Programming Assignment #2 -- Multi-Process Matrix Multiplication using Shared Memory

Introduction to Operating Systems

Prof. Li-Pin Chang

CS@NYCU

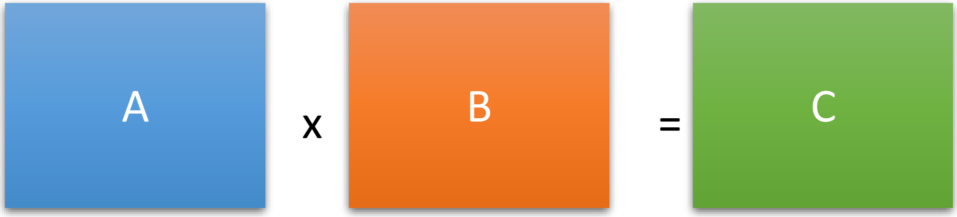
Overview

- Matrix multiplication using multiple processes
  - Basic parallel processing
  - Will be faster on multicore machines
- Input: the dimension of two square matrices A & B
  - E.g., 100 → A, B, and C are 100x100 square matrices
- Output: an execution time and a checksum

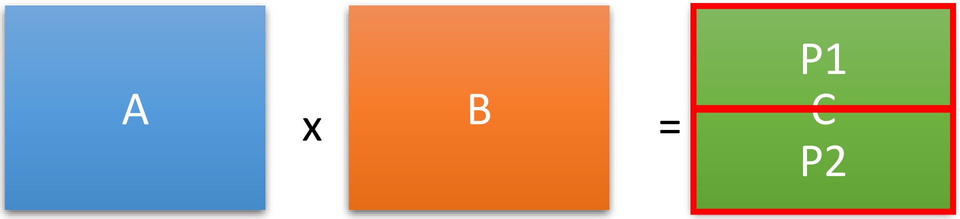
A x B = C

Task Partition

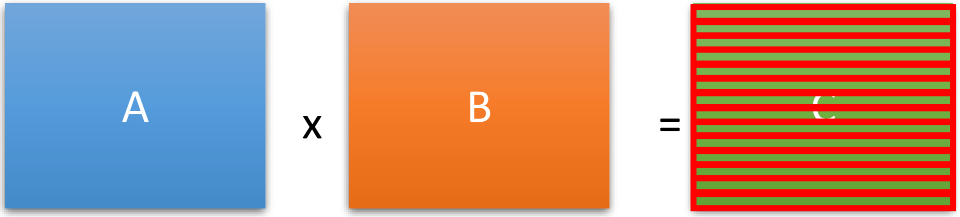
- 1-process matrix multiplication



- 2-process matrix multiplication

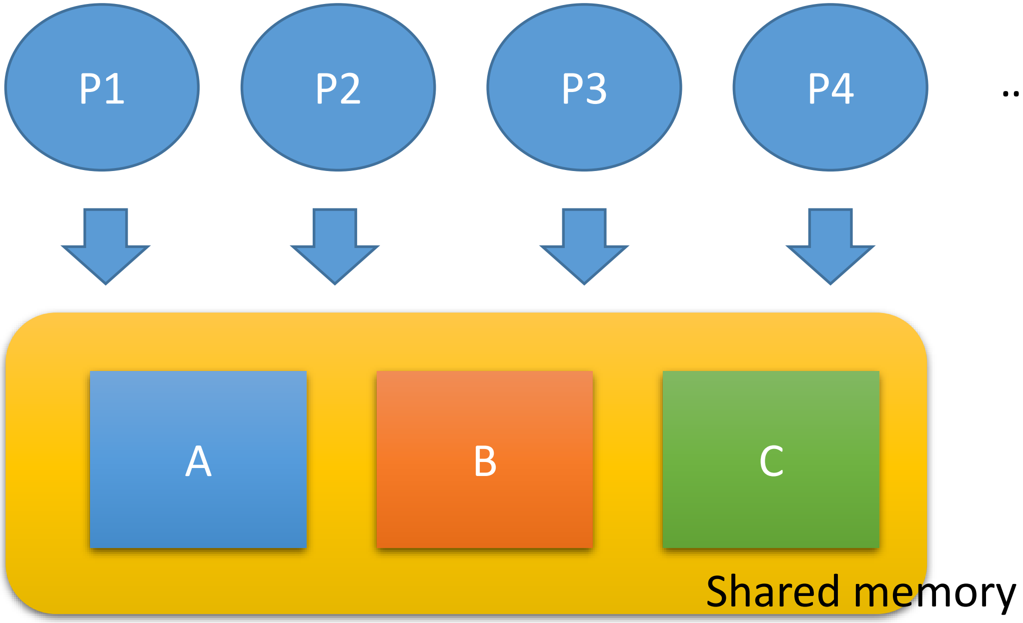


- 16-process matrix multiplication



Shared Memory

- Matrices A, B, and C are stored in a shared memory
- No locking/synchronization is required since multiplication on sub-metrices are mutually independent



Header Files

- unistd.h
- sys/ipc.h
- sys/shm.h
- sys/wait.h

- sys/time.h

APIs

- shmget() – create a block of shared memory
- shmat() – attach shared memory to the current process’s address space
- shmdt() – detach shared memory from the current process’s address space
- shmctl() – control shared memory

- gettimeofday()

shmget()

- **`int shmget(key_t`** *`key`*`,` **`size_t`** *`size`*`,` **`int`** *`shmflg`*`);`

- create a block of shared memory
- Return the id of the request shm of size equals to the value of *`size`*
- *`key`* : 0/IPC_PRIVATE for new allocate shm
- *`size`* : size in bytes
- *`shmflg`* : IPC_CREAT | mode_flags(9 bits)

  e.g. IPC_CREAT | 0600 for read-write shm

shmat()

- `void *shmat(int shmid, const void *shmaddr, int shmflg);`

- Attach shared memory to the current process's address space
- Return the address of the attached shared memory segment identified by *`shmid`*
- *`shmaddr`* : NULL for system to choose suitable address

- *`shmflg`* : 0 for read/write, **`SHM_RDONLY`** for read only

shmdt()

- **`int shmdt(const void *`*`shmaddr`*`);`**

- Detach the shared memory segment located at the address *`shmaddr`* from the address space of the calling process
- *`shmaddr`* must equal to the value returned by **`shmat()`**

shmctl()

- **int shmctl(int** *shmid*, **int** *cmd*, **struct shmid_ds ****buf***);**
- control shared memory
- Perform control operation
  - **IPC_STAT**
  - **IPC_SET**
  - **IPC_RMID**
  - …
- IPC_RMID: Marking a shared memory to be deleted. The share memory will be destroyed on when the last process detach the memory from its address space. Must be called by the owner or creator of the shared memory.
- [parent]shmget() → shmat()*N → shmdt()*N → [parent]shmctl(RMID)

gettimeofday()

```
struct timeval start, end;
gettimeofday(&start, 0);
//do something
gettimeofday(&end, 0);
int sec = end.tv_sec – start.tv_sec;
int usec = end.tv_usec – start.tv_usec;

printf(“elapsed %f sec”, sec+(usec/1000000.0));
```

Matrix Initial Values

- Matrix elements in A and B are initialized as follows (for example, an 8x8 matrix)

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55
56	57	58	59	60	61	62	63

Checksum of this matrix: 2016

Requirements

- Create worker processes using fork()
  - Wait until all worker processes have exit()'ed
- About matrices A, B, and C
  - A and B can be inside or outside of the shared memory
  - A and B can be the same or separate matrices
  - C must be in the shared memory
- Print the elapsed time and the matrix checksum
  - 16 cases, degree of process parallelism increases from 1 to 16
  - The final checksum must be correct
  - Matrix elements and the checksum are all 32-bit unsigned integers
  - Don't worry, just let the checksum integer overflow

Requirements – cont'd

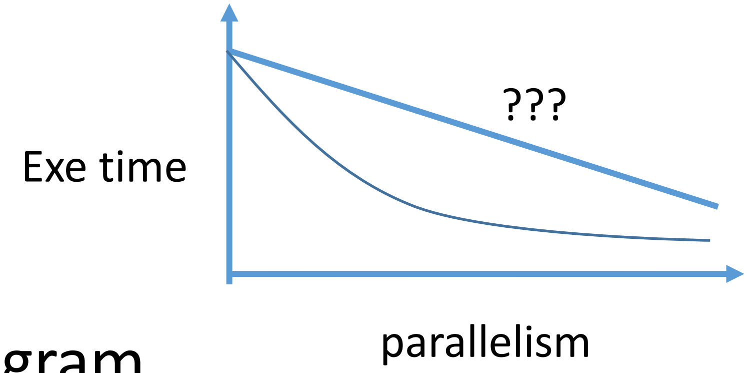
- TAs will test your program using any matrix dimension between 100*100 and 800*800
- [important] TAs will test your program on a multicore machine
  - Suppose that the test platform has 4 cores, then
  - your 2-process and 3-process versions must be noticeably faster than your 1-process version
  - But the speedup quickly saturates as process # increases
  - If not, there *must* be something wrong with your program
- Violating any of these requirements (and those in the prior page) will incur a score penalty

Output

```
Input the matrix dimension: 800↵

Multiplying matrices using 1 process
Elapsed time: 5.814723 sec, Checksum: 561324032
Multiplying matrices using 2 processes
Elapsed time: 3.10231 sec, Checksum: 561324032
Multiplying matrices using 3 processes
Elapsed time: 2.927338 sec, Checksum: 561324032
……
Multiplying matrices using 16 processes
Elapsed time: x.xxxxx sec, Checksum: 561324032
```

Amdahl’s Law



- P: parallelizable portion of a program
- S: non-parallelizable portion of a program
- [-----------P(16)-----------][-----S(8)-----]

- Deg. Of parallelism=2
- [-----P(8)-----][-----S(8)-----] : speedup=1.5
- Deg. Of parallelism=16
- [P(1)][-----S(8)-----] : speedup=2.66
- Deg. Of parallelism → infinity
- [→0][-----S(8)-----] : speedup=3

API Reference

- http://blog.csdn.net/guoping16/article/details/6584058
- http://man7.org/linux/man-pages/man2/shmget.2.html
- http://man7.org/linux/man-pages/man2/shmat.2.html

Testing OS Environment

- Ubuntu 22.04
- Install as a VM or on a physical machine

Header of your .c or .cpp

```
/*
Student No.: 31415926
Student Name: John Doe
Email: xxx@yyy.zzz
SE tag: xnxcxtxuxoxsx
Statement: I am fully aware that this program is not
supposed to be posted to a public server, such as a
public GitHub repository or a public web page.
*/
```