Lab 13 – Adrian Wanczewski

PVM (Parallel Virtual Machine) - zestaw narzędzi do tworzenia oprogramowania dla sieci równolegle połączonych komputerów. Został zaprojektowany i stworzony by umożliwić łączenie komputerów o różnych konfiguracjach sprzętowych w jeden równolegle działający komputer.
Jest narzędziem służącym głównie do pośrednictwa w wymianie informacji pomiędzy procesami uruchomionymi na oddzielnych maszynach oraz do zarządzania nimi za pośrednictwem konsoli pvm.

W dużym uproszczeniu PVM pozwala na "połączenie" większej ilości komputerów w jeden. Odbywa się to na zasadzie dołączania kolejnych hostów (komputerów), na których został zainstalowany i skonfigurowany pvm. Podłączanie hosta jest w uproszczeniu procedurą połączenia przez rsh, uruchomieniu demona pvm i dostarczenia mu informacji o tym, kto jest jego "rodzicem" oraz o parametrach istniejącej sieci.

W momencie, kiedy cała "maszyna wirtualna" już jest skonfigurowana, rola pvm sprowadza się do stanowienia pomostu wymiany informacji pomiędzy procesami, które się w pvm "zarejestrują", oraz umożliwienia administrowania stanem zarejestrowanych w pvm procesów.

Identyfikacja w PVM odbywa się przez oddzielne numery identyfikacyjne przydzielane procesom w momencie rejestrowania się w PVM. Numery te stanowią zupełną abstrakcję od systemu operacyjnego i sprzętu, na którym uruchomiony jest dany proces. Numer identyfikacyjny jest bardzo istotnym elementem PVM, ponieważ stanowi on niejako adres danego procesu i jest on niepowtarzalny w zakresie pojedynczej maszyny PVM. Adres ten jest wykorzystywany przy wszystkich procedurach dotyczących innych procesów - można przesłać informacje pod dany adres, można "zabić" zdalny adres i wykonać wiele innych czynności.

Można dostrzec więc pewne podobieństwa do MPI, ale pierwsza wersja PVM została napisana w ORNL w 1989 roku. Natomiast MPI zostało stworzone dopiero w maju 1994 r. więc całkiem możliwe ze PVM było w jakiejś części wzorem dla MPI.

Dzisiaj PVM jest nadal aktywnym projektem i nadal stosuje się go w programowaniu równoległym.

PRZYKŁAD:

Komunikacja punkt-punkt

pvm_send

```
int retval = pvm_send(int tid ,int msgtag); //
```

Parametry

- **tid** – identyfikator procesu, do którego ma zostać wysłany komunikat.
- **msgtag** – liczba całkowita definiowana przez użytkownika jako etykieta komunikatu (powinna być >=0).
- **retval** – kod statusu zwracany przez funkcję (retval < 0 oznacza błąd podczas wykonania operacji).

pvm_recv

```
int retval = pvm_recv(int tid ,int msgtag); //
```

Parametry

- **tid** – identyfikator procesu, od którego ma zostać odebrany komunikat.
- **msgtag** – liczba całkowita definiowana przez użytkownika jako etykieta komunikatu (powinna być >=0).
- **retval** – kod statusu zwracany przez funkcję (retval < 0 oznacza błąd podczas wykonania operacji).