WSTĘP DO PROGRAMOWANIA RÓWNOLEGŁEGO

1. KLASTER

Klaster jest zespołem pewnej liczby samodzielnych komputerów (**procesorów**). Każdy z nich posiada swoją pamięć wewnętrzną do jego "prywatnego" użytku. Procesory pogrupowane są w **węzły**. Wszystkie procesory są połączone między sobą siecią służącą do komunikacji między nimi. Podstawowa funkcja sieci, to przesyłanie danych i innych informacji. Procesory z jednego węzła łączy sieć działająca znacznie szybciej niż między procesorami z różnych węzłów. Można skomasować procesory z jednego węzła i korzystać z takiego węzła, tak jak z "grubego" procesora o większej pamięci wewnętrznej.

Każdy procesor może korzystać z dużej pamięci zewnętrznej (dyskowej). Pamięć taka może służyć na przykład do przechowywania wyników, danych, lub innych zasobów. Procesory mogą działać niezależnie i jednocześnie

IBM-Blue Gene/P NOTOS, który będzie nam służył, jest typowym klastrem. Posiada 1024 dostępne procesory, pogrupowane po 4 w 256 węzłach. Każdy procesor dysponuje pamięcią wewnętrzną o pojemności 1GB. Zegar generuje impulsy z częstotliwością 850MHz.

"Słowo" Notosa składa się z 32 bitów = 4 Bajty. Odpowiada to liczbom dziesiętnym siedmio-cyfrowym. Można liczyć wygodnie na podwójnych słowach ośmio-Bajtowych, co odpowiada liczeniu na liczbach 13 cyfrowych dziesiętnych. Na Notosie działa system operacyjny **Blue Gene/P linux**, zaś użytkowników obsługuje **system kolejkowy LoadLeveller**.

Jak posługiwać się Notosem. Każdy użytkownik ma swój kawałek maszyny, w którym trzyma swoje zasoby. Użytkownik otrzymuje ten swój "kawalek" Notosa w chwili zarejestrowania. Kazdy z użytkowników dostaje się do swojego "kawałka" poprzez sieć ze swojego komputera osobistego, lub z laboratorium, **najlepiej spod linux'a**, przy pomocy komendy **ssh** lub **ssh -Y**, gdy potrzebna będzie jakaś grafika (np. **gnuplot**). **Każde wejście wymaga logowania z użyciem swojego hasła**

Programy, które uruchomiamy, pierwotnie napisane w FORTRANIE lub C, muszą zostać skompilowane w katalogu użytkownika na jego "kawałku" Notosa.

Ponieważ na maszynie pracuje jednocześnie wielu użytkowników, programy użytkowników są ustawiane w kolejkę i kolejno wpuszczane do ma-

szyny przez system **LoadLeveller**. Wstawienie programu do kolejki polega na przesłaniu **SKRYPTU KOLEJKOWEGO** dla tego programu. Służy do tego polecenie wydane w linux'ie

llsubmit (NAZWASKRYPTU)

SKRYPT KOLEJKOWY to maleńki programik dla systemu linux, który opisany będzie niżej. Naprawdę, utworzenie takiego skryptu polega na wypełnieniu w małego, kilku-linijkowego formularza. Łatwo można śledzić los wstawionego do kolejki programu. Wynik działania programu pojawi się w katalogu użytkownika, z którego został wysłany skrypt kolejkowy.

Aby programować na Notosie (lub innym dostępnym klastrze) nie jest potrzebna żadna specjalna wersja C lub FORTRANU. Najlepiej posługieać sie najprostszą wersją podstawową. Obsługę wieloprocesorowości zapewnia **system podprogramów MPI: Message Passing Interface.** Ten system zawiera wersję dostosowaną do użycia w FORTRANIE i w C. Dołączamy do swoich zasobów **system MPI**, używając po każdym logowaniu komendy

module load mpi_default.

Wywołanie podprogramu z systemu **MPI** odbywa się w sposób standardowy, tak jak w przypadku podprogramów napisanych przez nas, dołączanych do programu przez nas napisanego. Na przykład, w przypadku FORTRANU wygląda to tak

call MPI_Bsend(....)

Wszelkie szczegóły dotyczące programowania na Notosie, obsługi jego wieloprocesorowości, wykorzystywania systemu **MPI** zostaną wyjaśnione w dalszych częściach tego skryptu (nie kolejkowego).

KIEDY WARTO LICZYĆ NA WIELU PROCESORACH

- Zagadnienia o wyraźnej strukturze sekwencyjnej często jest trudno zrównoleglić w sposób zadawalajcy. Takie zagadnienia można graficznie przedstawić tak:

A==>B==>C==>D==>E==>F==>G

gdzie **A, B, ... ,G** oznaczają częsci zagadnienia, które da się zaprogramować jako pewne niezależne całości, zaś strzałki oznaczają, że każda następna część zadania potrzebuje do działania wszystkich wyników działania części poprzedniej. Jednak czasami udaje się zrównoleglić takie zadanie, wykorzystując pewne indywidualne cechy zagadnienia.

- Nie jest elegancko liczyć na klastrze wiele wariantów jednego zadania, zaprogramowanego na jeden procesor, ładując każdy wariant w osobny procesor.

- Podstawowe korzyści liczenia na klastrze, to:

  1. szybkość
  2. możliwość rozwiązywania skomplikowanych numerycznie zadań wymagających wielkiej liczby danych lub/i produkujących wielkie ilości wyników

  Rodzajami zagadnień w których wykorzystywanie wieloprocesorowości zdecydowanie pomaga są rozmaite symulacje procesów technologicznych, procesów zachodzących w przyrodzie itp. (np. symulowanie biegnącego procesu technologicznego, w celu przewidzenia ewentualnej interwencji (**trzeba wtedy wyprzedzić w czasie biegnący już proces**); prognoza pogody potrzebna na określoną godzinę; przewidywanie trzęsień ziemi, powodzi i innych klęsk żywiołowych i wiele innych.)

  Także opracowywanie nowych aplikacji potrzebnych na przykład w badaniach naukowych lub w produkcji, tworzenie nowych wyspecjalizowanych podprogramów, często powinno być nastawione na stosowanie w systemie wieloprocesorowym.

TROCHĘ WIĘCEJ SZCZEGÓŁÓW

WSPÓŁPRACA Z LOADLEVELLEREM. Dla użytkowników klastra najważniejsze funkcje **LoadLeveller**'a to wstawianie programu do kolejki, usuwanie go z kolejki i informowanie o statusie programu w kolejce.

SKRYPT KOLEJKOWY. Oto formularz który trzeba przepisać do katalogu, w którym będziemy pracować, wypełnic, nazwać i wstawić do kolejki używając komendy **llsubmit ...**.

 # @ job_name = [NAZWA SKRYPTU]
 # @ account_no = [NUMER GRANTU]
 # @ class = [KLASA]
 # @ error = [NAZWA SKRYPTU].err
 # @ output = [NAZWA SKRYPTU].out
 # @ environment = COPY_ALL
 # @ wall_clock_limit = [hh : mm : ss]
 # @ notification = error
 # @ notify_user = [email address]
 # @ job_type = bluegene
 # @ bg_size = [LICZBA WEZLOW]
 # @ queue
mpirun -exe /home/users/U.../[K]/[P] -mode VN -np [L] {>[PK]}

- Kwadratowymi nawiasami zaznaczone są miejsca do wypełnienia
- **NUMER GRANTU i KLASA** to otrzyma każdy użytkownik podczas rejestracji
- **wall_clock_limit** zarezerwowany czas pracy maszyny. Im krótszy czas zamawiamy, tym prędzej program wejdzie do maszyny. Dla prostych testów wystarczy 5 - 10 minut.
- **LICZBA WĘZŁÓW i L; L**-to liczba procesorów. Pamiętamy, że Notos ma w jednym węźle 4 procesory, **U** - nazwa katalogu użytkownika, nadawana przy rejestracji, **K** nadana przez użytkownika nazwa katalogu, w którym jest program; **P** nazwa programu po kompilacji.

  Kompilator umieszcza skompilowany program w pliku **a.out**. Ten plik trzeba przepisać nadając mu nazwę, którą mamy wpisać jako **P**.
- **-mode VN** oznacza, że bedziemy pracować w konfiguracji **4 procesory w węźle**, to jest bez "grubych" procesorów.
- Część ostatniego wiersza w nawiasach {} jest opcjonalna: jeśli jej użyjemy, to do pliku **PK** w katalogu **K** będą wpisane informacje z tekstu

programu. Na przykład w FORTRANIE, umieszczenie w tekscie programu wiersza **write(*,*)'A=', a**, gdzie a jest zmienną, spowoduje wpisanie do pliku PK tekstu **A=wartość(a)**. Ta opcja bywa bardzo przydatna podczas uruchomiania nowego programu.

- Natychmiast po wstawieniu skryptu do kolejki (**llsubmit ....**), system przesyła komunikat z którego dowiadujemy się czy program został przyjęty. Jeśli tak, to otrzymamy przydzielony mu **numer zadania**, który warto zapamiętać.

USUWANIE PROGRAMU Z KOLEJKI. Jeśli program się "zapętlił" działa, ale nie daje znaków życia i nie chce sie zatrzymać, to pewnie należy wyciągnąc go z kolejki. Nie należy zwlekać zbyt długo z tą przykrą operacją, aby nie zajmować niepotrzebnie miejsca i czasu innym użytkownikom. Do tego służy komenda

llcancel numer zadania

System zawiadomi o wycofaniu zadania.

PYTANIE O STATUS ZADANIA W KOLEJCE. To pytanie można zadać na dwa sposoby:

- Pierwszy sposób

  llq

  System wypisze nam tablicę zawierającą wszystkie zadania z kolejki i ich aktualny status

- Drugi sposób

  llq -u nazwa użytkownika

  System poda inforację o ststusie wszystkich programów wstawionych do kolejki przez tego użytkownika

UWAGA! Obecnie liczba zadań wstawionych do kolejki przez jednego użytkownika nie może przekraczać liczby 10.

PRZYSTĘPUJEMY DO PRACY NA NOTOSIE

1. Ponieważ będziemy 'skazani' na pracę **'pod linuxem'**, być może będziemy musieli poświęcić trochę czasu na przypomnienie sobie kilku podstawowych poleceń tego systemu operacyjnego.

2. Jeśli to możliwe uruchamiamy na terminalu **linux**. Można również pracować z mniej wygodnego terminala 'windowsowego'. W każdym razie po połączeniu się z NOTOSEM i tak będziemy musieli pracować w środowisku **linuxowym**. Do połączenia z Notosem musimy mieć zainstalowane dwa programiki:

   - **ssh** który umożliwi nam połączenie z NOTOSEM i prace na tej maszynie
   - **sftp** który umożliwi nam przesyłanie (w obie strony) danych miedzy naszym terminalem a maszyną na której pracujemy. Warto wiedzieć, że system **linux** jest standardowo wyposarzony w te programy. W obu przypadkach **ssh** i **sftp** musmy najpierw połączyć się z maszyną **delta**, a następnie z **delty** łączymy się z **notosem**. Są potrzebne **dwa logowania** z użyciem tego samego hasła.

3. Możemy program napisać zarówno na naszym terminalu, jak i na notosie. W pierwszym przypadku trzeba go będzie przesłać przy pomocy **sftp** poprzez **deltę** do katalogu w którym zamierzamy pracować na **notosie**. Oczywiście trzeba najpierw ten katalog utworzyć. Trzeba go również wyposażyć środowisko właściwe do pracy na wielu procesorach, przy pomocy polecenia

   module load mpi_default

4. Następna czynność, to kompilacja programu, i jeśli nie wykryte zostały błędy formalne, to wstawienie programu do kolejki.