p otrzymujemy empirycznie dla każdej maszyny, uruchamiając na niej specjalnie przygotowany test.

Test zawiera blok operacji całkowitoliczbowych (dodawanie), blok operacji zmiennoprzecinkowych (pierwiastkowanie za pomocą funkcji *sqrt*), blok operacji zapisu i odczytu pamięci (kopiowanie pamięci funkcją *memcpy*), blok operacji na rejestrach procesora (operacje przesunięć i inkrementacji). Test wykonywany jest w pętli *4444* razy (liczba wybrana arbitralnie) i trwa kilka sekund. Otrzymany współczynnik jest proporcjonalny do czasu wykonania testu, przy czym im mniejsza moc maszyny, tym większa wartość współczynnika mocy. Dla maszyn, na których uruchamiano system (dwa typy maszyn), otrzymano następujące współczynniki: f_p(386DX40) = 1,88, f_p(486SX33) = 1,01. Wartości te dają dość dobre przybliżenie mocy maszyn, ponieważ w stanie jałowym obciążone są w podobnej proporcji (ok. 2:1). W takiej też proporcji zwiększa się obciążenie tych maszyn przy uruchamianiu różnych aplikacji. Test uruchamiany jest przy starcie SA, a wartość f_p dla danej maszyny przesyłana jest do SRO wraz z komunikatem START.

Licznik uśredniony (oznaczany jako v^a) jest to wartość średnia licznika znormalizowanego z ostatnich dziesięciu pomiarów (tzw. ruchoma średnia). Wartość 10 została wybrana arbitralnie.

Licznik zmodyfikowany jest bardziej złożony od licznika uśrednionego, choć jego działanie jest podobne. Jest on wyznaczany na podstawie wartości poprzedniej i aktualnej licznika znormalizowanego (lub surowego, jeśli zdecydujemy się nie brać pod uwagę współczynnika f_p), tj.

$$v_0^m = v_0^n$$

$$v_i^m = \frac{w\, v_{i-1}^m + v_i^n}{w+1}$$

Indeks i jest kolejnym numerem taktu pomiarowego, v^n jest licznikiem znormalizowanym, w jest pewną wagą. Współczynnik w nazywamy *wagą historii* lub w skrócie *historią*. Cała przeszłość danego licznika kumuluje się w liczniku modyfikowanym, z tym że im starszy pomiar, tym mniej on znaczy w ogólnej wartości licznika - inaczej niż dla licznika uśrednionego, gdzie były brane pod uwagę tylko wartości z przedziału uśredniania i wszystkie z taką samą wagą.

Moduł zbierania danych i statystyki zapamiętuje dane o obciążeniu całego monitorowanego systemu w postaci surowych wartości wybranych liczników oraz w wyliczaniu na bieżąco kilku prostych statystyk, które pozwalają ocenić zgrubnie obciążenie i zrównoważenie systemu. Statystyki te to: średnia wartość znormalizowanego czasu procesora, wariancja znormalizowanego czasu procesora, średnia odchyłka kwadratowa znormalizowanego czasu procesora. Statystyki liczone są dla wszystkich maszyn pracujących w systemie. Ich duża zmienność świadczy o nierównomiernym obciążeniu systemu. Wartości te są wyświetlane przez SRO, co daje operatorowi możliwość bieżącej oceny pracy systemu i ewentualnych zmian parametrów SRO. Zapamiętane szczegółowe dane o obciążeniu umożliwiają dokładną analizę pracy każdej maszyny z osobna, jak i systemu jako całości.

Moduł decyzyjny jest sercem SRO. Zawarte są w nim strategie przydziału zadań dla maszyn pracujących w systemie. Jakość pracy systemu równoważenia zależy od strategii podejmowania decyzji oraz parametrów SRO.

3.2.1. Parametry serwera równoważenia obciążeń

Zdefiniowano następujące parametry SRO:

- j_{max}, maksymalna ilość zadań, które mogą wykonywać się na jednej maszynie.
- l_{upper}, górne ograniczenie obciążenia (próg górny), powyżej którego nie należy obciążać maszyny, aby nie zablokować jej pracy. Parametr ten dotyczy wybranego licznika i dla każdego licznika ma specyficzną wartość.
- l_{lower}, dolne ograniczenie obciążenia (próg dolny), poniżej którego, niezależnie od obowiązującej w danej chwili strategii, zadanie należy wykonać lokalnie. Parametr ten dotyczy wybranego licznika i dla każdego licznika ma specyficzną wartość.

- M_{max}, maksymalna liczba migracji zadania w systemie, tj. dopuszczalna liczba realokacji zadania pomiędzy komputerami. Ograniczenie to zapobiega nadmiernej migracji zadań i „krążeniu zadań bez końca" pomiędzy komputerami.
- Δt, takt pomiarowy, tj. czas pomiędzy poszczególnymi pomiarami obciążenia maszyny (interwał).
- Rodzaj licznika wybrany jako podstawa do równoważenia obciążeń, identyfikowany przez swój indeks (tzw. *Title Index*).
- Tryb licznika, czyli sposób modyfikacji licznika (licznik surowy, znormalizowany, uśredniony, zmodyfikowany), jak opisano wcześniej.
- w, waga historii, współczynnik specyficzny dla licznika zmodyfikowanego, opisany wcześniej.

3.2.2. Strategie alokacji zadań SRO

Decyzja, gdzie wykonane ma zostać zadanie, podejmowana jest wg jednej ze strategii alokacji zadań zaimplementowanych w SRO. Operator ma możliwość dynamicznej zmiany strategii, co pozwala na przystosowanie systemu do nowych warunków bez przerywania pracy. Strategie podzielić można na dwie grupy: statyczne i dynamiczne. W strategiach statycznych nie jest brana pod uwagę aktualna sytuacja sieci, czyli stan obciążenia maszyn. Strategie statyczne to:

- *Wybrana maszyna* (*ang. Fixed Host*). Wszystkie zadania kierowane są na jedną, wybraną maszynę. Jeśli na maszynie tej nie zainstalowano aplikacji użytkowej, zadanie nie może być wykonane (SRO wyśle do SA komunikat *NO HOST*). Strategia ma zastosowanie w sytuacjach awaryjnych i przy uruchamianiu systemu.
- *Zawsze lokalnie* (*ang. Always Local*). Zawsze wykonuj zadanie lokalnie, jeśli nie można wykonać lokalnie, nie wykonuj. SRO w tym ostatnim przypadku na pytanie SA o to, gdzie wykonać zadanie, odpowie, że nie ma maszyny mogącej wykonać zadanie (SRO wyśle do SA komunikat *NO HOST*). Zastosowanie jw.
- *Cyklicznie do wszystkich maszyn* (*ang. Cycle All Hosts*). Zadania wysyłane są cyklicznie do maszyn, na których zainstalowana jest aplikacja użytkowa. Strategia ta wydaje się korzystna dla zadań jednorodnych, jednakowo obciążających maszynę.
- *Losowo* (*ang. Randomize Hosts*). Maszyna, na której ma zostać wykonane zadanie losowana jest z rozkładem równomiernym spośród wszystkich maszyn, na których zainstalowana jest aplikacja użytkowa.

Strategie dynamiczne to:

- *Wartość licznika* (*ang. Use Counter*). Zadanie zostanie przesłane do maszyny, której obciążenie, określane wg wybranego licznika w wybranym trybie, jest najmniejsze.

- *Liczba zadań* (*ang. Number of Jobs*). Zadanie zostanie przesłane do maszyny, na której w danej chwili wykonuje się najmniej zadań. Przy wyborze maszyny, może, ale nie musi, być wykorzystywany współczynnik normalizacyjny f_p (współczynnik mocy maszyny). Jeśli f_p będzie brany pod uwagę, wybrana zostanie maszyna, dla której najmniejszą wartość ma iloczyn $n_i \cdot f_p$, gdzie n_i jest ilością zadań wykonujących się aktualnie na maszynie i.

3.2.3. Algorytm działania modułu decyzyjnego SRO

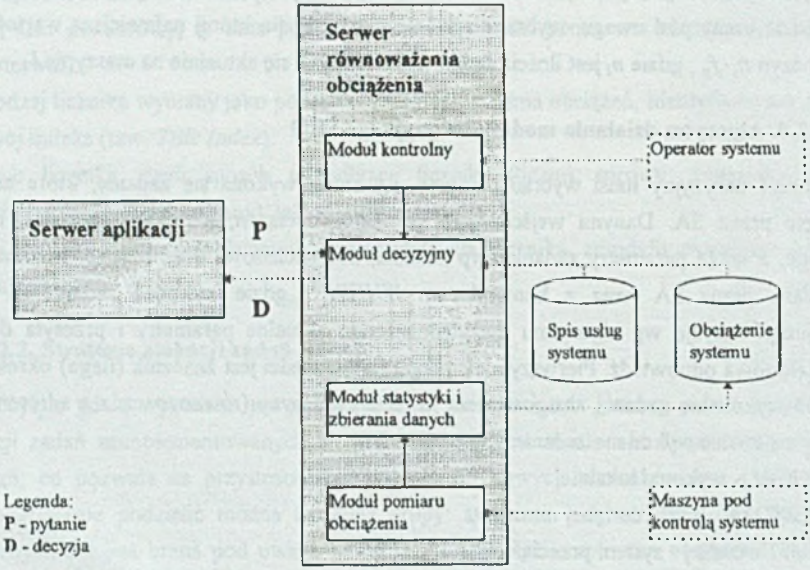
Moduł decyzyjny musi wybrać maszynę, na której wykona się zadanie, które zostało przyjęte przez SA. Danymi wejściowymi są tablice obciążeń, parametry systemu, rodzaj strategii, a także parametry zadanie (typ zadania, dotychczasowa ilość migracji w systemie) przysłane przez SA wraz z komunikatem *WHERE* („gdzie wykonać zadanie?"). SRO podejmuje decyzję wg algorytmu uwzględniającego aktualne parametry i przesyła do SA dwuczłonową odpowiedź. Pierwszym elementem odpowiedzi jest znacznik (flaga) określający sposób wykonania zadania, drugi element to nazwa maszyny (równoznaczna z adresem), na której ma zostać wykonane zadanie. Znaczniki te to:

- *LOCAL* - wykonaj lokalnie,
- *REMOTE* - wyślij do innej maszyny,
- *WAIT* - czekaj - system przeciążony,
- *NO HOST* - nie można wykonać.

Zadanie zaczyna się wykonywać bezpośrednio po tym, jak SA otrzymuje od SRO odpowiedź *LOCAL*. W innych przypadkach zadanie zostanie przesłane dalej (*REMOTE*), będzie oczekiwało w kolejce (*WAIT*) lub próba jego wykonania zakończy się niepowodzeniem (*NO HOST*).

Algorytm jest trójfazowy. W pierwszej fazie podejmowania decyzji rozpatrywane są parametry nadrzędne, takie jak maksymalna liczba migracji zadania M_{max} (rozpatrywana zawsze) i próg dolny l_{lower} (rozpatrywany w zależności od aktualnej konfiguracji SRO). W tej fazie może zostać podjęta decyzja o lokalnym wykonaniu zadania (*LOCAL*). Jeśli tak się nie stanie, algorytm przechodzi do fazy drugiej i wybiera maszynę wg aktualnej strategii. Może się okazać, że nie ma maszyny z wymaganą aplikacją użytkową - w takim przypadku zadania nie można wykonać (*NO HOST*). Jeśli zadanie można wykonać, wybrana zostaje najlepsza (dla danej strategii) maszyna i algorytm przechodzi do fazy trzeciej. Teraz interpretowane są parametry: próg górny l_{upper} (rozpatrywany w zależności od aktualnej konfiguracji SRO) i maksymalna ilość zadań j_{max} (rozpatrywana zawsze). W wyniku przekroczenia któregoś z tych parametrów może zostać podjęta decyzja, że w tej chwili nie można wykonać danego zadania, ale prawdopodobnie będzie je można wykonać później (*WAIT*). Jeśli zadanie można w danej chwili wykonać, SRO sprawdza, czy wybrana maszyna jest dla zadania lokalna czy

zdalna i wysyła do SA odpowiedni komunikat (*LOCAL* lub *REMOTE*), zlecając wykonanie zadania lub przesłanie go na inną maszynę.



Rys. 4. Schemat działania SRO
Fig. 4. The operation of the system

4. Aplikacje użytkowe w systemie równoważenia obciążeń

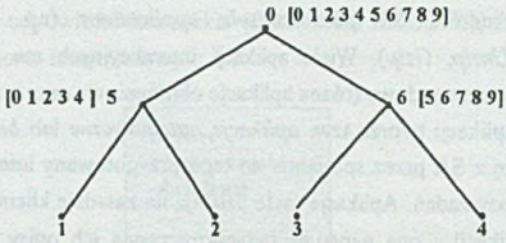
Aplikacje użytkowe, które mogą pracować pod kontrolą systemu, można podzielić na dwie grupy. Do pierwszej grupy należą aplikacje wsadowe. Mogą to być zarówno aplikacje Windows NT (GUI lub console), jak też aplikacje innych systemów, które mogą być uruchamiane w środowisku Windows NT (np. DOS). Aplikacje takie można uruchamiać poprzez serwer aplikacji, wpisując po prostu ich parametry w odpowiednim okienku dialogowym. System wybiera komputer, na którym zadanie ma zostać wykonane i uruchamia tam aplikację wsadową z odpowiednimi parametrami. Inną metodą jest uruchamianie całego wsadu zadań przez opisane wcześniej skrypty, które są interpretowane przez SA. Zadania opisane w takim skrypcie traktowane są jako niezależne i uruchamiane przez system w środowisku sieciowym. Aplikacje wsadowe nie są zbyt popularne ani też często spotykane w środowisku graficznym Windows, niemniej jednak istnieje wiele popularnych aplikacji ze środowisk tekstowych (DOS, UNIX) zarówno w wersjach oryginalnych, jak i przeniesionych

do środowiska Windows NT jako *console applications* (np. popularne programy kompresujące *Zip/Unzip, Gzip*). Wiele aplikacji interakcyjnych ma moduły wykonawcze działające jako aplikacje wsadowe (różne aplikacje obliczeniowe oraz graficzne).

Drugą grupę aplikacji tworzą tzw. *aplikacje automatyczne* lub *bezpośrednie*. Aplikacje takie komunikują się z SA przez specjalnie do tego przygotowany interfejs programowy i tą drogą przesyłają opisy zadań. Aplikacje takie działają na zasadzie klient/serwer. Klientem jest część sterująca aplikacji - ona generuje zadania, przesyła ich opisy do SA i ewentualnie odbiera i interpretuje wyniki (np. jeśli zadania są zależne). Serwerem jest zwykła aplikacja wsadowa, która wykonuje zadania. Połączenie pomiędzy serwerem a klientem zapewnia nasz system. Aplikacje te są tworzone z wykorzystaniem dołączonej do systemu biblioteki *API*. Do tej klasy aplikacji należy system rozpoznawania wieloetapowego [6], który wdrożono w postaci 32-bitowej okienkowej aplikacji Windows NT, pracującej pod kontrolą metakomputera zarządzanego naszym SRO.

4.1. Rozpoznawanie wieloetapowe w sieci

Rozpoznawanie wieloetapowe należy do technik rozpoznawania, w których decyzja o przynależności obiektu do określonej klasy nie jest czynnością jednorazową, ale jest wynikiem złożonego procesu decyzyjnego [6]. Nie wnikając w szczegóły algorytmizacji tego sposobu rozpoznawania, z punktu widzenia systemu obliczeniowego, wykonanie pojedynczego rozpoznania wieloetapowego polega na wykonaniu pracy będącej wielozadaniowym łańcuchem zadań. Poszczególne zadania są związane z wykonaniem lokalnego algorytmu decyzyjnego, których sekwencja decyduje o wyniku rozpoznania wieloetapowego. Reprezentacją schematu decyzyjnego rozpoznawania wieloetapowego jest drzewo decyzyjne z węzłami wewnętrznymi typu OR, reprezentującymi lokalne algorytmy decyzyjne. Węzłom terminalnym drzewa przypisane są klasy obiektów. Wieloetapowy proces decyzyjny rozpoczyna się zawsze od decyzji lokalnej w korzeniu drzewa. Następne decyzje lokalne, a więc także odpowiadające im zadania obliczeniowe, wybierane są na podstawie bieżącej decyzji lokalnej, która określa pewien podzbiór w zbiorze klas i jednocześnie wskazuje cechy obiektu, jakie należy zmierzyć, aby podjąć decyzję na etapie następnym. Oczywiście wskazany podzbiór klas jest osiągalny z danego węzła drzewa. Rysunek 5 przedstawia drzewo decyzyjne wykorzystywane przez wdrożony system rozpoznawania wieloetapowego.



Rys. 5. Przykładowe drzewo decyzyjne (w nawiasach kwadratowych podano numery cech przetwarzanych w węźle) wykorzystane w badaniach

Fig. 5. Sample decision tree with nodes labelled by features used at the nodes

4.1.1. Implementacja aplikacji rozpoznawania

Aplikacja rozpoznawania składa się z *managera rozpoznawania* i *serwera rozpoznawania*. Serwer rozpoznawania jest aplikacją wsadową realizującą rozpoznawanie w poszczególnym węźle drzewa decyzyjnego - zawiera ona lokalny algorytm decyzyjny, wspólny dla wszystkich węzłów. Rozpoznanie jednego obiektu to ciąg wywołań tego programu dla odpowiednich węzłów decyzyjnych na ścieżce prowadzącej od korzenia do węzła reprezentującego klasę obiektu.

Program manager rozpoznawania zarządza bazą danych zawierającą obiekty do rozpoznania, strukturę drzewa decyzyjnego oraz ciągi uczące dla algorytmu rozpoznawania z uczeniem, którym był algorytm najbliższego sąsiada. Manager znajduje się na jednym dedykowanym hoście. Przekazuje on serwerowi aplikacji zadanie (pliki zawierające obiekt, ciągi uczące i opis węzła drzewa decyzyjnego, w którym należy wykonać etap rozpoznawania). Serwer aplikacji pobiera te odpowiednie pliki i w postaci zadania uogólnionego zapisuje w swojej kolejce zadań wejściowych. Zadanie przekazywane jest do serwera rozpoznawania, a ten, na podstawie plików otrzymanych po rozpakowaniu zadania uogólnionego, wykonuje rozpoznawanie w danym węźle drzewa decyzyjnego. Wynikiem rozpoznawania jest numer węzła następnika i decyzja (koniec rozpoznawania - osiągnięto węzeł terminalny lub rozpoznajemy dalej). Wyniki są zwracane przez SA do managera rozpoznawania.

4.2. Eksperymenty i badania systemu

Badania systemu polegały na przeprowadzeniu szeregu eksperymentów mających określić wpływ parametrów systemu, a w szczególności parametrów SRO, na czas wykonania wsadu zadań rozpoznawania wieloetapowego oraz na jakość zrównoważenia sieci Windows NT.

Wykorzystano sieć złożoną z sześciu maszyn Windows NT (trzy komputery 386DX40 i trzy komputery 486SX33). SA zainstalowane były na wszystkich maszynach, SRO na maszynie I17NT07 (486SX33). Wszystkie maszyny dysponowały pamięcią operacyjną 16 MB.

Zadania były generowane przez manager rozpoznawania. Zadania przyjmował SA na maszynie I17NT09. Serwery rozpoznawania zainstalowane były na wszystkich maszynach, tak by cała sieć była wykorzystana. SA komunikują się z podejmującym decyzje SRO oraz bezpośrednio każdy z każdym podczas przesyłania zadań i wyników.

Przeprowadzono ok. 100 eksperymentów dla różnych zestawów wartości parametrów systemu. Jednorazowy wsad prac liczył 300 prac - aplikacja rozpoznawania miała rozpoznać 300 obiektów przy drzewie decyzyjnym o dwu poziomach nieterminalnych. Powodowało to wygenerowanie 600 zadań za każdym razem. Czas wykonania całego wsadu na jednym komputerze wynosił ok. 18 minut, a jego przetworzenie wymagało przesłania w sieci ponad 9 MB danych. Dla oceny obciążenia brano pod uwagę wszystkie zdefiniowane w systemie liczniki. Ze względu na brak miejsca w niniejszym artykule omówimy jedynie wyniki dotyczące licznika *Processor Time* (tj. czas procesora). Pełną prezentację wyników zawierają prace [1, 5].

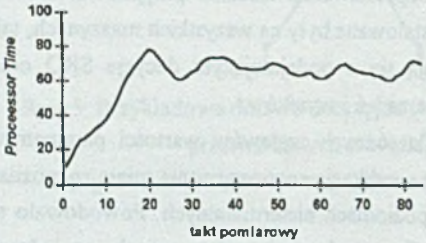
System oceniano badając:

- czas wykonania wsadu T_w,
- chwilowe średnie obciążenie liczone jako średnia arytmetyczna obciążenia po wszystkich maszynach w danym takcie pomiarowym,
- chwilowe średnie zrównoważenie liczone jako średnia odchyłka standardowa obciążenia po wszystkich maszynach w danym takcie pomiarowym,
- obciążenie O, liczone jako średnia arytmetyczna z chwilowego średniego obciążenia po czasie trwania eksperymentu,
- zrównoważenie Z, liczone jako średnia arytmetyczna z chwilowego średniego zrównoważenia po czasie trwania eksperymentu (większa wartość oznacza gorsze zrównoważenie).

Z porównania badanych strategii alokacji zadań wynika, że strategie dynamiczne dają lepsze wyniki co do czasu wykonania niż strategie statyczne. Najmniejsze wartości czasu T_w uzyskano dla strategii dynamicznej *wartość licznika* z licznikiem *Processor Time*. Czasy te przykładowo wynosiły dla strategii: *Cyklicznie*, *Losowo*, *Liczba zadań*, *Licznik Processor Time* wynosiły odpowiednio: 533, 507, 484 i 437 sekund.

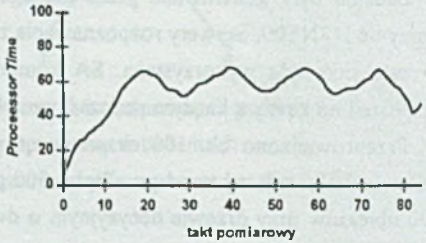
Przebadano skalowalność systemu, tj. możliwość uzyskiwania zmniejszenia czasu wykonania wsadu prac wraz ze wzostem liczby komputerów w sieci. Badano wykonanie wsadu prac w systemie przy zwiększającej się liczbie maszyn od 1 do 6. Następowało zmniejszenie czasu wykonania wsadu zadań po dołączeniu każdej nowej maszyny. Czas

wykonania wsadu zadań był krótszy od czasu wykonania na jednej maszynie od 1,8 do 2,8 razy. Zakładając, że system przy zwiększającej się liczbie maszyn będzie zachowywć się podobnie jak w zakresie maszyn 1-6, możemy skalowalność systemu oceniać na kilkanaście maszyn.



Rys. 6. Średnie chwilowe obciążenie (eksperyment 1.13) T_w=443, O=61.2, Z=34.8

Fig. 6. Average load (experiment 1.13) T_w=443, O=61.2, Z=34.8



Rys. 7. Średnie chwilowe obciążenie (eksperyment 1.17) T_w=476, O=54.8, Z=38.4

Fig. 7. Average load (eksperiment 1.17) T_w=476, O=54.8, Z=38.4

Rysunki 6 i 7 przedstawiają przebiegi średniego chwilowego obciążenia systemu w dwóch przykładowych eksperymentach (1.13) i (1.17). Obciążenie rośnie w czasie napływu zadań do systemu i później oscyluje wokół wartości ok. 70% w czasie przetwarzania wsadu zadań. Eksperymenty różnią się tylko parametrami systemu, które wpływają na czułość systemu na chwilowe zmiany obciążenia. Czułość systemu jest mniejsza w eksperymencie (1.17). Jak widać, wpłynęło to negatywnie na jakość przetwarzania wsadu. Czas T_w dla eksperymentu (1.17) był wyższy niż (1.13), obciążenie O było również mniejsze - nie wykorzystano w pełni mocy obliczeniowej systemu; zrównoważenie Z miało wartość większą, a więc gorszą. Jak widać, przebieg chwilowego obciążenia w eksperymencie (1.13) jest bardziej ustabilizowany niż dla (1.17), który jest silnie zmienny, co potwierdza związek zrównoważenia i czasu wykonania wsadu. Widać tu też ciekawą właściwość sieci Windows NT jako maszyny wirtualnej: obciążenie w skali sieci zmienia się w odpowiedzi na wsad zadań w sposób podobny jak obciążenie jednej maszyny w odpowiedzi na uruchamianie kolejnych wątków aplikacji wielowątkowej. Obserwowane były przebiegi o charakterze drgań gasnących podobne do tego, jaki przedstawia rys.6 dla eksperymentów, gdzie osiągnięto lepszą jakość zrównoważenia oraz przebiegi nieregularne, o charakterze zmian okresowych, podobne do przebiegu z rys. 7 - dla eksperymentów, gdzie osiągnięto gorsze zrównoważenie.

Analizując wpływ parametrów systemu na jakość jego działania można sformułować następujące ważne wnioski:

- Dla małych wartości taktu pomiarowego Δt system daje gorsze wyniki co do czasu wykonania wsadu i warunków zrównoważenia. Spowodowane jest to znacznym obciążeniem

generowanym przez *Performance Library Windows NT* w trakcie wykonywania pomiarów. Pomiary w takiej sytuacji tracą wiarygodność.

- Duża dopuszczalna liczba migracji (ok. 5) pogarsza działanie systemu. Dla dużej liczby migracji koszty działania systemu przekraczają zyski, jakie daje rozpraszanie i równoważenie obciążeń.
- Zrównoważenie było gorsze, jeśli system był mniej czuły na zmiany obciążenia. Można to wytłumaczyć tym, że reakcja na zmianę obciążenia była wolniejsza.
- Progi nie polepszają ani zrównoważenia systemu, ani czasu wykonania wsadu T_w. Zastosowanie progów wydłuża czas wykonywania wsadu. Jedynym skutkiem działania progu górnego było zmniejszenie średniego obciążenia systemu, ale kosztem czasu T_w, co może być niekiedy celowe.

5. Podsumowanie

Idea metaprzetwarzania, czyli wirtualnego środowiska obliczeniowego (metakomputera) jest bardzo atrakcyjną propozycją realizacji obliczeń w sieci komputerowej. Prezentowany w niniejszym artykule system równoważenia obciążenia SRO realizuje wskazaną ideę w środowisku komputerów Windows NT. Wykorzystanie podobnego systemu komunikacji zarówno w uprzednio opracowanym systemie równoważenia obciążenia w sieci komputerów UNIX [3, 4], jak i w prezentowanym tutaj systemie dla komputerów Windows NT umożliwi szybką realizację systemu podziału obciążenia pomiędzy różnymi systemami operacyjnymi w sieci komputerowej.

LITERATURA

[1] Borzemski L., Kieda A.: Metaprzetwarzanie w środowisku Windows NT dla potrzeb rozpoznawania wieloetapowego, (praca w przygotowaniu).

[2] Borzemski L., Błażków R., Wronka J.: Wieloetapowe podejmowanie decyzji w sieci komputerowej z równoważeniem obciążenia. Zeszyty Naukowe Politechniki Śląskiej, s. Informatyka z. 24, Gliwice 1993.

[3] Borzemski L., Gajewski D., Głowacz S., Szczypiński M., Wojtczak K.: A Load Balancing System for Unix-Based Local Area Networks, Microprocessing and Microprogramming, vol. 39, 1993.

[4] Borzemski L., Głowacz S., Wojtczak K.: System równoważenia obciążeń w sieci komputerów UNIX, Zeszyty Naukowe Politechniki Śląskiej, s. Informatyka z. 24, Gliwice 1993.

[5] Kieda A., Niekrasz K., Świstek J., Iluk A.: System równoważenia obciążeń w sieci Windows NT, Praca dyplomowa, Instytut Sterowania i Techniki Systemów PWr., Wrocław, 1995.

[6] Kurzyński M.: Algorytmy rozpoznawania wieloetapowego oraz ich zastosowania medyczne i techniczne, Prace Naukowe Instytutu Sterowania i Techniki Systemów PWr., Wrocław 1987.

[7] Microsoft Technet CD, May 1994, Volume 2, Issue 5, Microsoft, 1994.

[8] Richter J.: Advanced Windows NT: Developer's Guide to Win32 Application Programming Interface, Microsoft, 1994.

[9] Stroiński M., Węglarz J.: Metaprzetwarzanie - stan aktualny i perspektywy. Materiały II Krajowej Konferencji KOWBAN '95, Wrocław 1995.

Recenzent: Doc. dr hab. inż. Adam Mrózek

Wpłynęło do Redakcji 21 grudnia 1995 r.

Abstract

This paper presents a distributed load balancing computer system developed for Windows NT based local area network. The load balancing features were added to the Windows NT operating system without any modification of its kernel. The load balancing algorithms can distribute computational tasks to processors such that the processor utilization and the throughput of the system are maximized. The tasks can arrive to each host. The system has a central allocation module that is running in one of the host, whereas other hosts together with the central one can run as the servers. The performance library for Windows NT is used as a means for obtaining performance data about hosts.