System NetSolve

Marcin Rociek
Informatyka, II rok

2 czerwca 2000

Spis treści

1 Wstęp

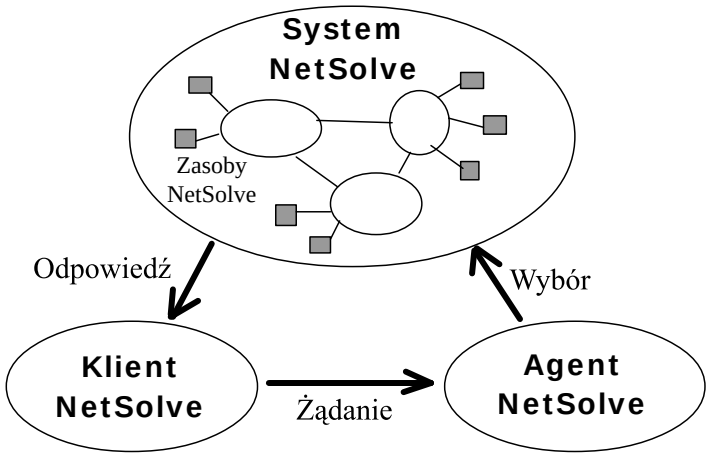
System NetSolve umożliwia użytkownikom prosty dostęp poprzez sieć do zasobów obliczeniowych (sprzętu i oprogramowania) rozproszonych po świecie. Potrzeba istnienia łatwego w użyciu mechanizmu zdalnego dostępu do nich była główną motywacją twórców NetSolve. Wysoka wydajność tego systemu jest wynikiem optymalnej strategii wyboru zasobów z wszystkich aktualnie dostępnych w sieci. NetSolve wyszukuje dostępne zasoby, wybiera najlepsze z nich, za pomocą których rozwiązywany jest zadany problem, a następnie zwraca rozwiązanie użytkownikowi.

NetSolve jest aplikacją typu klient-serwer stworzoną do rozwiązywania problemów obliczeniowych poprzez sieć. Stworzono wiele interfejsów dla różnych platform, tak aby z NetSolve mogli korzystać użytkownicy C, Fortrana, MATLABA itp. Właściwym oprogramowaniem liczącym może być jakikolwiek pakiet obliczeniowy zapewniający wystarczającą szybkość działania.

2 Budowa i działanie

2.1 Architektura

System NetSolve jest zbiorem luźno połączonych komputerów, tzn. mogą być one połączone zarówno poprzez sieć lokalną, jak i dużą sieć międzynarodową (Internet). Ponadto w systemie mogą przebywać jednocześnie maszyny wykorzystujące różny format zapisu danych.



Rysunek 1: System NetSolve

Rysunek 1 przedstawia ogólną koncepcję systemu: klient wysyła do agenta żądanie, ten z kolei wybiera z systemu najbardziej odpowiednie zasoby, biorąc pod uwagę rozmiar problemu i jego rodzaj.

Klient NetSolve jest biblioteką dołączoną do aplikacji użytkownika. Aplikacja wykonuje wywołania odpowiednich funkcji w NetSolve API w celu udostępnienia odpowiednich usług systemu. Poprzez API użytkownik ma dostęp do zagregowanych zasobów obliczeniowych, bez konieczności posiadania wiedzy na temat sieci komputerowych czy obliczeń rozproszonych. W rzeczywistości użytkownik nie musi nawet wiedzieć, że zostały użyte zdalne zasoby obliczeniowe.

Agent NetSolve posiada bazę serwerów NetSolve wraz z ich możliwościami (wydajność sprzętowa i przydzielone oprogramowanie) oraz statystyki obciążeń. Te informacje wykorzystywane są w celu przydzielenia zasobów serwera na żądanie klienta. Agent wyszukuje serwery, które obsłużą dane żądanie najszybciej i wybiera najlepszego z nich. W sieci może istnieć jednocześnie wiele agentów, przy czym dobrą strategią jest posiadanie jednego agenta w każdej podsieci lokalnej. Nie jest to jednak obowiązkowe i możliwa jest oczywiście sytuacja, w której na cały system przypada tylko jeden agent.

Serwer NetSolve to proces typu *deamon*, który czeka na żądania klientów. Serwer może pracować na pojedynczych stacjach roboczych, grupach stacji roboczych bądź maszynach wieloprocesorowych. Kluczowym składnikiem serwera jest generator kodu źródłowego, który przetwarza zawartość pliku opisującego problem (*PDF - Problem File Description*). Pliki PDF zawierają informacje, które umożliwiają systemowi NetSolve tworzenie nowych modułów, rozszerzających jego funkcjonalność.

Ważnym zagadnieniem w takim systemie opartym na serwerach jest to, że każdy agent 'widzi' system z innej perspektywy. Może to prowadzić do tego, że niektóre instancje agentów będą posiadały więcej informacji o systemie niż inne, w zależności od lokalizacji.

2.2 Komunikacja

Komunikacja w systemie NetSolve została rozwiązana na poziomie socketów. Wybranym protokołem jest TCP/IP, zapewniający niezawodną komunikację między procesami.

Jak już wspomniano wcześniej, system NetSolve może być niejednorodny, tzn. mogą w nim przebywać komputery wykorzystujące różny format zapisu danych. Aby zapewnić komunikację między takimi komputerami, w pierwszym etapie komunikacja między nimi realizowana jest w oparciu o protokół XDR. Kiedy zaś dwa hosty stwierdzą, że używają tego samego formatu zapisu danych, rezygnują z protokołu XDR, co jest podyktowane spodziewaną transmisją dużej ilości danych od użytkownika.

2.3 Bilansowanie obciążenia

Jednym z ciekawszych zagadnień systemu NetSolve jest bilansowanie obciążenia[1] serwerów. Ponieważ NetSolve wykonuje obliczenia w sieci zawierającej dużą ilość maszyn o różnych charakterystykach, wydaje się logiczne sądzić, iż jedna z tych maszyn nadaje się najlepiej do rozwiązania danego problemu. Zanim jednak zostanie rozpatrzony sposób, w jaki NetSolve decyduje którą maszynę ma wybrać, przyjrzyjmy się bliżej co w rzeczywistości określa najlepszą maszynę.

2.3.1 Wyznaczanie najlepszej maszyny

Hipotetyczną najlepszą maszyną jest taka, która dla danego problemu P wykonuje go w najkrótszym czasie T. Musimy więc oszacować czas T dla każdej maszyny M znajdującej się w systemie NetSolve. Najprościej jest podzielić czas T na T_n i T_c, gdzie

- T_n jest czasem potrzebnym na przesłanie danych do M oraz na odebranie wyniku
- T_c jest czasem obliczenia przez M

Czas T_n może być obliczony na podstawie

1. opóźnień w sieci oraz przepustowości między hostem a M
2. rozmiaru danych wysyłanych
3. rozmiaru wyniku, który zostanie odebrany

Obliczenie czasu T_c wymaga znajomości

1. rozmiaru problemu
2. złożoności algorytmu, który ma być użyty
3. wydajności M, która zależy od
   - obciążenia M
   - wydajności nieobciążonej M

2.3.2 Model wydajności

Autorzy systemu NetSolve opracowali prosty model teoretyczny pozwalający na oszacowanie wydajności, znając wydajność maksymalną (dla nieobciążonej M) oraz aktualne obciążenie. Model ten szacuje wydajność p, jako funkcję obciążenia w, maksymalnego obciążenia P, oraz ilości procesorów n:

$$p = \frac{P \times 100 \times n}{100 \times n + max(w - 100 \times (n-1), 0)}$$

Model ten poparto szeregiem eksperymentów (więcej na ten temat można znaleźć w pracy [1]).

[1] w oryginale *load-balancing*

2.3.3 Wyznaczenie czasu obliczenia T

Obliczanie T ma miejsce w serwerze komunikacyjnym dla każdego problemu do rozwiązania oraz dla każdego serwera obliczeniowego. Obliczenie to wykorzystuje wszystkie poprzednio wypisane parametry. Parametry te dzielą się na trzy klasy:

- Parametry zależne od klienta
    1. Rozmiar wysyłanych danych
    2. Rozmiar odpowiedzi
    3. Rozmiar problemu
- Parametry **statyczne** zależne od serwera
    1. Charakterystyka sieci pomiędzy hostem a M
    2. Złożoność algorytmu, który będzie wykorzystany przez M
    3. Maksymalna wydajność M
- Parametry **dynamiczne** zależne od serwera
    1. Obciążenie M

Parametry zależne od klienta są włączone w problem przesłany przez klienta do serwera komunikacyjnego, obliczenie ich następuje więc w sposób bezpośredni. Parametry statyczne zależne od serwera są zwykle wyznaczane jednorazowo, kiedy nowy serwer obliczeniowy kontaktuje się z serwerwami już przebywającymi w systemie.

Charakterystyka sieci – jest wyznaczana kilka razy, tak aby dostać rozsądną średnią wartość opóźnień i przepustowości. Parametry te traktowane są jako *statyczne*, gdyż zakłada się, że nie zmieniają się one drastycznie w czasie.

Złożoność algorytmu – gdy nowy serwer obliczeniowy dołącza do systemu, wysyła informacje na temat złożoności wszystkich problemów, które jest skłonny rozwiązać. Złożoność ta nie zmienia się w czasie, gdyż jest zależna od zainstalowanego na serwerze oprogramowania.

Maksymalna wydajność - pod tym pojęciem rozumiana jest wydajność maszyny, w której żaden inny proces nie zajmuje czasu procesora. Wartość ta jest określana przez każdy serwer obliczeniowy przy uruchamianiu.

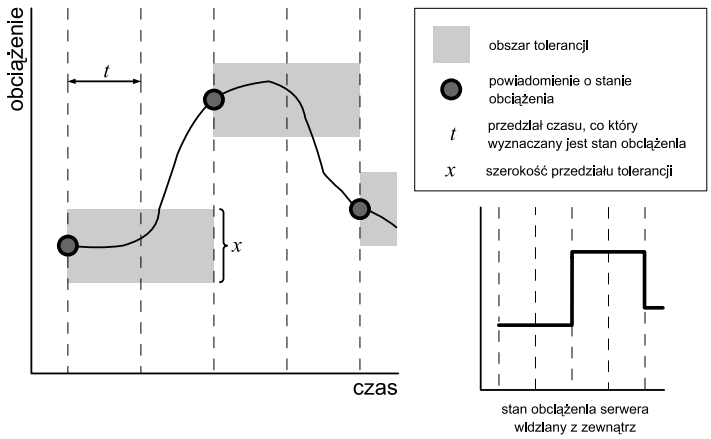
2.3.4 Model obciążenia

Parametry obciążenia są jedynymi dynamicznymi parametrami zależnymi od serwera, które muszą zostać wyznaczone w celu przewidzenia czasu wykonania T.

Każdy agent posiada okresowo uaktualniane informacje na temat obciążenia każdego z serwerów obliczeniowych. Informacje te mogą być jednakże nieaktualne i prowadzić do błędnego oszacowania czasu T. Nie mniej jednak lepiej jest ponieść ryzyko złego oszacowania tej wartości niż płacić za jej stałą dokładność.

Problem ten rozwiązano w sposób następujący: każdy serwer obliczeniowy sprawdza stopień swojego obciążenia w stałych odstępach czasu t. Często jednak zachodzi taka sytuacja, w której serwer pozostaje w tym samym stanie przez długi czas. Aby zapobiec bezcelowemu nawiązywaniu połączenia z agentami za każdym razem, serwer obliczeniowy powiadamia o swoim stanie obciążenia tylko wtedy, gdy zmienił się on w sposób znaczący. Zaznaczone na rysunku 2 szare pola, to obszar tolerancji. Serwer po wyznaczeniu stanu swojego obciążenia porównuje uzyskaną wartość z wartością, którą ostatnio powiadamiał inne serwery. Jedynie w przypadku, gdy wartości te różnia się między sobą o więcej niż $x/2$ (gdzie x jest szerokością przedziału tolerancji), serwer powiadamia innych o swoim stanie obciążenia.

Bardzo ważne jest odpowiednie dobranie parametrów t i x. Zbyt mała wartość przedziału czasowego t prowadzi do zbyt częstego sprawdzania stanu obciążenia przez serwer i tym samym do marnowania jego mocy obliczeniowej. Zbyt mała wartość przedziału x może zaś prowadzić do częstej transmisji informacji o niewielkich zmianach stanu serwera, co w praktyce oznacza właściwie tylko zapychanie łącz.



Rysunek 2: Wyznaczanie obciążenia serwera obliczeniowego

2.4 Awarie

W tak rozproszonym systemie jakim jest NetSolve, branie pod uwagę awarii jest oczywistą kwestią. Uszkodzenie jednego z komponentów systemu nie powinno jednakże prowadzić do katastrofalnego w skutkach uszkodzenia całego systemu. Ponadto ilość efektów ubocznych spowodowanych uszkodzeniem jednego ze składników powinna być jak najmniejsza i nie powinna prowadzić do spadku wydajności systemu.

2.4.1 Wykrywanie awarii

Usterki mogą wystąpić na wielu poziomach systemu NetSolve. Są one przeważnie spowodowane uszkodzeniem sieci, zniknięciem serwera, bądź jego uszkodzenem. Procesy systemu NetSolve (np. serwery, klienci) wykrywają tego typu uszkodzenia w momencie nawiązywania połączenia z serwerem. Gdy połączenie nie może zostać nawiązane bądź czas oczekiwania na odpowiedź przekrocza ustalone limity, o usterce zawiadamiany jest agent NetSolve, który zaznacza dany serwer w swojej bazie danych. Serwer jest z niej usunięty dopiero po upłynięciu pewnego przedziału czasu, i tylko wtedy, gdy nie został uprzednio włączony. Jednym z założeń systemu NetSolve jest to, że serwer może być w każdej chwili bezpiecznie wyłączony i włączony. Dlatego też, wszystkie raporty o błędach zawierają informacje pozwalające stwierdzić, czy serwer został włączony po wystąpieniu błędu.

2.4.2 Wrażliwość na błędy

Aby jak najbardziej zminimalizować uboczne skutki uszkodzeń, opracowano następujący schemat działania:

Gdy agent dostaje od klienta żądanie rozwiązania jakiegoś problemu, zwraca klientowi listę dostępnych serwerów obliczeniowych, posortowaną od najbardziej do najmniej użytecznego. Klient sprawdza sobie po kolei każdy z nich, aż napotka serwer który zgodzi się na rozwiązanie problemu. Strategia ta pozwala klientowi sprawdzić wiele serwerów, bez konieczności wysyłania żądania agentowi za każdym razem, gdy jakiś serwer nie działa. Gdy klient dojdzie do końca listy i wciąż nie znajdzie działającego serwera, wysyła ponownie żądanie rozwiązania do agenta, a ponieważ go wcześniej sam powiadomił o wszystkich usterkach, dostanie już inną listę serwerów.

Gdy połączenie z serwerem obliczeniowym zostało nawiązane, wciąż jednak nie ma gwarancji, że problem zostanie rozwiązany. Może np z jakiegoś powodu zostać wyłączony proces obliczeniowy. W takim wypadku klient zwraca się do kolejnego serwera. Problem wędruje przez wszystkie serwery aż do momentu w którym zostanie rozwiązany, bądź nie będzie już serwera, który mogły by go rozwiązać. Cały ten proces jest ukryty przed użytkownikiem, i oczywiście zwiększa czas obliczenia problemu.

2.4.3 Zarządzanie awariami

Po wystąpieniu awarii, agenci uaktualniają swój obraz całego systemu. Cały czas śledzą informacje odnośnie każdego hostów, czy jest osiągalny, czy też nie. Oprócz tego śledzą informacje na temat procesów serwerów działających na tych hostach: czy działają, czy też nie. Gdy host jest nieosiągalny lub serwer tam uruchomiony nie działa, agent usuwa informacje na jego temat ze swojej bazy danych po 24 godzinach.

Agent posiada również informacje o liczbie błędów serwerów obliczeniowych. Gdy liczba ta przekroczy wartość krytyczną, serwer jest usuwany z bazy danych. Dlatego też serwer obliczeniowy, który jest źle zaimplementowany i np błędnie wywołuje funkcje bibliotek obliczeniowych, może zniknąć z systemu.

3 Użytkowanie

3.1 Zarządzanie systemem

Główną ideą architektury całego systemu NetSolve jest to, że każdy proces (agent lub serwer obliczeniowy) jest niezależną jednostką. System może więc być modyfikowany bez narażania jego integralności; w każdym momencie można utworzyć nowy proces, bądź usunąć już istniejący. W szczególnym przypadku może istnieć np. system bez agentów, do którego po prostu użytkownicy nie mają dostępu. W takim systemie można dopiero stworzyć nowego agenta, aby przywrócić systemowi używalność.

Zarządzanie systemem NetSolve może być uciążliwe, dlatego potrzebne jest odpowiednie narzędzie, umożliwiające globalne spojrzenie na cały system. Aby narzędzie to było wystarczająco wygodne w użyciu, stworzono do tego celu skrypty CGI dostępne z przez WWW. Skryptom tym podajemy lokalizację agenta (nazwę hosta, na którym on działa) w celu identyfikacji systemu, który zamierzamy sprawdzić. Wynikiem działania skryptów jest lista agentów oraz serwerów obliczeniowych dostępnych w danym systemie, oraz lista problemów które mogą być rozwiązane za pomocą tego systemu wraz ze specyfikacjami wywołań dla C i Fortrana. Ponadto lista możliwych do rozwiązania problemów jest bezpośrednio dostępna w pakietach Matlab i Mathematica, oraz w Java GUI.

Administratorzy systemu NetSolve mają do dyspozycji zestaw sześciu programów zarządzająco - monitorujących:

- `NS_conf` – po podaniu lokalizacji agenta należącego do jakiegoś systemu, program wypisuje listę wszystkich hostów (agentów i serwerów obliczeniowych) działających w danym systemie
- `NS_problems` – po podaniu lokalizacji agenta, program wypisuje listę problemów, które mogą być rozwiązane poprzez połączenie z tym agentem
- `NS_probdesc` – po podaniu lokalizacji agenta oraz skrótu oznaczającego problem, program ten wypisuje dokładny opis tego problemu
- `NS_killagent` – po podaniu lokalizacji agenta, następuje jego wyłączenie
- `NS_killserver` – po podaniu lokalizacji serwera obliczeniowego oraz agenta, następuje wyłączenie serwera (agent jest wykorzystywany w celu wejścia do systemu)
- `NS_killall` – po podaniu lokalizacji agenta, następuje wyłączenie jego oraz wszystkich znanych mu hostów (innych agentów oraz serwerów)

3.2 Zarządzanie serwerem obliczeniowym

3.2.1 Specyfikacja problemu

Problem zdefiniowany jest jako trójka: <*nazwa*, *wejście*, *wyjście*>, gdzie

- *nazwa* jest ciągiem znaków identyfikujących problem
- *wejście* jest listą obiektów wejściowych
- *wyjście* jest listą obiektów wyjściowych

Obiekt jest zdefiniowany przez *typ obiektu* oraz *typ danych*. Dostępne w aktualnej wersji typy obiektów i danych to (za [2]):

Typy danych:

`NETSOLVE_I` – liczba całkowita typu integer

`NETSOLVE_CHAR` – znak

`NETSOLVE_BYTE` – bajt

NETSOLVE_FLOAT – liczba rzeczywista pojedynczej precyzji

NETSOLVE_DOUBLE – liczba rzeczywista podwójnej precyzji

NETSOLVE_SCOMPLEX – liczba zespolona pojedynczej precyzji

NETSOLVE_DCOMPLEX – liczba zespolona podwójnej precyzji

Typy obiektów:

NETSOLVE_SCALAR – skalar

NETSOLVE_VECTOR – wektor

NETSOLVE_MATRIX – macierz

Typy obiektów, których dane muszą być typu NETSOLVE_CHAR:

NETSOLVE_FILE – plik

NETSOLVE_PACKEDFILES – skompresowane pliki

NETSOLVE_UPF – funkcja zadana przez użytkownika

NETSOLVE_STRING – ciąg znaków

NETSOLVE_STRINGLIST – lista ciągów znaków

Wejścia i wyjścia problemu zdefiniowano jako listy *obiektów*. Interfejsy NetSolve w pakietach Matlab, Mathematica oraz w Javie mogą operować obiektami bezpośrednio i z tego względu stosunkowo łatwo jest wywołać NetSolve z ich poziomu, jeśli tylko znany jest opis problemu. Natomiast z poziomu interfejsów NetSolve, któe nie są zorientowane obiektowo (C, Fortran), konieczne jest zastosowanie odpowiedniej *sekwencji wywołującej*, która opisuje indywidualne cechy obiektów.

3.2.2 Konfiguracja serwera

Konfiguracja serwera dokonywana jest przez odpowiedni plik konfiguracyjny. Standardowo tym plikiem jest plik $NETSOLVE_ROOT/server_config. W pliku tym znajduje się między innymi:

- '@AGENT:<nazwa_hosta>'[*] – określa agenta, z którym serwer musi się skontaktować w celu zarejestrowania się w systemie NetSolve. Agent jest określony przez nazwę hosta, na którym działa. Podanie dodatkowo znaku '*' powoduje, że serwer będzie widoczny także przez wszystkich agentów znanych temu agentowi, w przeciwnym wypadku serwer będzie widoczny tylko przez tego jednego agenta.
- '@PROC:<numer>' – określa ilość procesorów, które mogą być używane przez serwer do obliczeń równoległych.
- '@MPIHOSTS <nazwa_pliku>' – określa ścieżkę do pliku zawierającego listę hostów, które mogą być używane przez MPI.
- '@MPIPATH <sciezka>' – określa ścieżkę do katalogu zawierającego pliki wykonywalne MPI (np. mpirun).
- '@WORKLOADMAX:<max>' – określa wartość krytyczną obciążenia, po przekroczeniu której serwer odmawia przyjmowania nowych żądań. Wartość -1 oznacza, że serwer przyjmuje żądania niezależnie od obciążenia.
- '@SCRATCH:<sciezka>' – określa miejsce w którym serwer może umieszczać pliki i katalogi tymczasowe. Domyślnie /tmp/.
- '@CONDOR:<sciezka>' – określa, czy serwer NetSolve korzysta z zasobów obliczeniowych Condor.
- '@PROBLEMS:' – określa początek listy plików PDF (*Problem Description File*), które są zezwolone w danej instalacji serwera. Jeśli jakiś plik jest na tej liście, oznacza to, że zawarte w nim problemy mogą być rozwiązane przez ten serwer.

- `'@RESTRICTIONS:'` – określa początek listy ograniczeń dostępu stosownych dla danego serwera. Lista ta składa się z linii postaci:

  ```
  <domena> <ilość dopuszczonych równocześnie żądań>
  ```

  Na przykład linia

  ```
  *.edu 10
  ```

  oznacza, że tylko 10 klientów znajdujących się w domenie `.edu` może zostać obsłużonych równocześnie.

3.3 Wykorzystywanie systemu

Aktualnie dostępne oprogramowanie serwera i agenta działa wyłącznie pod kontrolą systemów z rodziy UNIX. Natomiast sam klient jest szerzej dostępny.

Możliwe jest wykorzystywanie systemu NetSolve w środowiskach interaktywnych: takich jak MATLAB, bądź bezpośrednio w shellu. Środowiska interaktywne mają tę zaletę, że nie wymagają od użytkownika programowania, dając mimo to możliwość korzystania z ogromnych zasobów obliczeniowych.

Ponadto można oczywiście wykorzystać system NetSolve pisząc programy w języku C lub Fortran. Założeniem systemu jest to, że wymaga on od użytkownika jedynie minimalnych umiejętności programistycznych. Jest dostępnych tylko kilka prostych funkcji, intuicyjnych w użyciu.

Generalnie istnieją dwie metody wywoływania funkcji NetSolve: tzw. blokowa i asynchroniczna. W pierwszej z nich wywołujemy funkcje, podając jako parametr problem do rozwiązania oraz dane, funkcja zaś zwraca nam wynik. W metodzie drugiej wywołujemy żądanie rozwiązania, a następnie możemy sprawdzać, czy rozwiązanie już 'przybyło', i dopiero wtedy je odczytać. Umożliwia nam to przeprowadzanie innych obliczeń w czasie, w którym NetSolve stara się rozwiązać problem.

Bibliografia

[1] Henri Casanova, Jack Dongarra. *NetSolve: A Network Server for Solving Computational Science Problems*, 1996.

[2] Dorian Arnold, Susan Blackford, Jack Dongarra. *Users' Guide to NetSolve V1.3*, 2000.

[3] CRPC News Archive. *Jack Dongarra's Netsolve Aims to Create Virtual SW Library*