ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 2000
Seria: INFORMATYKA z. 38 Nr kol. 1451

Agnieszka DEBUDAJ-GRABYSZ
Politechnika Śląska, Instytut Informatyki

O PEWNEJ ORGANIZACJI PRZETWARZANIA WSADOWEGO Z WYKORZYSTANIEM POCZTY ELEKTRONICZNEJ

Streszczenie. Praca dotyczy tematyki związanej ze współbieżnym realizowaniem obliczeń w systemach rozproszonych. Zawiera charakterystykę systemów pozwalających na współbieżne wykonywanie aplikacji w klastrze wraz z przykładami systemów obecnie najpopularniejszych. Przedstawiona jest także propozycja programu pozwalającego przyjmować żądania wykonania prac za pomocą poczty elektronicznej.

ON A CERTAIN ORGANIZATION OF THE BATCH PROCESSING APPLYING THE ELECTRONIC MAIL

Summary. The paper concerns the realisation of the concurrent computation in distributed systems. Cluster management systems are characterized and the most known ones are reviewed. A proposal of the program to receive job execution demands via the electronic mail is also presented.

1. Wstęp

Badania na temat współbieżnego wykonywania obliczeń są coraz bardziej popularne. Szacuje się, że gwałtowny wzrost szybkości obliczeniowej procesora związany z postępem technologii jego wykonania osiągnie wkrótce swój kres [8]. Stąd też droga do uzyskania większych szybkości przetwarzania prowadzi przez współbieżność.

Jednym ze sposobów na jej osiągnięcie jest wykorzystanie sieci komputerowej jako zbioru połączonych ze sobą procesorów i odpowiednie zarządzanie dostępnym w sieci oprogramowaniem. Niniejszy artykuł dotyczy zagadnień związanych ze współbieżnym realizowaniem obliczeń w systemach rozproszonych.

Artykuł rozpoczyna się od zdefiniowania zestawu podstawowych pojęć stanowiących wprowadzenie w omawianą tematykę. Następnie zaprezentowane i omówione są kryteria, według których ocenia się tak badawcze, jak i komercyjne narzędzia umożliwiające przetwarzanie rozproszone. Krótka charakterystyka kilku wybranych narzędzi stanowi uzupełnienie informacji.

W dalszej części artykułu przedstawiony jest program nazwany AUTOMAT, pozwalający realizować jeden z elementów systemu umożliwiającego prowadzenie obliczeń rozproszonych – przetwarzanie wsadowe. Oryginalnym rozwiązaniem zastosowanym w AUTOMACIE jest kierowanie żądań do systemu za pomocą poczty elektronicznej.

Omówiony jest także przykład zastosowania AUTOMATU ze szczególnym uwzględnieniem możliwości wznowienia i kontynuowania obliczeń w przypadku ich przerwania.

1.1. Słownik pojęć podstawowych

Problem (ang. *problem*) – praktyczne zagadnienie wymagające rozwiązania. Jedną, a czasem jedyną metodą rozwiązania problemu jest posłużenie się komputerem.

Algorytm (ang. *algorithm*) – jednoznaczny, ścisły i kompletny zapis metody rozwiązania problemu.

Program (ang. *program*) – zapis algorytmu w postaci zrozumiałej przez komputer.

Proces (ang. *process*) – działanie komputera związane z wykonywaniem programu, skojarzone w danej chwili z konkretnym procesorem i wydzieloną częścią pamięci. Wykonanie programu może powołać do życia jeden lub więcej procesów.

Praca (ang. *job*) – wykonanie programu służące rozwiązaniu części problemu. Przykładem może być sytuacja, w której problemem do rozwiązania jest przeprowadzenie analizy wytrzymałościowej poszczególnych elementów projektowanej maszyny. Analiza ta jest przeprowadzana tym samym programem dla różnych zestawów danych wejściowych. Wykonanie programu dla pojedynczego zestawu danych jest pracą.

Zadanie (ang. *task*) – część programu, która przy zapewnieniu komunikacji i synchronizacji z resztą programu może być wykonywana jako oddzielny proces. Zadaniami mogą być podprogramy lub iteracje pętli.

Obliczenia współbieżne (ang. *concurrent computing*) – wykonywanie programu, w którym pewne prace, zadania lub mniejsze jednostki programu realizowane są przez odrębne procesory (lub moduły procesora) działające w tym samym czasie. Synonim: obliczenia równoległe (ang. *parallel computing*).

1.2. Klasyfikacja obliczeń współbieżnych

Biorąc za podstawę klasyfikacji poziom, na jakim realizowana jest współbieżność (porównaj [1]), można wyszczególnić:

1) obliczenia współbieżne na poziomie instrukcji i części instrukcji:
   Jest to poziom najniższy. Do jego realizacji potrzebny jest specjalny procesor wyposażony w kilka autonomicznych jednostek przetwarzających mogących operować na oddzielnych danych w tym samym czasie. Sposób ten sprowadza się do przetwarzania przez procesory wektorowe i macierzowe oraz do przetwarzania potokowego (ang. *pipelining*), stąd do tej klasy zaliczamy przypadek, w którym kompilator zapewnia automatyczną wektoryzację albo rozwijanie pętli (ang. *unrolling*). Poziom ten — z wyjątkiem ustawień opcji kompilatora — jest niedostępny dla programisty;
2) obliczenia współbieżne na poziomie zadań:
   Do realizacji obliczeń współbieżnych na poziomie zadań potrzebny jest więcej niż jeden procesor. Procesory mogą należeć do jednego lub kilku komputerów. Współbieżność na tym poziomie realizuje się przez zmianę kodu programu. Wykorzystuje się albo mechanizmy niskopoziomowe (semafory, pamięć dzielona itp.), bądź odpowiednie narzędzia programowania równoległego, np. HPF, MPI, PVM;
3) obliczenia współbieżne na poziomie prac:
   Do jego realizacji także wymagana jest obecność wielu procesorów należących do jednego lub kilku komputerów. Kod programu pozostaje niezmieniony, wymaga się natomiast odpowiedniego zarządzania pracami.

2. Systemy zarządzania pracami

Środowisko obliczeniowe utworzone z wielu komputerów (stacji roboczych) połączonych siecią tworzy *system rozproszony* (ang. *distributed system*). Jego koncepcja jest identyczna z architekturą komputera MIMD (ang. *multiple instructions, multiple data*), pracującego z przesyłaniem komunikatów (ang. *message passing*). Część takiego systemu przeznaczona do określonych zastosowań i wyposażona w odpowiednie narzędzia programowe nazywana jest *klastrem*. Oprogramowanie zarządzające klastrem określa się w literaturze mianem *cluster management software*.

Podstawowym celem stosowania klastra jest efektywne wykorzystanie mocy obliczeniowej sieci stacji roboczych. U podstaw jego działania leży wykorzystanie wolnych cykli procesorów, ponieważ szacuje się, że stacje robocze wykorzystywane są jedynie w dziesięciu procentach [5].

Klaster może być wykorzystywany przez wielu użytkowników wykonujących programy nie związane z rozwiązywaniem tego samego problemu, a jego specjalizowanym zastosowaniem może być realizacja współbieżności na poziomie prac.

Użytkowanie klastra rozpoczyna się zwykle od utworzenia pliku z opisem pracy. Zawiera on między innymi informacje o aplikacji do uruchomienia, zestaw danych wejściowych, informacje o wymaganych zasobach (np. rodzaj CPU, ilość pamięci) itp. Plik opisu pracy nie jest przez użytkownika kierowany do konkretnej stacji, lecz do całego systemu. W systemie zaimplementowany jest *organizator* (ang. *scheduler*), czyli mechanizm, który dysponując informacjami o stanie wszystkich stacji w klastrze, kieruje pracę do kolejki w stacji docelowej, spełniającej wymagania podane w pliku opisu pracy. Zadaniem organizatora jest także wyrównywanie obciążenia klastra.

3. Ważniejsze cechy idealnego systemu zarządzania pracami

Wraz z rozwojem i upowszechnianiem systemów zarządzania pracami rosną stawiane im wymagania. Kilka z nich zasługuje na szczególną uwagę [2].

- **Przetwarzanie wsadowe** (ang. *batch processing*) — możliwość uruchomienia pracy na podstawie pliku (zwykle tekstowego) zawierającego zestaw słów kluczowych interpretowanych przez system. Przetwarzanie wsadowe eliminuje konieczność interakcyjnej pracy użytkownika z systemem.
- **Interakcyjne uruchamianie prac** (ang. *interactive support*) — możliwość dialogu systemu z użytkownikiem w celu uzgodnienia warunków rozpoczęcia pracy.
- **Wyrównywanie obciążenia** (ang. *load balancing*) — mechanizm pozwalający uzyskać równomierne obciążenie węzłów klastra. Obciążenie może być wyrównywane statycznie (analizuje się obciążenie jednorazowo, bezpośrednio przed uruchomieniem pracy) lub dynamicznie (praca może być w trakcie wykonywania przekierowana do innej stacji, jeżeli wystąpiło zachwianie równowagi obciążenia w sieci).
- **Zapisywanie punktów kontrolnych** (ang. *checkpointing*) – cykliczne zapamiętywanie stanu obliczeń wraz ze wszystkimi danymi istotnymi z punktu widzenia wykonywanego programu.

Mechanizm zapisywania punktów kontrolnych jest wykorzystywany przede wszystkim jako zabezpieczenie na wypadek awarii stacji, na której wykonywana jest praca. Zapamiętanie stanu obliczeń pozwala — po ponownym uruchomieniu pracy — na ich kontynuowanie od ostatniego zachowanego punktu. Innym zastosowaniem jest wspomaganie dynamicznego wyrównywania obciążenia.

Wyróżnia się następujące sposoby zapisywania punktów kontrolnych:

- w sytuacji najprostszej i najbardziej ogólnej aplikacja w trakcie działania tworzy pliki wynikowe, do których na bieżąco zapisuje stan wybranych zmiennych. Pliki te nie są w założeniu przeznaczone do realizacji omawianego mechanizmu, jednak informacje w nich zawarte pozwalają na odtworzenie stanu wybranych zmiennych i kontynuowanie obliczeń,
- na poziomie aplikacji — aplikacja zostaje napisana w taki sposób, że sama celowo tworzy pliki umożliwiające restart,
- na poziomie użytkownika — nie wymaga się zmiany kodu istniejącej aplikacji, lecz użytkownik dokonuje jego skonsolidowania z odpowiednimi bibliotekami. Mechanizm zapisywania punktów kontrolnych jest tu implementowany jako procedura obsługi sygnału. System zarządzający pracami w określonym momencie czasu generuje sygnał do rozpatrywanego procesu, a dostarczona procedura obsługi tego właśnie sygnału zapamiętuje odpowiednie informacje,
- na poziomie jądra systemu — zapisywanie punktów kontrolnych realizowane jest przez system operacyjny i jest całkowicie przezroczyste dla użytkownika. Nie wymaga się zmian w kodzie programu.

Zapisywanie punktów kontrolnych na trzech ostatnich poziomach polega na cyklicznym zachowywaniu w pliku informacji o stanie biegnącego procesu. Informacje te obejmują zawartość rejestrów, stosu systemowego, segment danych, kody używanych bibliotek, stan otwartych plików, procedury obsługi przerwań, oczekujące sygnały itp. Po utworzeniu następnego pliku ze stanem procesu poprzedni plik może zostać usunięty.

Głównym kosztem zapisywania punktów kontrolnych jest pochłanianie dużych ilości pamięci dyskowej potrzebnej na przechowywanie plików z danymi o stanie wykonywanej pracy.

- **Migracja procesów** (ang. *process migration*) — przenoszenie prac w trakcie ich wykonywania z jednej stacji do innej. Migracja procesów jest używana między innymi do realizacji wyrównywania obciążenia (przenoszenie ze stacji bardziej obciążonej do mniej obciążonej), a także do zminimalizowania wpływu wykonywanej pracy (obciążenie CPU, zajmowanie pamięci, zajmowanie przestrzeni dyskowej itp.) na komfort użytkowania stacji przez jej właściciela. Warunkiem stosowania migracji jest możliwość zapisywania punktów kontrolnych. Jeśli używany sposób realizacji zapisywania stosuje przechowywanie zawartości rejestrów procesora, stosu itd., to migracja jest ograniczona do stacji o tych samych architekturach i systemach operacyjnych.
- **Uruchamianie prac równoległych** (ang. *parallel support*) — możliwość wykonywania prac będących pracami równoległymi, napisanymi przy użyciu takich narzędzi, jak np. MPI, HPF, PVM, Linda.

- **Zarządzanie kolejkami prac** – konfigurowanie wielu rodzajów kolejek, które obsługują prace o różnych złożonościach (czasowych, pamięciowych). Zarządzanie obejmuje przydzielanie zasobów, blokowanie, odblokowywanie, ustalanie właściciela itp.
- **Inne:** posiadanie GUI, definiowanie limitu prac dla poszczególnych użytkowników, definiowanie priorytetu prac dla poszczególnych użytkowników, praca w systemach homo- i heterogenicznych.

4. Wybrane systemy zarządzania pracami

4.1. CODINE [2], [4], [5]

Jest pakietem komercyjnym firmy GENIAS Software GmbH. Przeznaczony jest do użytkowania w heterogenicznym środowisku sieciowym w celu optymalnego wykorzystania zasobów sprzętowych i programowych. Jego główne cechy to:

- możliwość uruchamiania prac wsadowych, interakcyjnych, równoległych,
- możliwość zarządzania wieloma rodzajami kolejek prac,
- opcjonalna realizacja zapisywania punktów kontrolnych,
- wyrównywanie obciążenia statyczne i dynamiczne,
- posiadanie graficznego interfejsu użytkownika.

Standardowy organizator realizuje zasadę FIFO (praca, która pierwsza została zgłoszona do systemu, jako pierwsza jest rozpatrywana). W sieci wyszukiwany jest zestaw stacji spełniających wymagania co do uruchomienia pracy, po czym spośród nich wybierana jest stacja najmniej obciążona.

Istnieją podejścia alternatywne:

- pracom mogą być nadawane priorytety,
- może być zastosowana zasada równego podziału liczby prac o tym samym priorytecie, zgłoszonych przez poszczególnych użytkowników,
- mogą być zastosowane *ważone wskaźniki obciążenia*, czyli zgłaszane wartości obciążenia poszczególnych stacji mogą otrzymywać swoje wagi (np. koszt przetwarzania na maszynie A może być większy niż na maszynie B, dlatego mniejsze bezwzględne obciążenie maszyny A nie oznacza jednoznacznie, że ma ona zostać wybrana do przyjęcia pracy),
- można specyfikować obciążenia innych zasobów niż CPU, np. pamięci.

System CODINE realizuje zapisywanie punktów kontrolnych na poziomie użytkownika oraz na poziomie jądra systemu.

4.2. CONDOR [2], [3]

Jest pakietem badawczym, który został opracowany w University of Wisconsin, USA. Przeznaczony jest do uruchamiania prac wsadowych na nieobciążonych stacjach w heterogenicznym środowisku sieciowym. Użytkownik, który chce uruchomić aplikację w klastrze pod nadzorem systemu CONDOR, musi dysponować jej kodem źródłowym i dokonać konsolidacji z odpowiednimi bibliotekami.

Do systemu kierowane są zgłoszenia prac do uruchomienia oraz zgłoszenia gotowości ze strony wolnych stacji. System dokonuje kojarzenia prac i stacji.

System CONDOR nie jest wyposażony w graficzny interfejs użytkownika, nie umożliwia ani interakcyjnego uruchamiania prac, ani uruchamiania prac równoległych.

Cechą wyróżniającą pakiet jest zapisywanie punktów kontrolnych dla prac jednoprocesowych (wykonywanych przez jeden proces). System CONDOR dostarcza zestaw bibliotek dla różnych platform Unixowych, które skonsolidowane z aplikacją użytkownika pozwalają zrealizować zapisywanie punktów kontrolnych także w innym systemie zarządzania pracami (np. w CODINE).

4.3. GNQS (Generic Network Queuing System) [2], [10]

Jest pakietem badawczym (nie komercyjnym), rozprowadzanym przez University of Sheffield. Jego przeznaczeniem jest między innymi:

- uruchamianie prac wsadowych oraz realizacja zgłoszeń dostępu do urządzeń fizycznych (np. drukarki),
- zdalne kolejkowanie i marszrutowanie zgłoszeń w klastrze,
- zdalne informowanie o stanie odległych zgłoszeń.

GNQS umożliwia kilka sposobów rozwiązania problemu wyrównywania obciążenia:

1) całkowity brak wyrównywania. Zgłoszenia, które mogą być realizowane na kilku stacjach docelowych, są wszystkie przesyłane do jednej stacji z pominięciem innych;
2) stacje przeznaczone do przyjęcia zgłoszeń oczekujących w kolejce są wybierane metodą „round-robin" (w zbiorze dostępnych stacji, każda stacja kolejno, w sposób cykliczny, otrzymuje zgłoszenie do realizacji), aby bardziej równomiernie obciążyć sieć;
3) zgłoszenia czekają na stacji źródłowej aż do momentu, w którym stacja docelowa może natychmiast przyjąć pracę do uruchomienia;
4) w systemie zaimplementowany jest organizator. Organizator dysponuje informacjami o liczbie prac (zgłoszeń) wykonywanych na poszczególnych stacjach, wydajności stacji, a także o średnim obciążeniu w ciągu 1, 5, 15 minut. Organizator rozdziela zgłoszenia w systemie homogenicznym.

GNQS umożliwia też uruchamianie prac interakcyjnych. Nie realizuje zapisywania punktów kontrolnych, brak jest graficznego interfejsu użytkownika.

4.4. PBS (Portable Batch System) [2], [9]

Jest pakietem badawczym, opracowanym przez NASA Ames. Przeznaczony jest do szeroko rozumianego przetwarzania wsadowego w heterogenicznej sieci stacji roboczych. Umożliwia inicjowanie, śledzenie oraz rozpraszanie prac wsadowych. Mechanizm zajmujący się przydzielaniem prac do konkretnych stacji pozwala na wyspecyfikowanie, jakich zasobów i w jakich ilościach może używać dana praca. Mechanizm ten dysponuje informacjami o wykorzystywanych w danej chwili zasobach w sieci. Użytkownik ma możliwość wpływu na algorytm rozdzielający prace.

Pakiet PBS pozwala także na uruchamianie prac interakcyjnych i równoległych. Nie realizuje zapisywania punktów kontrolnych i nie posiada graficznego interfejsu użytkownika.

5. AUTOMAT

Program AUTOMAT opisany w niniejszym punkcie stanowi propozycję przetwarzania wsadowego z wykorzystaniem poczty elektronicznej.

5.1. Mechanizmy komunikacji między komputerami

Do komunikacji między odległymi komputerami (zdalnego zgłaszania się do systemu oraz zdalnego uruchamiania aplikacji) stosuje się między innymi następujące mechanizmy [7]:

- telnet — umożliwia zdalne zgłaszanie się do systemu. Wymaga interakcyjnego udziału użytkownika, więc nie może być wywołany przez program;
- remote shell (rsh, remsh) — umożliwia zdalne wykonywanie poleceń. Wymaga zmian w pliku `.rhosts` z katalogu użytkownika. Ponieważ plik `.rhosts` specyfikuje adresy, z których dany użytkownik może dostać się bez hasła, jest to element osłabiający bezpieczeństwo pracy w sieci;
- gniazdka (ang. *sockets*) — interfejs programu użytkowego dla warstwy protokołu komunikacyjnego sieci. Na komputerze, na którym ma być uruchomiona aplikacja, musi cały czas działać program oczekujący na zgłoszenie;
- secure shell (ssh) — zalecany obecnie ze względów bezpieczeństwa zamiast `telnet` i `remsh`. Jest jednak produktem komercyjnym, a co za tym idzie — płatnym.

Wydaje się, że w niektórych sytuacjach rozsądną alternatywą jest wykorzystanie poczty elektronicznej do komunikacji między komputerami.

5.2. Idea AUTOMATU

Proponowane rozwiązanie nie aspiruje do miana systemu zarządzania pracami, realizuje bowiem jego fragment, tzn. możliwość przetwarzania wsadowego. Ideą jego realizacji jest wykorzystanie poczty elektronicznej do zdalnego uruchamiania prac. Podejście to ma wiele zalet:

- istnieje powszechnie zaakceptowany standard dotyczący poczty, który jest niezależny od platformy,
- usługa ta jest zbadana pod względem bezpieczeństwa w sieci i chociaż nie jest w pełni bezpieczna, to mniej naraża system na włamanie, niż udostępnienie `remote shell` przez ustawienia w pliku `.rhosts`,
- obsługa poczty jest prosta,
- użytkownik nie musi posiadać własnego konta w systemie zdalnym.

Celem postawionym przez autora było opracowanie programu-usługodawcy (zwanego też serwerem), który po otrzymaniu za pomocą poczty opisu pracy do wykonania, automatycznie uruchomi odpowiedni program, a po jego wykonaniu prześle zwrotnie wyniki do usługobiorcy (zwanego też klientem).

Stworzony system działa przy następujących założeniach:

- czas potrzebny do wykonania zadania jest znacznie dłuższy niż czas potrzebny na transmisję,
- przesłane wraz z opisem pracy dane wejściowe są wystarczające do jego uruchomienia i wykonania, i nie jest wymagana dodatkowa wymiana informacji między usługodawcą i usługobiorcą.

AUTOMAT jest przeznaczony do uruchamiania w systemie UNIX. Jego główna część została napisana w języku powłoki Bourne'a [6], niemniej jednak sama idea może być zaimplementowana dla innych platform. W obecnym kształcie wymagane jest, aby system był wyposażony w pocztę elektroniczną akceptującą polecenie `mailx`, oraz mechanizm `forward`, a także niezależny procesor poczty `procmail`.

5.3. Działanie AUTOMATU

Działanie AUTOMATU rozpoczyna się z chwilą nadejścia nowej wiadomości pocztowej. Wiadomość jest przez mechanizm `forward` kierowana do `procmail`. Ten wykorzystując plik `procmailrc` rozpoznaje poprawną formę pola poczty `from` oraz analizuje postać pola

subject. Pole subject pozwala różnicować wiadomości na *zgłoszenia prac do wykonania* i *zapytania o stan pracy*

5.3.1. Zgłoszenie pracy do wykonania

Pole subject w postaci:

„**<project_name> job <job_label>"

oznacza zgłoszenie do wykonania pracy identyfikowanej przez <project_name> i opatrzonej etykietą <job_label>. Procesor poczty uruchamia specjalny skrypt StartJob sterujący działaniem AUTOMATU.

Skrypt StartJob jest wywoływany z nazwą projektu jako parametrem, a na jego standardowe wejście skierowana zostaje nadesłana wiadomość. Skrypt StartJob pobiera z tekstu wiadomości dane wejściowe i umieszcza je w pliku nazwanym input.

Jeżeli uruchomienie pracy nie jest możliwe (brak pola <job_label> lub przekroczona liczba prac przeznaczonych do uruchomienia równoległego), skrypt StartJob kończy działanie i drogą pocztową informuje o tym klienta. W przeciwnym razie wywoływany jest manager projektu (program o nazwie <project_name>, a więc takiej samej, jak nazwa projektu podana w nadesłanej wiadomości). Pobiera on dane ze standardowego strumienia wejściowego (do którego zostaje przyporządkowany plik input), a wyniki działania wysyła do standardowego strumienia wyjściowego (do którego zostaje przyporządkowany plik output).

Fakt uruchomienia pracy jest zgłaszany klientowi drogą pocztową. Następnie skrypt StartJob czeka na zakończenie wykonywania pracy.

Po zakończeniu pracy StartJob ponownie wywołuje managera projektu, przekazując mu jako parametry powstałe pliki input i output, a jako wynik otrzymuje informację, co było powodem zakończenia pracy. W przypadku prawidłowego zakończenia skrypt StartJob informuje o tym klienta i przesyła wyniki umieszczone w pliku output. Dodatkowo wyniki wszystkich zakończonych prac przechowywane są w archiwum.

Przepływ sterowania w AUTOMACIE w przypadku otrzymania wiadomości pocztowej o prawidłowym formacie i wykonania pracy bez przerwania przedstawiony jest na rysunku 1.

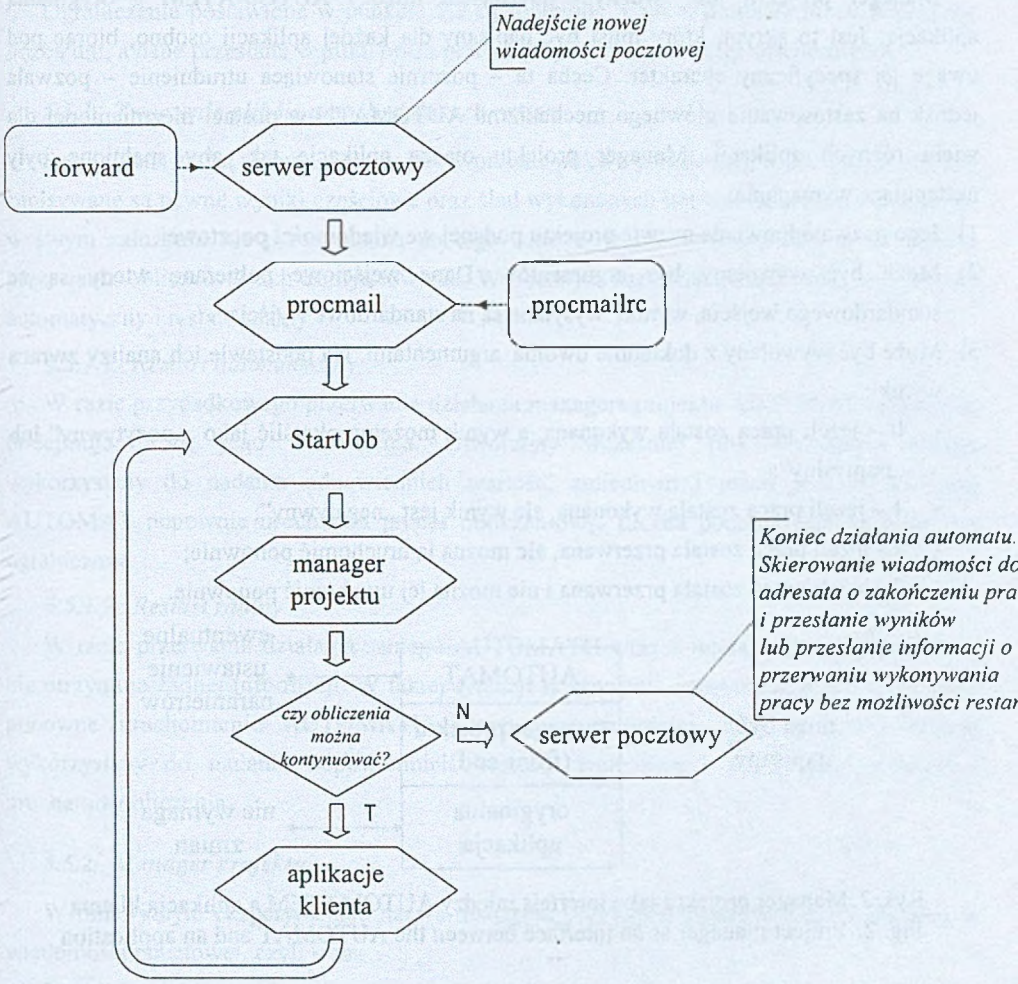
5.3.2. Zapytanie o stan pracy

Pole subject w postaci:

„**<project_name> job <job_label> ?"

oznacza zapytanie o stan pracy identyfikowanej przez <project_name> i opatrzonej etykietą <job_label>. Procesor poczty uruchamia skrypt QueryJob, pozwalający uzyskać informacje o zakończeniu, wykonywaniu lub przerwaniu pracy. W określonych przypadkach

wynikiem skierowania zapytania może być także pobranie wyników zakończonej pracy lub ponowne uruchomienie przerwanej pracy.



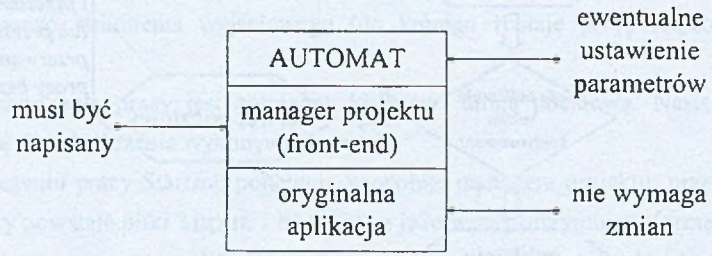
Rys. 1. Przepływ sterowania w AUTOMACIE w przypadku zgłoszenia pracy do wykonania
Fig. 1. AUTOMAT control flow, when receiving a job execution demand

W pierwszym etapie skrypt QueryJob poszukuje informacji o pracy w archiwum. Jeżeli ślad jej wykonania został tam umieszczony, informacja o zakończeniu pracy wraz z plikiem zawierającym wyniki zostaje przesłana do klienta. W przeciwnym razie skrypt QueryJob ustala, czy rozpatrywana praca jest ciągle w trakcie wykonywania lub czy została uruchomiona, a następnie przerwana. W przypadku pierwszym skrypt QueryJob ogranicza się do przesłania klientowi komunikatu, a w przypadku drugim podjęta zostaje próba wznowienia obliczeń.

5.4. Specyfikacja managera projektu

Manager projektu jest swoistym interfejsem między AUTOMATEM a oryginalną aplikacją. Jest to skrypt, który musi być napisany dla każdej aplikacji osobno, biorąc pod uwagę jej specyficzny charakter. Cecha ta – pozornie stanowiąca utrudnienie – pozwala jednak na zastosowanie głównego mechanizmu AUTOMATU w postaci niezmienionej dla wielu różnych aplikacji. Manager projektu otacza aplikację tak, aby spełnione były następujące wymagania:

1) Jego nazwa odpowiada nazwie projektu podanej we wiadomości pocztowej.
2) Może być wywołany bez argumentów. Dane wejściowe pobierane wtedy są ze standardowego wejścia, wyniki wysyłane są na standardowe wyjście.
3) Może być wywołany z dokładnie dwoma argumentami. Na podstawie ich analizy zwraca wyniki:
   - 0 – jeżeli praca została wykonana, a wynik możemy określić jako „pozytywny" lub „pomyślny";
   - 1 – jeżeli praca została wykonana, ale wynik jest „negatywny";
   - 2 - jeżeli praca została przerwana, ale można ją uruchomić ponownie;
   - 3 - jeżeli praca została przerwana i nie można jej uruchomić ponownie.



Rys. 2. Manager projektu jako interfejs między AUTOMATEM a aplikacją klienta
Fig. 2. Project manager as an interface between the AUTOMAT and an application

5.5. Przykładowe zastosowanie

Opracowany AUTOMAT został zainstalowany i przetestowany. W obecnym kształcie musi być on zainstalowany na każdej stacji, do której będą kierowane żądania wykonania pracy. W ramach eksperymentu AUTOMAT został przystosowany do akceptowania wiadomości o polu `subject` w postaci: „`** ecm job <job_label>`" i polu `from` zawierającym ciąg znaków: „Zimmerman". Problemem do rozwiązania (patrz punkt 1.1) jest tu przeprowadzenie obliczeń mających na celu rozkład bardzo dużych liczb na czynniki. Obliczenia te wykorzystują tzw. algorytmy ECM (Eliptic Curve Method), które wymagają między innymi wielokrotnego uruchamiania pewnych procedur dla różnych parametrów.

Pojedyncze uruchomienie aplikacji dla pojedynczego zestawu parametrów stanowi pracę (patrz punkt 1.1).

Ograniczenia postawione w punkcie 5.2 są spełnione. Czas wykonania pracy mierzy się liczbą dni, a dane przesłane w pliku opisu pracy są wystarczające do jej uruchomienia.

5.5.1. Tworzenie plików umożliwiających restart

Specyficzną cechą wybranej pracy jest tworzenie pliku wynikowego, w którym na bieżąco zapisywane są pewne wyniki częściowe oraz ślad wykonanych iteracji. Informacja ta — choć w swym założeniu nie przeznaczona do tego celu — jest wykorzystywana w przypadku przerwania obliczeń do ich kontynuowania. W obecnym kształcie system umożliwia restart automatyczny i restart zdalny.

5.5.1.1. Restart automatyczny

W razie przypadkowego przerwania działania managera projektu AUTOMAT samoistnie podejmuje próbę jego wznowienia. Utworzony wcześniej plik wynikowy zostaje wykorzystany do nadania odpowiednich wartości zmiennym i jeżeli jest to możliwe, AUTOMAT ponownie uruchamia proces obliczeniowy. Liczba podejmowanych prób jest ograniczona.

5.5.1.2. Restart zdalny

W razie przerwania działania samego AUTOMATU wraz z managerem projektu klient nie otrzymuje żadnej informacji. W takiej sytuacji skierowanie zapytania o pracę spowoduje ponowne uruchomienie AUTOMATU. Utworzony wcześniej plik wynikowy zostaje wykorzystany do nadania odpowiednich wartości zmiennym i AUTOMAT ponownie uruchamia obliczenia.

5.5.2. Manager projektu

W omawianym eksperymencie nazwa managera projektu odpowiada nazwie podanej w wiadomości pocztowej, czyli `ecm`.

Przy wywołaniu bez parametrów na standardowe wejście podawany jest plik `input` z zestawem parametrów stanowiących zestaw danych wejściowych, a wyniki kierowane są do pliku `output`. Obydwa pliki są plikami tekstowymi.

Przy wywołaniu z dwoma parametrami są nimi plik `input` i utworzony plik `output`. Manager projektu porównuje ich zawartość i wnioskuje, jaki był powód zakończenia pracy.

1) Praca została zakończona, jeżeli liczba iteracji zadana w pliku `input` odpowiada liczbie iteracji, których ślad wykonania został w pliku `output`.
2) Pozytywne zakończenie pracy sygnalizowane jest obecnością w pliku `output` ciągu znaków Factor found i oznacza znalezienie liczby pierwszej.

3) Negatywne zakończenie pracy oznacza wykonanie wszystkich zadanych iteracji i nieznalezienie liczby pierwszej.
4) Przerwanie pracy z możliwością restartu wnioskowane jest na podstawie obecności pliku `output`, w którym znajduje się ślad mniejszej liczby wykonanych iteracji niż liczba zadanych iteracji w pliku `input`.

   Przerwanie pracy bez możliwości restartu wnioskowane jest w przypadku błędnej lub niekompletnej postaci pliku `output`.

6. Uwagi końcowe

W artykule przedstawiono sposób realizacji przetwarzania wsadowego. Wykorzystuje on do uruchomienia zlecenia pocztę elektroniczną. Podstawową zaletą wykorzystania poczty jest jej prostota oraz powszechna znajomość wśród użytkowników komputerów — nie tylko informatyków. Można więc powiedzieć, że jest odpowiedzią na postulat przybliżenia możliwości współczesnej techniki obliczeniowej użytkownikom spoza grona „wtajemniczonych".

Niewątpliwie AUTOMAT ma pewne wady. Po pierwsze, mimo iż opiera się na założeniu nieingerowania w kod uruchamianej aplikacji, wymaga pisania dla każdej z nich managera projektu, a to jest sprzeczne z jego główną zaletą, czyli małymi wymaganiami w kwestii informatycznego przygotowania użytkownika. Oczywiście można też powiedzieć, że użytkownik będzie wielokrotnie korzystał z raz napisanego przez specjalistę managera, niemniej jednak wydaje się, że ciekawe byłoby kontynuowanie tego wątku. W szczególności można by przeanalizować wybrane aplikacje, takie jak programy matematyczne ogólnego zastosowania (Mathematica, MuPAD, MATLAB) oraz np. procesory obliczeń wytrzymałościowych (NASTRAN). Wynikiem analizy byłyby interfejsy dla tych aplikacji, które jednocześnie służyłyby jako wzorce do tworzenia kolejnych.

Drugim tematem dalszego rozwoju powinno być rozszerzenie AUTOMATU o możliwości działania w klastrze. Zlecenie adresowane do konkretnego komputera de facto przyjmowane byłoby przez klaster i wykonywane na maszynie najlepiej do tego nadającej się, w ambitniejszej wersji - także z możliwością dynamicznego wyrównywania obciążenia. Należałoby tu skorzystać z rozwiązań zastosowanych w omówionych w artykule systemach zarządzania pracami.

LITERATURA

1. Kozielski S., Szczerbiński Z.: Komputery równoległe, Wydawnictwa Naukowo-Techniczne, Warszawa 1993.
2. Baker M.A., Fox G.C., Yan H.W.: Review of Cluster Management Software, NHSE Review, 1996, Volume 1, No. 1.
3. Condor Version 6.1.8 Manual, Condor Team, University of Wisconsin-Madison 1998. Dokument elektroniczny dostępny pod adresem www.cs.wisc.edu/condor/manual/v6.1.
4. CODINE. Quick Start Guide, Version 4.2, GENIAS Software 1997.
5. CODINE 4.2, GENIAS Software 1998. Dokument elektroniczny dostępny pod adresem www.genias.de/genias/english/codine/Welcome.html.
6. Kaniewski M., Wieremiejczyk K.: Po prostu Unix, MIKOM, Warszawa 1992.
7. Gabassi M., Dupuy B.: Przetwarzanie rozproszone w systemie UNIX, Lupus, Warszawa 1995.
8. Foster I.: Designing and Building Parallel Programs. Concept and Tools for Parallel Software Engineering, Addison-Wesley 1995.
9. Portable Batch System, External Reference Specification, NASA Ames Research Center 1998.
10. GNQS Manual Pages. Dokument elektroniczny dostępny pod adresem www.gnqs.org/docs/manuals/manpages.

Recenzent: Dr inż. Krzysztof Nałęcki

Wpłynęło do Redakcji 6 października 1999 r.

Abstract

The concurrent computing is seen today as a way to speed up and better take advantage of the potential computational power. The concurrent computing can be assigned to one of three levels:

- the instruction level,
- the task level,
- the job level.

The burden of the organisation of the computation on the job level is appointed to cluster management systems, fulfilling some important features. The batch processing support is one

of them. The batch processing to be applied on a cluster of workstations must use a low level mechanism to evoke a job on a remote machine. Cluster management systems employ telnet, remsh, sockets and ssh to perform the task. The paper shows an alternative approach using the electronic mail, which benefits from the platform independence, not bad security and simplicity.

The paper presents a proposal of a program called AUTOMAT, which aim is to receive via e-mail a description of the job to do. The AUTOMAT runs the appropriate application automatically and after the completion sends back the result data.

Besides starting jobs, AUTOMAT let the user to query the state of the running job and in case it for some reasons died — to restart it.