Введение в параллельные вычисления. Технология программирования MPI

(день первый)

Антонов Александр Сергеевич, к. ф.-м.н., н.с. лаборатории Параллельных информационных технологий НИВЦ МГУ

Координаты для связи:

- E-mail: asa@parallel.ru, parallel@parallel.ru
- Тел: 939-23-47
- Web: http://parallel.ru

План занятий:

- 1-й день: введение, коротко об операционной системе UNIX, практические сведения, параллелизм и способы его использования
- 2-7-й дни: технология MPI
- 8-й день: обсуждение результатов, подведение итогов, ответы на вопросы

UNIX:

UNIX – это многозадачная, многопользовательская система, обладающая широкими возможностями. Ее реализации существуют практически на всех распространенных компьютерных платформах.

LINUX – один из наиболее известных свободно распространяемых диалектов UNIX.

UNIX:

Все объекты в UNIX делятся на два типа:

файлы и *процессы*.

Все данные хранятся в файлах, доступ к периферийным устройствам осуществляется через специальные файлы.

Вся функциональность операционной системы определяется выполнением различных процессов.

UNIX:

Важнейшим пользовательским процессом является основной *командный интерпретатор (login shell)*. login, password

`passwd` – смена пароля

пароль должен хорошо запоминаться и быть трудным для подбора!

`exit` – выход из системы

`man` – получение справки о командах

`man man`

UNIX:

Идентификатор пользователя (**UID**), идентификаторы групп (**GID**).

Принадлежность к группе определяет дополнительные права пользователей.

Информация о пользователях и группах обычно хранится в системных файлах **`/etc/passwd`**, **`/etc/shadow`** и **`/etc/group`**

UNIX:

Файловая система, каталоги.

/ - корневой каталог

/home/asa/myfile.txt

. – текущий каталог

. . – каталог на единицу более высокого уровня

С каждым пользователем ассоциируется его *домашний каталог.*

UNIX:

Атрибуты файлов. **`ls -l`**

```
    1          2   3     4      5     6       7      8
-rwxr-xr--  1 asa group 3422 Feb 28 13:30 test
```

`-` – обычный файл; **`d`** – каталог, **`l`** – ссылка и др.

Права доступа к файлу:

`-` – отсутствие права доступа, **`r`** – право на чтение, **`w`** – право на запись или удаление, **`x`** – право на выполнение файла.

Владелец-пользователь, владелец-группа и все остальные пользователи

UNIX:

Смена прав доступа к файлу:

`chmod [u g o a][+ - =][r w x] file1`…

`u` – смена права доступа для пользователя,

`g` – для группы, **`o`** – для других пользователей,

`a` – для всех трех категорий.

`+` – добавление соответствующего права,

`-` – удаление, а **`=`** – присвоение

`chmod g+w test`

`chown` и **`chgrp`** – смена владельца-пользователя и владельца-группы файла

UNIX:

- **cd [dir]** – переход в каталог **dir**

  Если каталог не указан, то переход осуществляется в домашний каталог пользователя
- **cp file1 file2** – копирование файла
- **mv file1 file2** – перемещение (изменение имени) файла
- **rm file1…** – удаление файлов
- **rmdir dir1…** – удаление каталогов
- **mkdir dir1…** – создание каталога

UNIX:

- **pwd** – вывести имя текущего каталога
- **cat file, more file, less file** – утилиты просмотра содержимого файла
- **find dir** – поиск в файловой системе, начиная с каталога **dir**
- **grep <рег_выражение> file1…** – поиск в файлах вхождений регулярного выражения **рег_выражение**
- …

UNIX:

Процесс - программа в стадии ее выполнения.

ps – список выполняющихся процессов

Уникальный идентификатор процесса (**PID**).

Сигналы.

Завершить выполнение процесса:

kill –9 PID

Список процессов, занимающих наибольшее количество процессорного времени или системных ресурсов:

top

UNIX:

Потоки ввода/вывода: стандартный ввод, стандартный вывод и стандартный вывод ошибок. Для перенаправления стандартного ввода можно использовать символ **<**, для стандартного вывода – **>** или **>>** (с добавлением), для потока ошибок – **2>**

```
program > file.log
```

Конвейер команд:

```
program1 | program2 | program3…
```

Фоновый режим:

```
program &
```

UNIX:

who – список пользователей, работающих в данный момент в системе

uname – некоторые сведения о системе

Редактирование файлов: **vi, joe** и др., встроенный редактор файлового менеджера Midnight Commander (**mc**), удаленное редактирование.

Компиляторы с языка Си **cc** (**CC** для Си++), компилятор с языка Фортран – **f77** (**f90** для Фортрана 90).

time program – время работы программы

Параллелизм:

Конвейерность и *параллельность*.

Параллельная обработка. Одна операция - за единицу времени, то 1000 - за тысячу единиц. Пять устройств 1000 операций выполнит за 200 единиц времени. N устройств ту же работу выполнит примерно за 1000/N единиц времени.

Параллелизм:

Конвейерная обработка. Операция разбивается на ряд подопераций, выполняемых последовательно и независимо. Пусть 5 микроопераций, каждая из которых выполняется за единицу времени. Последовательное устройство 100 пар аргументов обработает за 500 единиц. Конвейерное устройство: первый результат через 5 единиц времени, каждый следующий – через одну единицу после предыдущего, а весь набор из ста пар будет обработан за 5+99=104 единицы времени.

Параллелизм:

Необходимо выделить группы операций, которые могут вычисляться одновременно и независимо. Возможность этого определяется наличием или отсутствием в программе истинных информационных зависимостей.

Две операции программы называются *информационно зависимыми*, если результат выполнения одной операции используется в качестве аргумента в другой.

Параллелизм:

Крупноблочное распараллеливание:

```
   if (MyProc = 0) then
C операции, выполняемые 0-ым процессором
   endif
   ...
   if (MyProc = K) then
C операции, выполняемые K-ым процессором
   endif
```

Параллелизм:

Наибольший ресурс параллелизма в программах сосредоточен в циклах!

Распределение итераций циклов:

```
   do i = 1, N
     if (i ~ MyProc) then
C операции i-й итерации для
C для выполнения процессором MyProc
     endif
   enddo
```

Параллелизм:

Примеры способов распределения итераций циклов:

- *Блочное распределение* – по **⌈N/P⌉** итераций.

- *Блочно-циклическое распределение* – размер блока меньше, распределение продолжается циклически.

- *Циклическое распределение* – циклически по одной итерации.

Параллелизм:

Рассмотрим простейший цикл:

```
do i = 1, N
  a(i) = a(i) + b(i)
enddo
```

Параллелизм:

Блочное распределение:

```
C размер блока итераций
  k = (N-1)/P + 1
C начало блока итераций
C процессора MyProc
  ibeg = MyProc * k + 1
C конец блока итераций
C процессора MyProc
  iend = (MyProc + 1) * k
```

Параллелизм:

```
C если не досталось итераций
  if (ibeg .gt. N) then
    iend = ibeg - 1
  else
C если досталось меньше итераций
    if (iend .gt. N) iend = N
  endif
  do i = ibeg, iend
    a(i) = a(i) + b(i)
  enddo
```

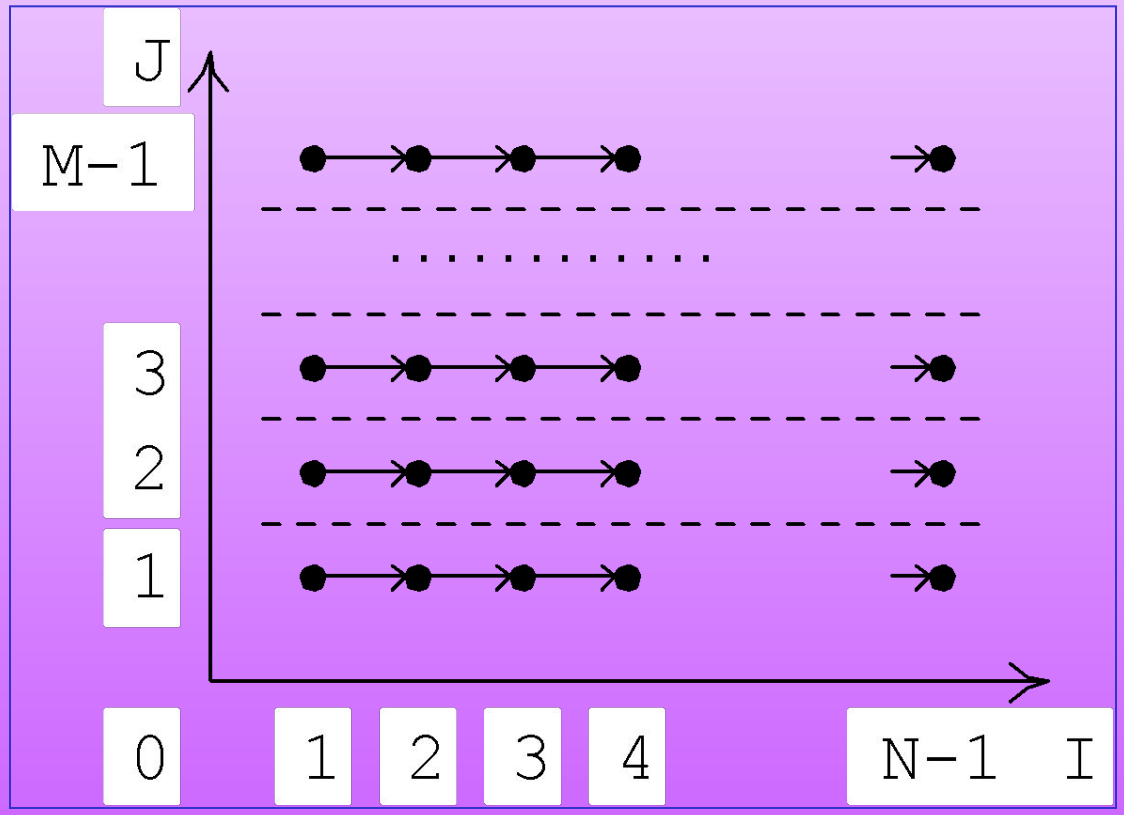
Параллелизм:

Циклическое распределение:

```
do i = MyProc+1, N, P
  a(i) = a(i) + b(i)
enddo
```

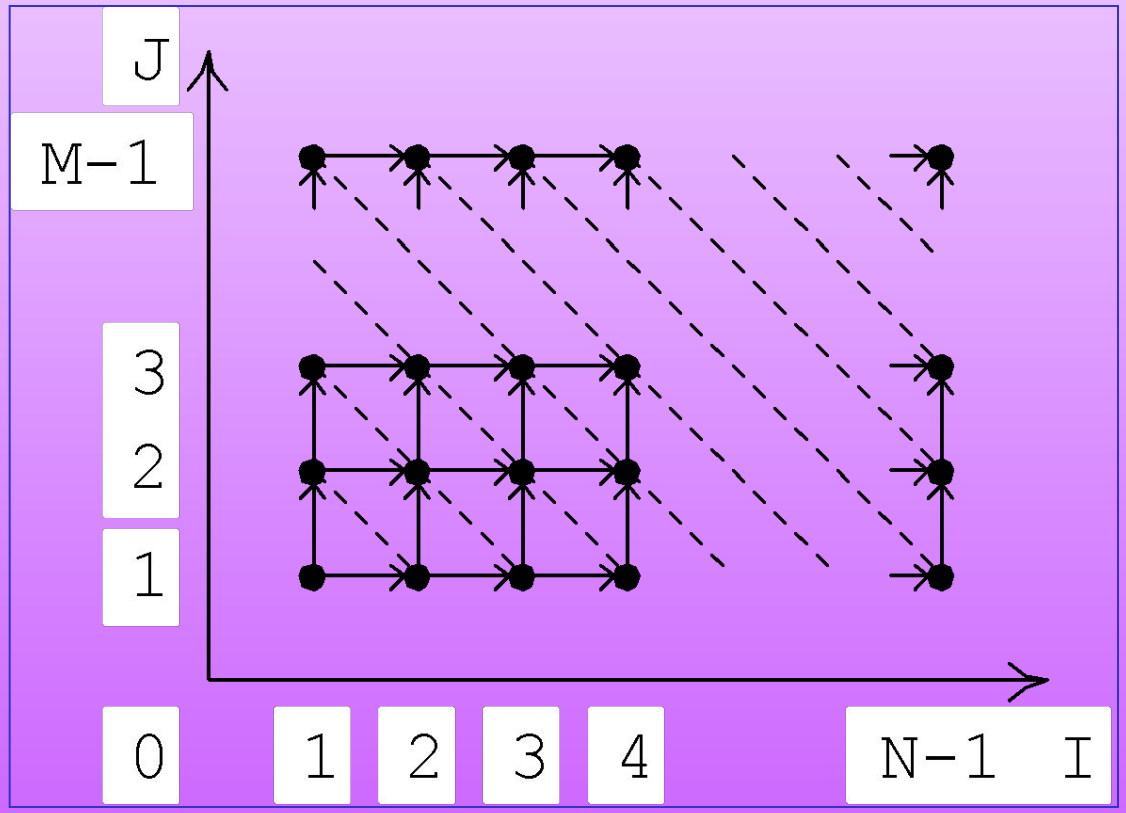
Параллелизм:

```
   do 1 i = 1, N-1
      do 1 j = 1, M-1
1     a(i,j) = a(i-1,j) + a(i,j)
```



Параллелизм:

```
   do 1 i = 1, N-1
      do 1 j = 1, M-1
1     a(i,j) = a(i-1,j) + a(i,j-1)
```



Параллелизм:

Цели распараллеливания:

- *равномерная загрузка процессоров*
- *минимизация количества и объема необходимых пересылок данных*

Пересылка данных требуется, если есть информационная зависимость между операциями, которые при выбранной схеме распределения попадают на разные процессоры.

Параллелизм:

Закон Амдала:

Пусть **f** – доля последовательных операций,

0 ≤ f ≤ 1,

1-f – доля параллельных операций,

S – ускорение, **p** – число процессоров

$$S \le \frac{1}{f + \frac{1-f}{p}}$$