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

TODO:
https://pl.wikipedia.org/wiki/MPI
https://pl.wikipedia.org/wiki/Message_Passing_Interface
https://en.wikipedia.org/wiki/Message_Passing_Interface
https://pl.wikipedia.org/wiki/OpenMP
https://pl.wikipedia.org/wiki/MPICH
https://pl.wikipedia.org/wiki/Parallel_Virtual_Machine

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

ścieżka instalacji:
`/usr/lib64/openmpi/`

dla wygody można dodać do ścieżki systemowej komendą:
`export PATH=/usr/lib64/openmpi/bin:$PATH`

W katalogu openmpi sprawdzić, czy obecne są programy: mpic++ i mpicc. Jeśli ich brakuje instalujemy paczkę openmpi-devel przy pomocy polecenia (jako root):
`yum install openmpi-devel`

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

Jeśli podczas kompilacji pojawia się informacja o braku g++ należy doinstalować paczkę gcc-c++ przy pomocy polecenia (jako root):
`yum install gcc-c++`

Na każdej maszynie biorącej udział w obliczeniach w katalogu `/tmp/` muszą znaleźć się:
a. plik wykonywalny (binarka) – uzyskiwany po skompilowaniu kodu
b. plik z listą hostów (adresami IP) biorących udział w obliczeniach

można skopiować je poleceniem (w katalogu `/tmp/`):
`scp nazwa_pliku adresIP:/tmp/`

a) kompilacja programu w c++
`/usr/lib64/openmpi/bin/mpic++ -o matvec.out matvec_mpi.cpp`
LUB
`mpic++ -o program.out source.cpp (program w c++)`
`mpicc -o program.out source.c (program w c)`

gdzie:
`program.out` – nazwa wyjściowego pliku wykonywalnego (może być dowolna)
`source.cpp, source.c` – kod źródłowy program

b) na komputerach biorących udział w obliczeniach sprawdzamy czy usługa ssh jest uruchomiona oraz firewall wyłączony:

```
service sshd status
service firewalld status
```

c) jeśli jest taka potrzeba włączamy ssh i wyłączamy firewall:

```
service sshd status
systemctl stop firewalld
```

d) tworzymy plik 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`.
Na każdym komputerze biorącym udział w obliczeniach powinien znaleźć się identyczny plik zarówno pod względem zawartości, jak i nazwy.

e) uruchamianie program

```
mpirun -hostfile hosts.txt -n 24 --oversubscribe -bind-to core:overload-allowed matvec.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 może być dowolna
`--oversubscribe -bind-to core:overload-allowed` – umożliwienie podziału programu na ponadstandadową liczbę wątków
`program.out` – nazwa wyjściowego pliku do wykonywania
`-n 24` – liczba podzadań, na które należy podzielić zadanie główne

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

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 student

Oprogramowanie:
OpenMPI

Parametry ćwiczenia:

Ilość maszyn	Ilość proc/skok	Skok	Uwagi
			Każdy pomiar 3 razy, notować czasy obliczeń i węzły

Wykonanie ćwiczenia:

Przygotowanie:
1. zalogować się jako student na konsoli graficznej bądź tekstowej
2. wejść na stronę http://heavy.metal.agh.edu.pl/ gdzie dostępne są kody źródłowe programów

Wykonanie ćwiczenia:
1. ustalić z prowadzącym wersję scenariusza
    a. wersję kodu programu
    b. wersję sprzętową testów
    c. w zależności od ustalonej ilości maszyn testy wykonujemy najpierw na 1 maszynie, później na 2 itd..
2. na każdym komputerze wyznaczonym do testów, w katalogu `/tmp/` :
    a. skompilować program
    b. stworzyć plik hosts.txt z adresami IP komputerów wykorzystywanych w scenariuszu (na każdym komputerze zawartość pliku musi być identyczna!)
3. uruchomić Monitor Systemu w celu zanotowania liczby obciążonych rdzeni dla każdej z wykonywanych konfiguracji
4. uruchomić `mpirun` z odpowiednimi opcjami
5. dla każdej konfiguracji wykonywać 3 pomiary, notując czas wykonania, liczbę maszyn i liczbę procesów

Wyniki pomiarów:
a) opracować statystycznie uzyskane wyniki
b) wykresy czasów wykonywania w zależności od ilości hostów, i procesów
c) opracować wnioski szczegółowe i końcowe

Do sprawozdania należy dołączyć scenariusz z uwagami oraz podpisane notatki!