Obliczenia rozproszone MPI

1. Informacje ogólne

MPI (Message Passing Interface) nazwa standardu biblioteki przesyłania komunikatów dla potrzeb programowania równoległego w sieciach rozproszonych. Aktualna wersja standardu to 2.1. MPI zawiera tylko specyfikacje interfejsu, nie zawiera żadnego konkretnego pakietu oprogramowania. Jest zbiorem funkcji API umożliwia programistom pisanie programów równoległych o wysokiej wydajności, które przekazują komunikaty pomiędzy procesami danego zadania.

MPICH to darmowa i przenośna implementacja standardu MPI zrealizowana w Argonne National Laboratory, ośrodku prowadzącym badania nad przetwarzaniem rozproszonym. Dostępna jest wersja dla platform UNIX jak i Windows, dla kompilatorów Fortran 77 i C. Biblioteka zawiera funkcje konieczne do uruchomienia, przesyłania komunikatów, synchronizacji i zakończenia programu. Możliwe jest przenoszenie programów na inne komputery.

PVM środowisko oprogramowania do realizacji programów rozproszonych. Komputery połączone siecią tworzą równoległą maszynę wirtualną. Poszczególne procesy mogą pracować w różnych systemach operacyjnych. PVM dostarcza programiście procedury i funkcje przesyłania komunikatów między procesami.

2. Komunikacja

Systemy rozproszone opierają się na dzieleniu zasobów i na przezroczystości ich rozmieszczenia Składowe systemu są rozłączne logicznie i fizycznie. Rozproszone systemy i aplikacje są tworzone z oddzielnych składowych oprogramowania, współpracujących przy wykonywaniu zadań. Komunikacja między parą procesów obejmuje działania po stronie procesu nadawczego i odbiorczego, dając:

a. *przenoszenie* danych ze środowiska procesu nadawczego do środowiska procesu odbiorczego; komunikujące procesy muszą wspólnie użytkować kanał komunikacyjny
b. *synchronizację* czynności odbiorczych z czynnościami nadawczymi, tak aby powstrzymać działanie procesu nadawczego lub odbiorczego do czasu, aż inny proces go uwolni

Równoważenie obciążenia może odbywać się z wywłaszczaniem lub bez wywłaszczania. W mechanizmie bez wywłaszczania nazywanym także dynamicznym przydzielaniem zadań – zadanie jest wyznaczane do przeniesienia na inny węzeł przed swoim startem. W mechanizmie z wywłaszczaniem zadanie może być przeniesione po rozpoczęciu jego wykonywania. Zarówno MPI jak i PVM stosują tę pierwszą metodę.

Konstrukcje programowania przyjmują postać działań wyślij i odbierz. Działania te dokonują przekazania komunikatu między parą procesów. Przekazanie komunikatu obejmuje przesłanie przez proces nadawczy zbioru wartości danych (komunikat) za pomocą mechanizmu komunikacji (kanału lub portu) i akceptację komunikatu przez proces odbiorczy. Mechanizm może być:

a. *synchroniczny z blokowaniem* - nadawca czeka po wysłaniu komunikatu do czasu, aż odbiorca wykona operację odbioru
b. *asynchroniczny bez blokowania* - komunikat jest umieszczany w kolejce komunikatów oczekujących na przyjęcie przez odbiorcę, a proces nadawczy może kontynuować działanie

Najczęściej stosowane schematy komunikacji stosowane w systemach rozproszonych:

1. **klient - serwer** służą do łączności między parami procesów; model komunikacji nastawiony na dostarczanie usług, można w nim wyróżnić następujące etapy:
   a. przesłanie zamówienia do procesu klienta do procesu serwera
   b. wykonanie zamówienia przez serwer
   c. przesłanie odpowiedzi do klienta

Schemat komunikacji klient-serwer można zrealizować za pomocą podstawowych operacji: przekazywanie komunikatów - wywołanie procedury zdalnej RPC (ang. Remote procedurę calling) realizowanej w postaci protokołu zamówienie - odpowiedź. Proces serwera w chwili uruchomienia przedstawia się usłudze nazewniczej, podając swój adres sieciowy i nazwę dostarczanej przez siebie usługi. Klient uzyskuje adres serwera przez zapytanie służby nazewniczej przy pomocy nazwy usługi.

2. **rozsyłanie grupowe** (ang. group multicast) do łączności między grupami współpracujących procesów. W tym schemacie komunikacji rozsyłania, procesy współdziałają , przekazując komunikaty. Adresatem komunikatu nie jest jeden proces, lecz grupa procesów. Jednej operacji wyślij odpowiada operacja odbierz każdego członka grupy procesów - rozsyłanie grupowe (ang. multicasting). Cechy rozsyłania grupowego:

a. odnajdywanie obiektu - klient rozsyła komunikat z nazwą katalogu pliku do grupy
b. procesów serwerów plików. Odpowiada tylko ten który przechowuje dany katalog
c. tolerowanie uszkodzeń - klient rozsyła zamówienie do grupy procesów usługowych, każdy
d. przetwarza je identycznie a następnie jeden lub więcej wysyła odpowiedź. Grupa serwerów
e. może zapewnić ciągłą obsługę, nawet w przypadku awarii pozostałych.
f. zwielokrotnione aktualizacje - zdarzenie w rodzaju "jest godzina 18:01" można wysyłać
g. do grupy zainteresowanych procesów.

W zależności od sposobu realizacji rozproszenia system może być:

a. rozproszonym systemem operacyjnym – rozproszenie realizowane jest na poziomie jądra systemu operacyjnego, każda poprawnie napisana wielozadaniowa aplikacja może podlegać migracji (migruje w całości lub wybrane jej podprocesy), rozproszenie jest przeźroczyste dla aplikacji
b. maszyną wirtualną – uwspólniane są wybrane zasoby np. pamięć i moc obliczeniowa – implementacja odbywa zwykle przez użycie specjalistycznych bibliotek programistycznych
c. aplikacja rozproszona – poszczególne funkcje aplikacji dzielone są pomiędzy różne komputery np. tzw architektura trójwarstwowa (warstwa składowania danych – serwery baz danych plików – filery, warstwa logiki biznesowej, warstwa prezentacji danych)

3. Dynamiczne przydzielanie zadań

Dwie najpopularniejsze implementacje równoważenia obciążenia w oparciu o dynamiczne przydzielanie zadań to MPI (Message Passing Interface) oraz PVM (Parallel Virtual Machine) są przykładem wykorzystania bibliotek programistycznych dla uzyskania rozproszenia. Aby możliwe było wykonywanie obliczeń konieczne są dwa elementy:

• środowisko uruchomieniowe, które zapewnia przydzielenie zadań odpowiednim węzłom przed ich utworzeniem (mpirun i ew. mpdboot)
• zestaw bibliotek umożliwiających wymianę komunikatów pomiędzy procesami
• program skompilowany z użyciem w/w bibliotek

W przeciwieństwie do rozproszonych systemów operacyjnych, które zwykle muszą wykorzystywać tę samą architekturę systemu np. te same wersje jądra wymiana komunikatów może być w prostszy sposób uniezależniona od systemu operacyjnego i jego architektury. W przypadku MPI możliwe jest wykorzystanie np. węzłów z systemem Windows 32 i 64b, linux 32 i 64b, SunOS, MacOSX i innych pod jednym warunkiem, dla każdej z architektur muszą znajdować się na tej samej ścieżce przygotowane odpowiednie binaria (skompilowany na dany system program).

3.1 Message Passing Interface

MPI – jest to specyfikacja standardu dla bibliotek wspomagających przesyłanie komunikatów zdefiniowana przez MPI Forum. Prace na standardem przesyłania komunikatów (MPI) w 1992r. Standard został zatwierdzony w 1994r. MPI zawiera interfejs programistyczny, zawiera specyfikacje semantyki protokołu i sposób implementacji (buforowanie komunikatów i sposób ich dostarczania). MPI obsługuje połączenia punkt-punkt i zbiorcze (globalne). MPI udostępnia abstrakcje dla procesów na dwóch poziomach Istnieje kilka implementacji tego standardu najpopularniejsze to:

a. MPICH wersja 1 i 2
b. OpenMPI
c. LAM

Literatura:
[1] Hunt, Craig; TCP/IP : administracja sieci. Warszawa : Oficyna Wydaw. READ ME, 1996.
[2] Blank, Andrew G, Podstawy TCP/IP / Andrew G. Blank ; przekł. z jęz. ang. Grzegorz Kowalski, Warszawa : Mikom, 2005.

4. Instrukcje do użytego oprogramowania

OpenMPI

W laboratorium 404 w budynku B5 oprogramowanie OpenMPI udostępnione jest na komputerach:
- W40425, W40426 (druga część laboratorium)

Aby zrealizować laboratoria z użyciem OpenMPI, należy zalogować się na konto użytkownika **mpiuser1** na obydwu komputerach.
Hasła: **mp1us3r1** (komputer W40425)
mp1us3r2 (komputer W40426)

Następnie należy sprawdzić, czy możliwe jest uruchomienie programów **mpirun** i **mpic++**. Jeżeli nie jest to możliwe, należy dodać pakiet do ścieżki systemowej komendą:

```
export PATH=/usr/lib64/openmpi/bin:$PATH
```

Następnie za pomocą narzędzia **ifconfig** sprawdzić obecność interfejsów sieciowych.
Jeżeli znalazłby się tam interfejs **virbr***, należy usunąć go uruchamiając skrypt:

```
sudo /usr/local/sbin/virt_down
```

po czym sprawdzić programem **ifconfig**, czy interfejs został wykluczony (działa hasło mpiuser1 na danym komputerze).
Te kroki należy wykonać **na każdym komputerze** wykonującym obliczenia.

Programy należy kompilować i uruchamiać w katalogu `/tmp/`

Na **każdej maszynie** biorącej udział w obliczeniach **w katalogu /tmp/** musi znaleźć się plik wykonywalny (binarka) – uzyskiwany po skompilowaniu kodu.

Kompilacja programów dla programów w C++:
```
mpic++ -o program.out source.cpp
```

Kompilacja programów dla programów w C++:
```
mpicc -o program.out source.c
```

gdzie:
`program.out` – nazwa wyjściowego pliku wykonywalnego (może być dowolna)
`source.cpp`, `source.c` – kody źródłowe programów (w c++ i w c) – zmienić odpowiednio no podanych przez prowadzącego
„Binarki" kompilatorów znajdują się w katalogu **/usr/lib64/openmpi/bin/**

Pliki skompilowane na jednym komputerze można skopiować na drugi komputer poleceniem (uruchamianym w katalogu `/tmp/`):

```
scp nazwa_pliku NazwaUzytkownika@adresIP:/tmp/
```

Na jednym z komputerów (na tym, na którym będą uruchamiane obliczenia) w katalogu **/tmp/** powinien znajdować się plik z listą hostów (adresami IP) biorących udział w obliczeniach

Tworzenie pliku z listą hostów - hosts.txt

Otwieramy dowolny edytor tekstu i tworzymy plik, który zawiera listę adresów IP komputerów, które będą brały udział w obliczeniach. Każdy adres wpisujemy w nowej linijce. Pierwszy adres to adres komputera, gdzie będzie uruchamiane polecenie `mpirun`.

Uruchamianie obliczeń (w katalogu /tmp/):

```
mpirun -hostfile hosts.txt -n 12 program.out
```

gdzie:
`hosts.txt` – nazwa (ścieżka) pliku tekstowego w którym umieszczone są adresy IP maszyn biorących udział w obliczeniach (parametr pomijamy jeśli liczymy na jednej maszynie), nazwa pliku może być dowolna
`-n 12` – liczba podzadań, na które należy podzielić zadanie główne (w tym przypadku 12)

UWAGI!
W przypadku uruchamiania aplikacji na jednym komputerze parametr -hostfile pomijamy.

W przypadku gdy nie jest możliwe zalogowanie się do drugiej maszyny pomimo prawidłowego pliku hostfile, należy sprawdzić, czy połączenie poprzez **ssh** między komputerami wymaga hasła.

Wtedy i tylko wtedy należy zgłosić tą kwestię prowadzącemu, i po jego akceptacji:
- Wygenerować klucz:

```
ssh-keygen -t rsa -b 4096
```

- Skopiować klucz do drugiego komputera:

```
ssh-copy-id mpiuser1@AdresIP:
```

5. Opis programów wykorzystywanych w scenariuszu

prime_mpi.c
Program służy do obliczania liczby liczb pierwszych w przedziale od n_lo do n_hi.
Zmienne, które można modyfikować:
(linijka 51) n_lo: dolny zakres poszukiwania liczb pierwszych
(linijka 52) n_hi: górny zakres poszukiwania liczb pierwszych
(linijka 53) n_factor:

quad_mpi.c
Program służy do całkowania numerycznego funkcji liniowej jednej zmiennej. Całkowana funkcja jest zdefiniowana w linijce 204.
Domyślnie program oblicza całkę następującej funkcji:

$$f(x) = \frac{50}{\pi * (2500 * x^2 + 1)}$$

Zmienne, które można modyfikować to:
(linijka 56) a: dolna granica zakresu całkowania
(linijka 57) b: górna granica zakresu całkowania
(linijka 58) n: ilość przedziałów, na które jest podzielony zakres całkowania

Scenariusz nr 1

Sprzęt:
Komputer PC
Procesor ……..
RAM ……..
OS ……..
Użytkownik: **mpiuser1** (hasła powyżej, w zależności od maszyny)

Oprogramowanie:
OpenMPI

Parametry ćwiczenia:

Liczba komputerów wykorzystanych w ćwiczeniu: ……

Zmiana parametru n (liczby zdań): od …… do ……. ze skokiem co ……..
od …… do ……. ze skokiem co …….. ……..
od …… do ……. ze skokiem co …….. ……..

Program wykorzystany do testów: ……

Modyfikacje kodu źródłowego programu:. ……..

Wykonanie ćwiczenia:

1. Zalogować się na konto **mpiuser1** na wszystkich komputerach wykorzystywanych w ćwiczeniu
2. Na komputerze głównym (na tym na którym będą uruchamiane obliczenia)
   - pobrać kod źródłowy wybranego programu ze strony http://heavy.metal.agh.edu.pl/ do katalogu **/tmp/**
   - wprowadzić odpowiednie modyfikacje kodu źródłowego programu
   - w katalogu /tmp/ skompilować kod programu
   - skopiować binarkę na pozostałe komputery biorące udział w obliczeniach (do katalogów /tmp/)
3. Na wszystkich biorących udział w obliczeniach maszynach uruchomić
   a) nowy terminal, i wykonać w nim polecenie `top -d 1`
   b) Monitor systemu
   w celu uzyskania informacji o liczbie przydzielonych zadań i obciążeniu rdzeni procesora
4. Przy wykonywaniu obliczeń na więcej niż 1 maszynie – na komputerze głównym, w katalogu **/tmp/** stworzyć plik hosts.txt z adresami IP komputerów wykorzystywanych w scenariuszu

5. uruchomić obliczenia przez wywołanie polecenia `mpirun` z odpowiednimi opcjami (przykładowo:) **`mpirun -hostfile hosts.txt -n 12 program.out`**

   Dla każdej konfiguracji pomiar wykonać 2-krotnie notując parametry w tabeli poniżej.
   Parametry można notować na kartce, lub przez wykonanie zrzutów ekranu.
   Przy wykonywaniu zrzutów ekranu należy uwzględnić to, że czas wykonania dostępny jest dopiero po zakończeniu testu, a obciążenie procesora w trakcie jego trwania. Stąd (jeśli wykonywane są zrzuty ekranu) najlepiej wykonać po jednym zrzucie na każdej maszynie w trakcie obliczeń (liczba przydzielonych zadań, obciążenie rdzeni) i dodatkowo jeden zrzut po zakończeniu testów – na maszynie na której uruchomiono test (całkowity czas obliczeń)

Pomocnicza tabela do notowania rezultatów testów

Liczba dostęp-nych hostów	Liczba zdań (n)	Liczba zadań przydzielonych do poszczegól. maszyn	Liczba zadań zakończonych przed końcem wykonania programu.	Obciążenie procesora i rdzeni	Całkowity czas wykonywania zadania	Uwagi

<u>Wyniki pomiarów:</u>

a) opracować statystycznie uzyskane wyniki
b) wykresy czasów wykonywania w zależności od liczby hostów, rdzeni i zadań
c) opracować wnioski szczegółowe i końcowe

<u>Do sprawozdania należy dołączyć scenariusz z uwagami oraz podpisane notatki!</u>