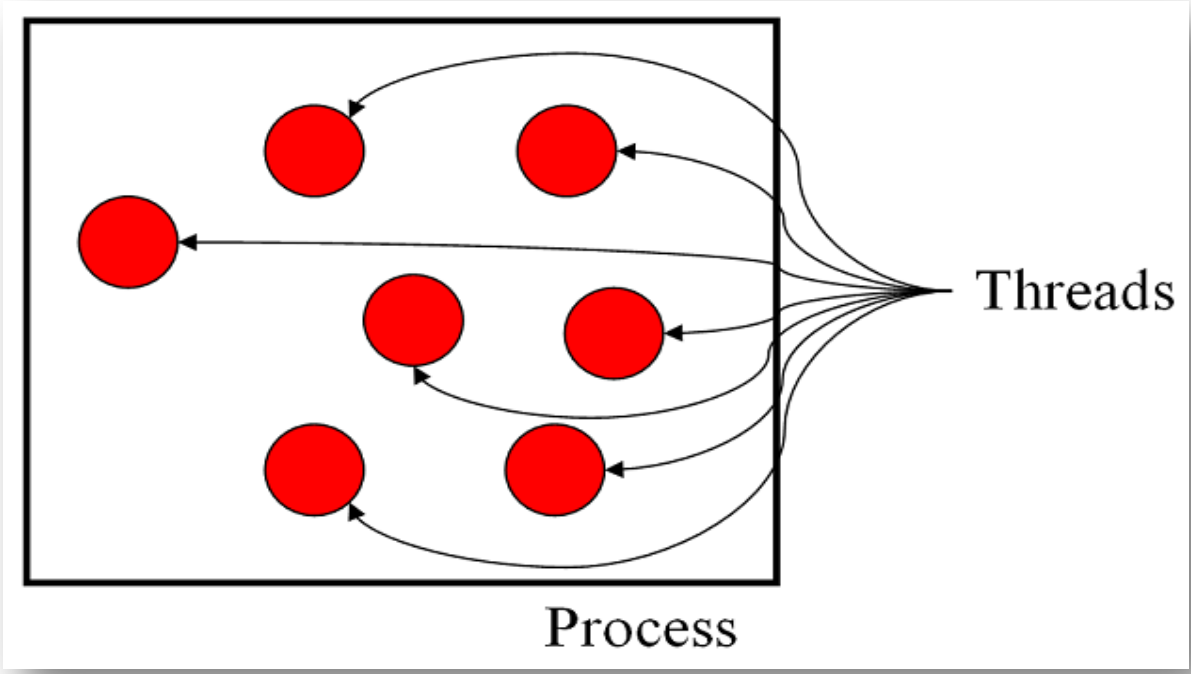
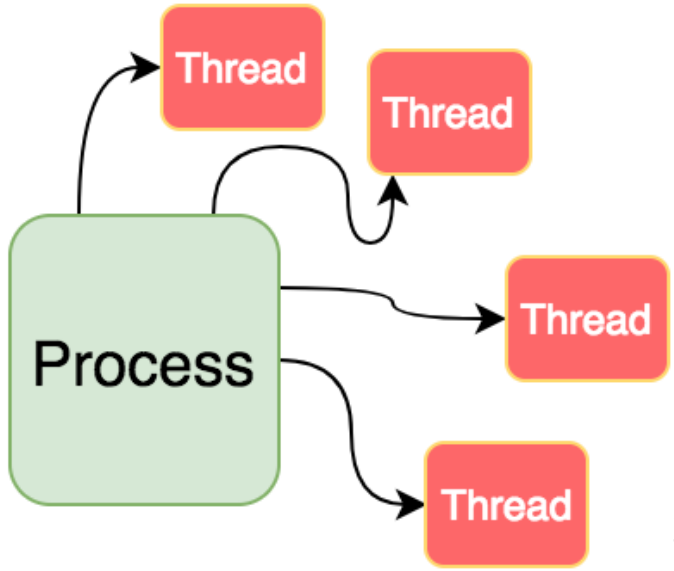
Bài 3
Threads & Concurrency

Khoa CNTT – ĐH Sài Gòn

Thread là gì?

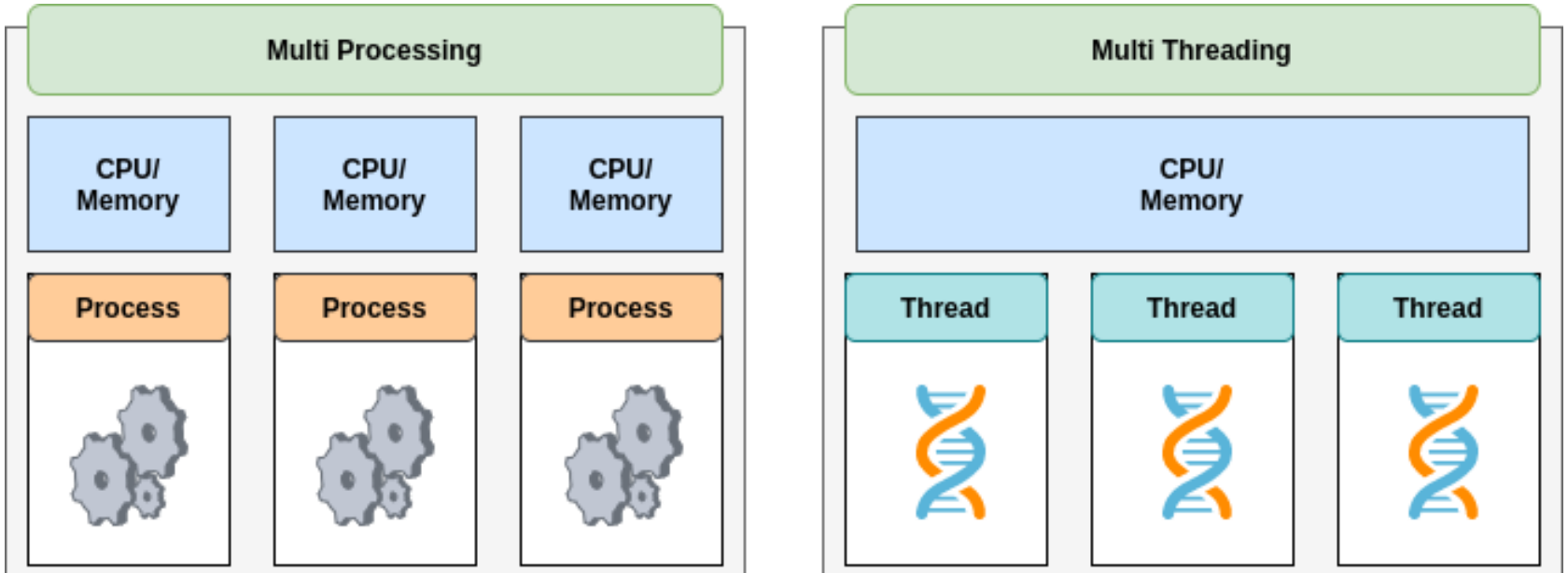


- Thread (còn gọi là “luồng”) là:
  - Một phần của tiến trình, là đơn vị cơ bản để phân bổ thời gian sử dụng CPU.
  - Một tiến trình nhẹ “lightweight process”
- Một process thường chứa 1 hoặc nhiều thread, mỗi thread đóng vai trò như 1 process con. Tại 1 thời điểm, chỉ có 1 thread có thể thực thi / 1 CPU core.
- Thread có thể thực thi các công việc tương tự process.



Vì sao cần thread

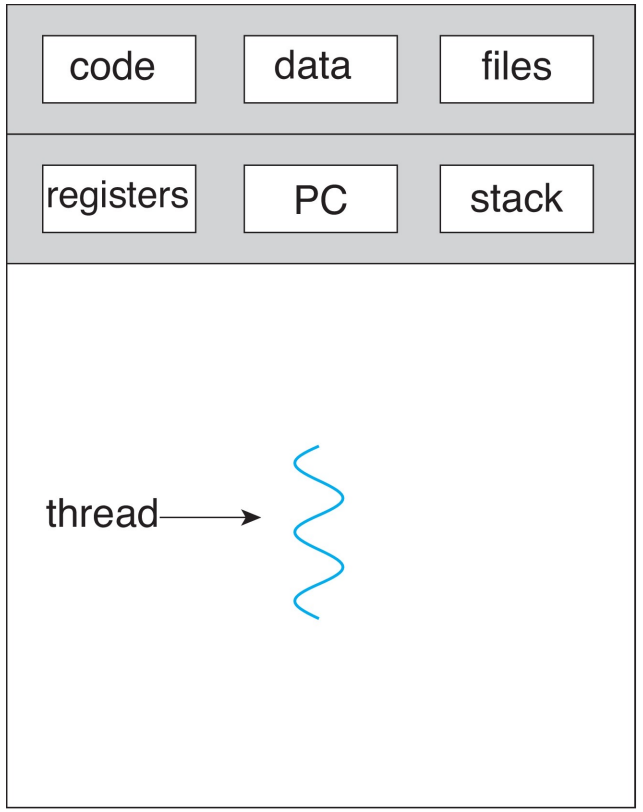
- Tạo và hủy tiến trình tốn nhiều chi phí (context switch)
- Các tiến trình độc lập tài nguyên.
- Giao tiếp giữa các tiến trình sử dụng IPC phức tạp.
- Multitasking bằng process không hiệu quả bằng thread:
  - Thời gian chuyển đổi giữa các thread ngắn.
  - Chia sẻ bộ nhớ giữa các thread trong 1 process.



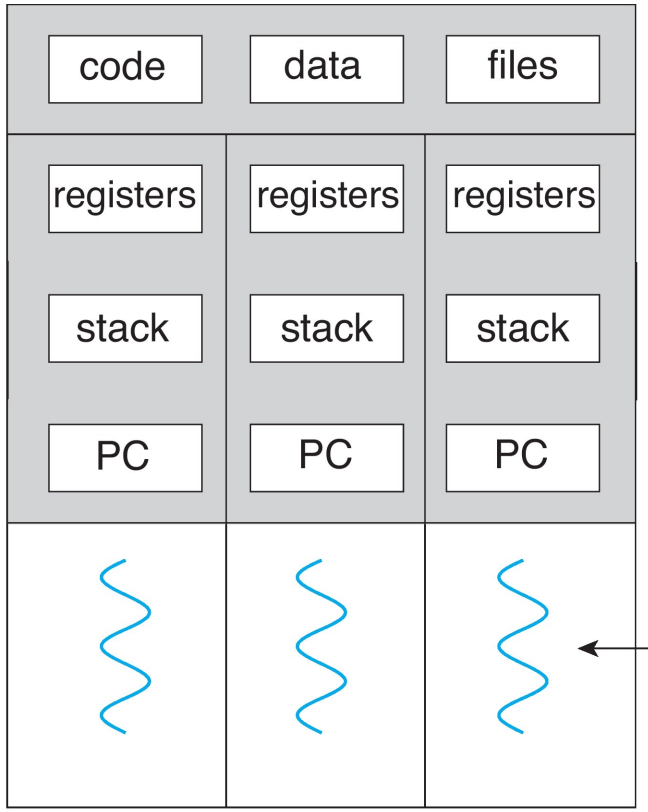
Vì sao cần thread

- Các thread trong một process chia sẻ một số loại tài nguyên, đặc biệt là vùng nhớ

→ giao tiếp giữa các thread trong một process dễ dàng hơn giữa các process



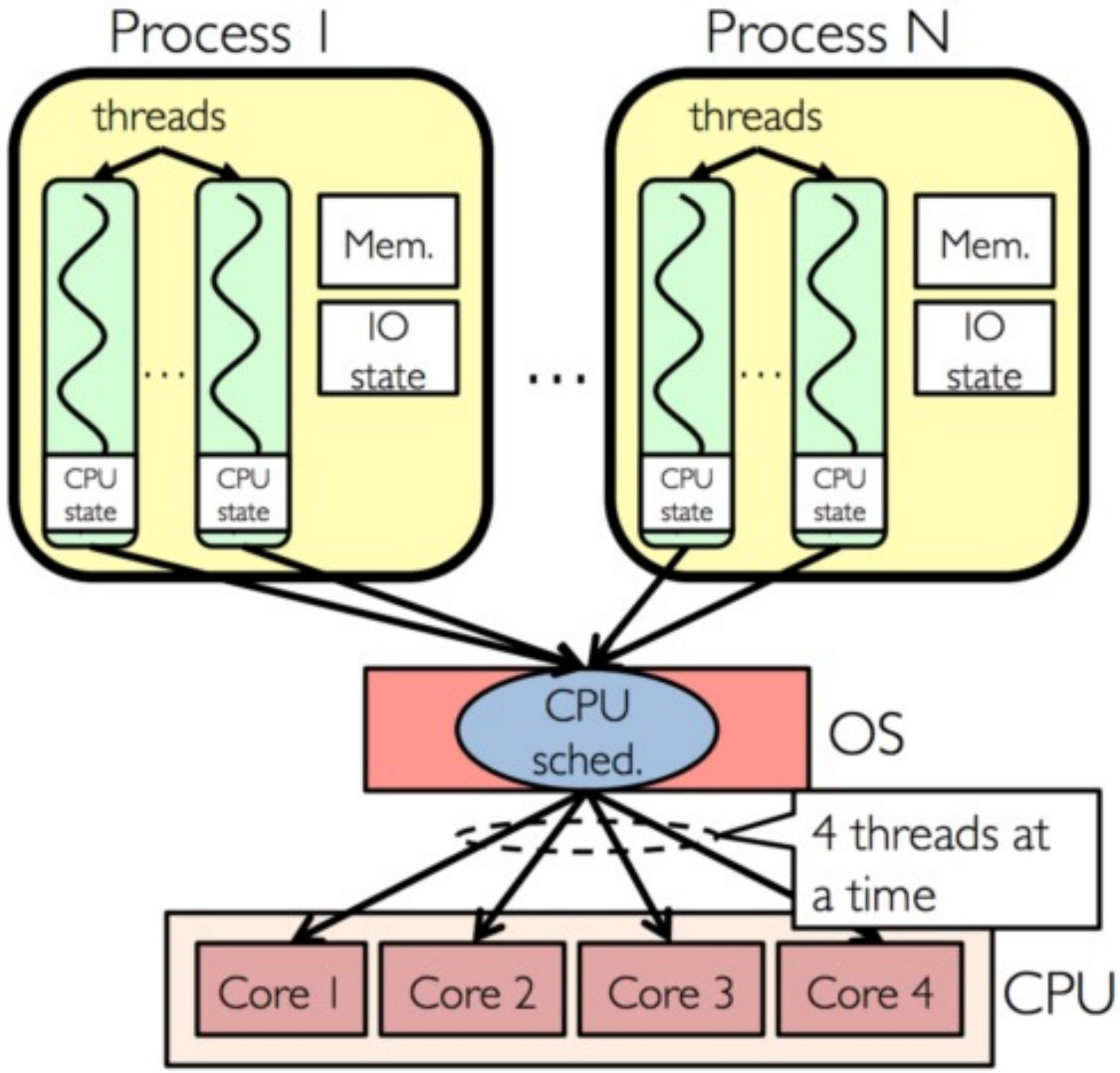
single-threaded process



multithreaded process

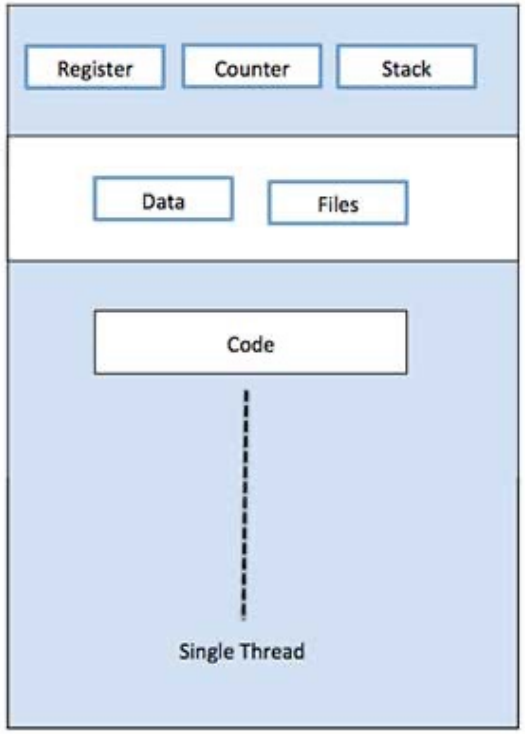
Ưu điểm của thread

- Tăng khả năng đáp ứng/phản hồi: process được chia thành nhiều thread nhỏ, mỗi khi 1 thread hoàn tất công việc thì nó sẽ trả về kết quả ngay lập tức.
- Chuyển ngữ cảnh (context switch) nhanh hơn do tốn ít chi phí hơn.
- Sử dụng hiệu quả hệ thống đa vi xử lý: phân các threads → các CPU cores

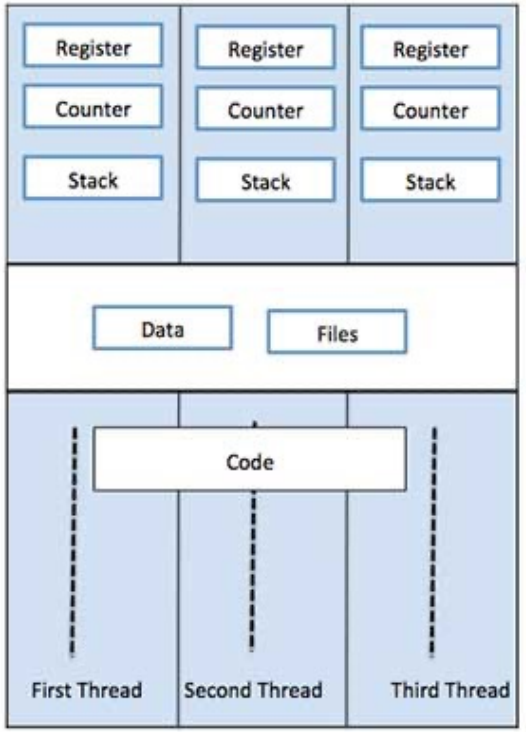


Ưu điểm của thread

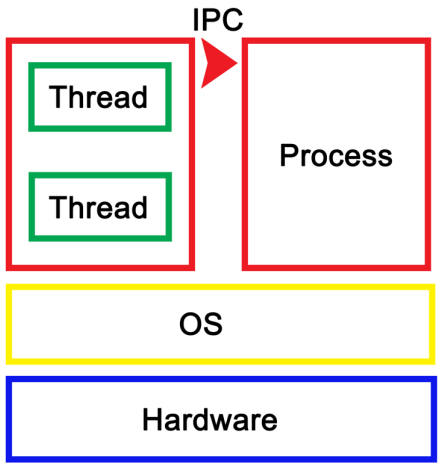
- Chia sẻ tài nguyên: các thread có thể cùng chia sẻ code section, data section, open files,… (stack, PC và registers không thể chia sẻ).
- Giao tiếp dễ & hiệu quả: do chia sẻ không gian nhớ chung.
- Tăng thông lượng (throughput) hệ thống: chia nhiều thread giúp công việc thực thi nhanh hơn.



Single Process P with single thread



Single Process P with three threads

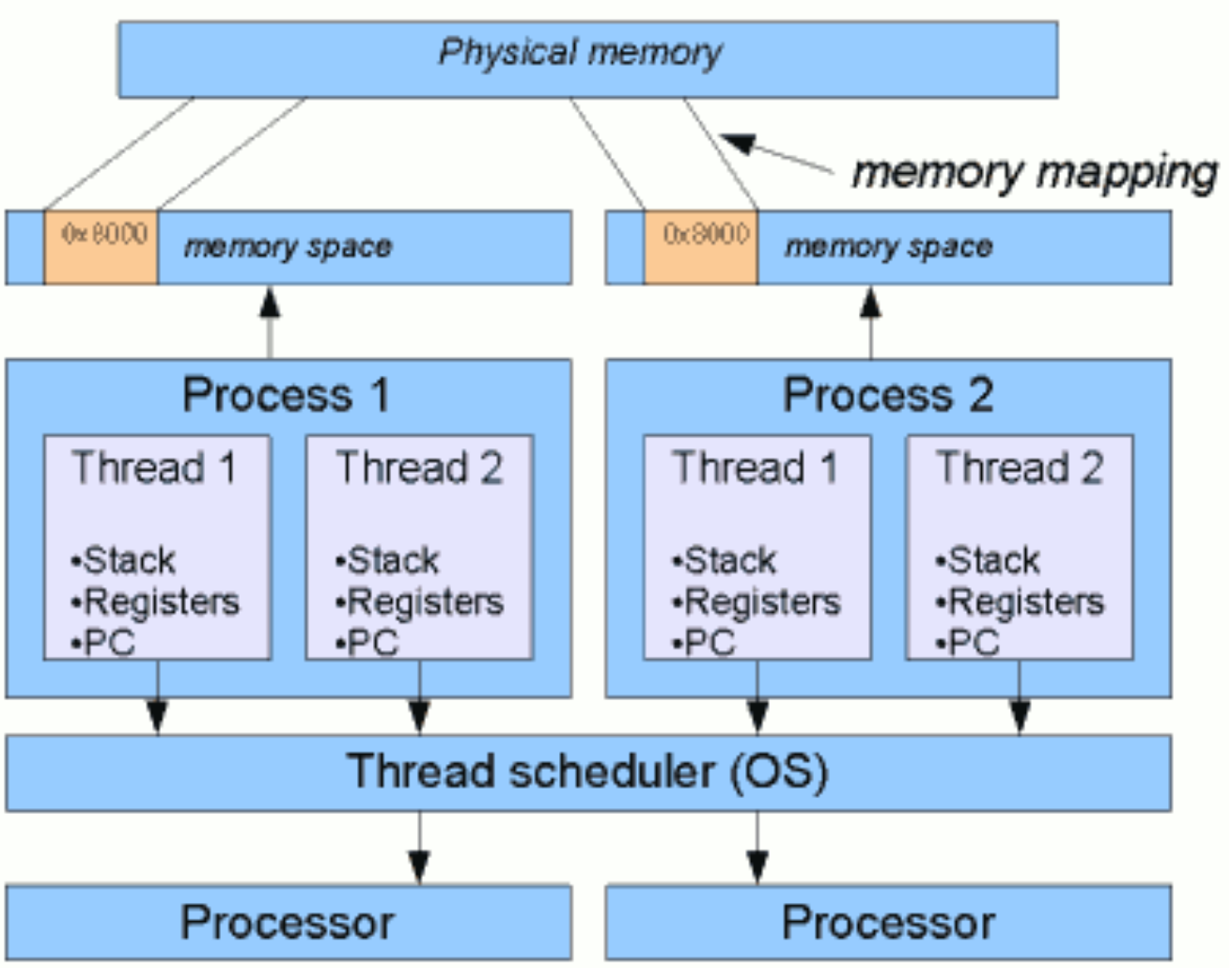


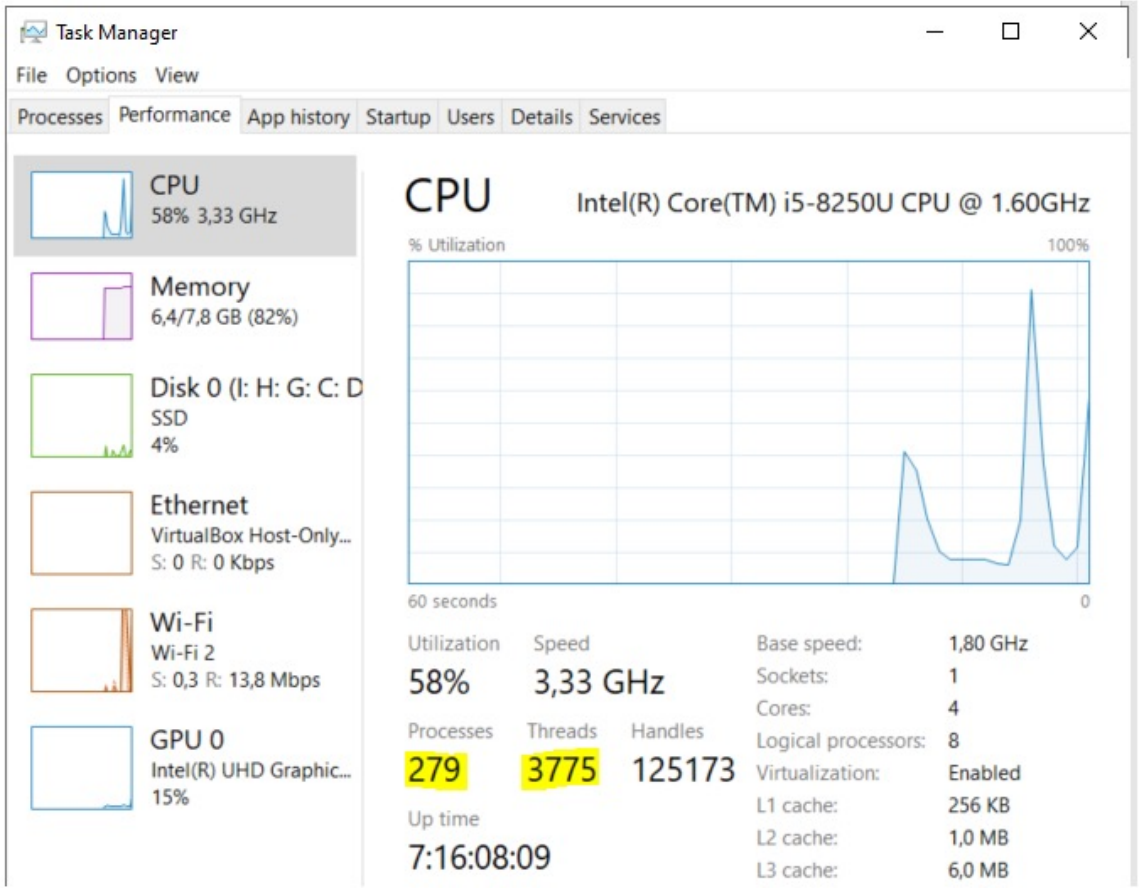
Khác biệt giữa process vs thread

	Process	Thread
1	Chương trình đang thực thi	Một phần của process
2	Tốn nhiều thời gian để tạo/kết thúc, chuyển ngữ cảnh	Tốn ít thời gian tạo/kết thúc, chuyển ngữ cảnh
3	Giao tiếp giữa các P phức tạp, sử dụng IPC	Giao tiếp giữa các thread đơn giản
4	Các process độc lập tài nguyên	Các thread trong một process chia sẻ tài nguyên
5	Process có PCB, stack và không gian nhớ	Thread có parent's PCB, TCB (thread control block), stack và chia sẻ không gian nhớ.
6	Một process bị block có thể dẫn đến process khác bị block	Một thread ở trạng thái blocking hoặc waiting thì thread khác vẫn có thể hoạt động
7	Thay đổi ở Parent process không ảnh hưởng đến child process	Thay đổi ở main thread có thể ảnh hưởng đến các thread khác trong cùng process
8	Các trạng thái: new, ready, running, waiting, terminated	Các trạng thái: running, ready, blocked

Khác biệt giữa thread vs process

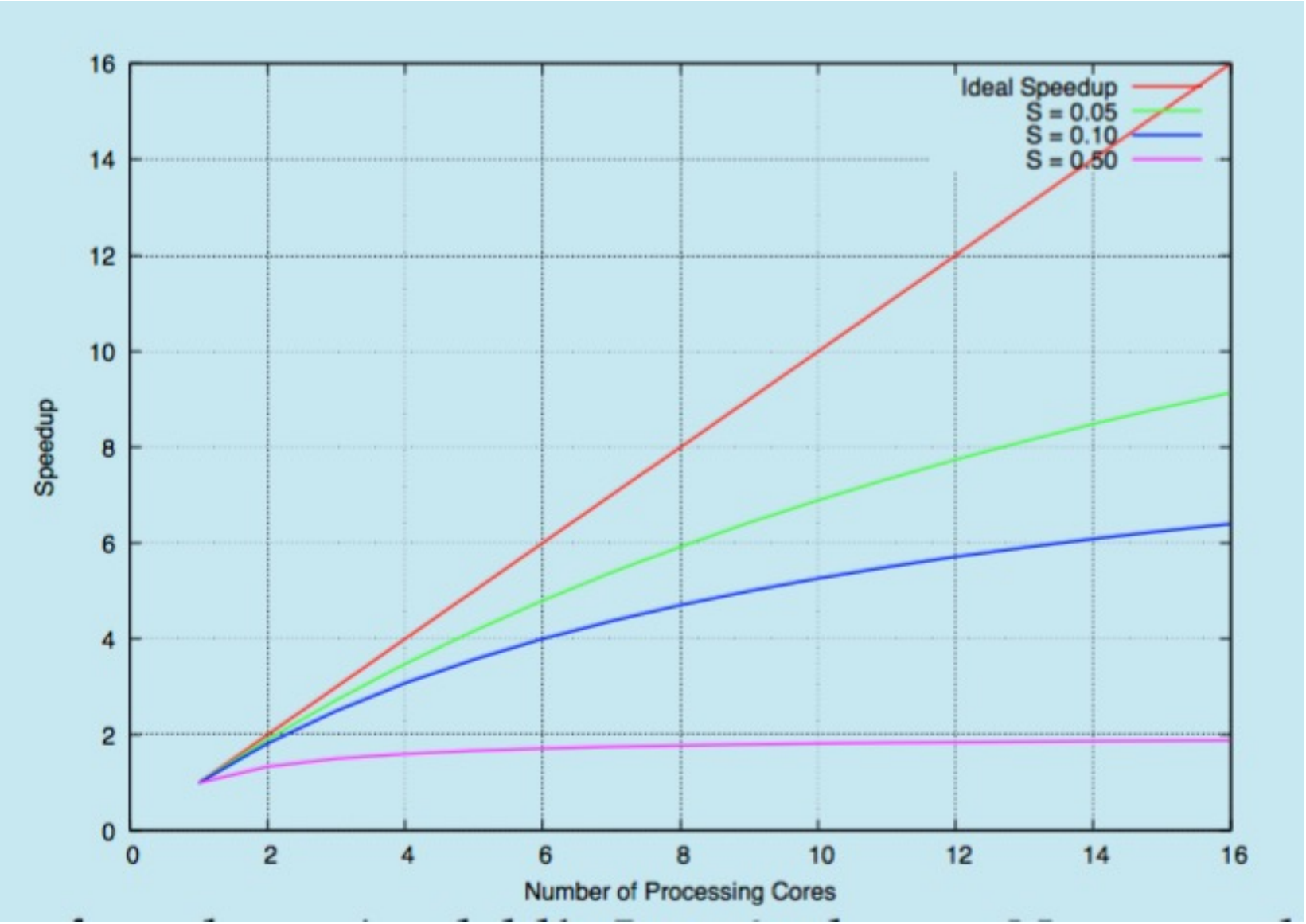
- OS hiện đại thường sử dụng multithreading





Amdahl's Law

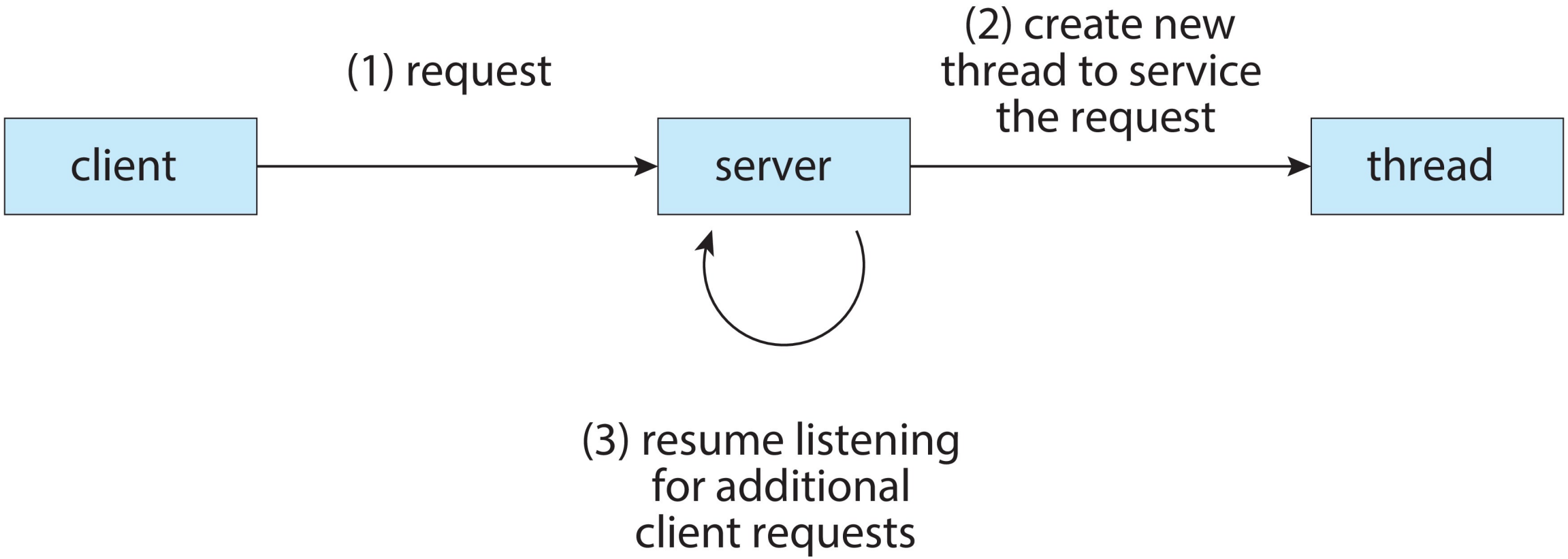
- Có phải càng tạo nhiều thread càng tăng tốc độ thực thi chương trình?



$$speedup \leq \frac{1}{S + \frac{(1-S)}{N}}$$

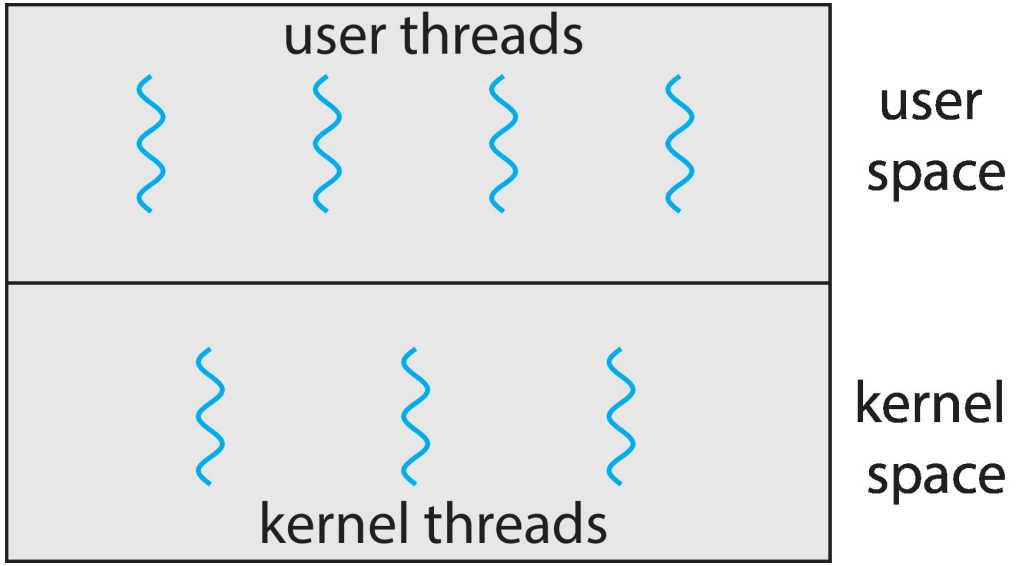
- S: tỉ lệ code thực hiện tuần tự.
- N: số CPU core
- Nếu N → ∞ ?

Kiến trúc multithread tại server



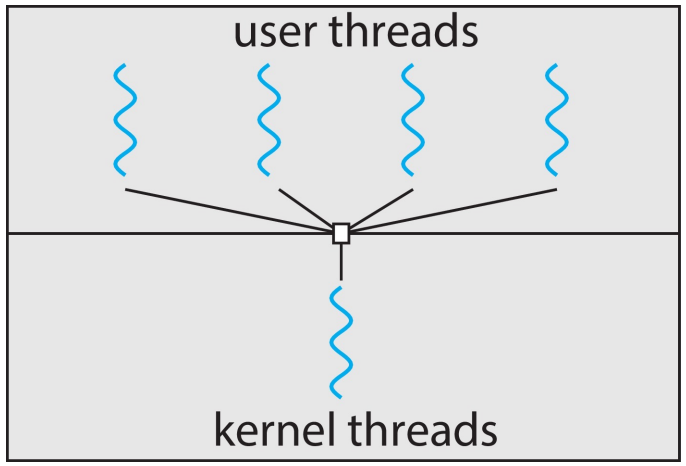
Phân loại thread

- User level thread: do người dùng tạo bằng các thư viện:
  - POSIX Pthreads
  - Java threads
- Kernel level thread: hỗ trợ bởi kernel
  - Windows threads



Phân loại thread

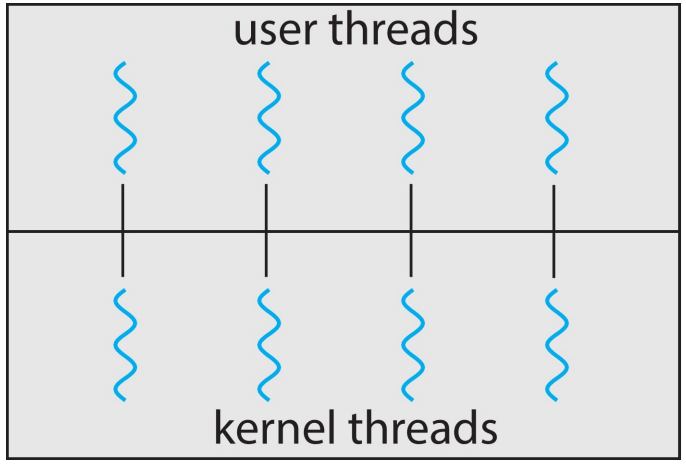
- Multithreading models:
  - Many to one
  - One to one
  - Many to many



user space

kernel space

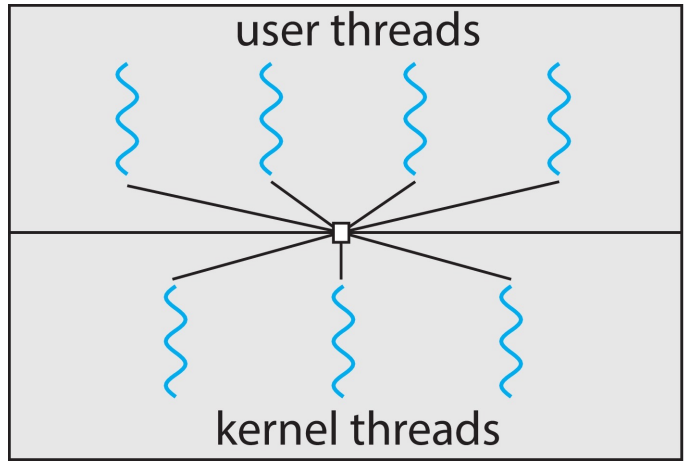
Many to one, ex: Solaris Green Thread, GNU Portable Thread



user space

kernel space

One to one, ex: Windows, Linux

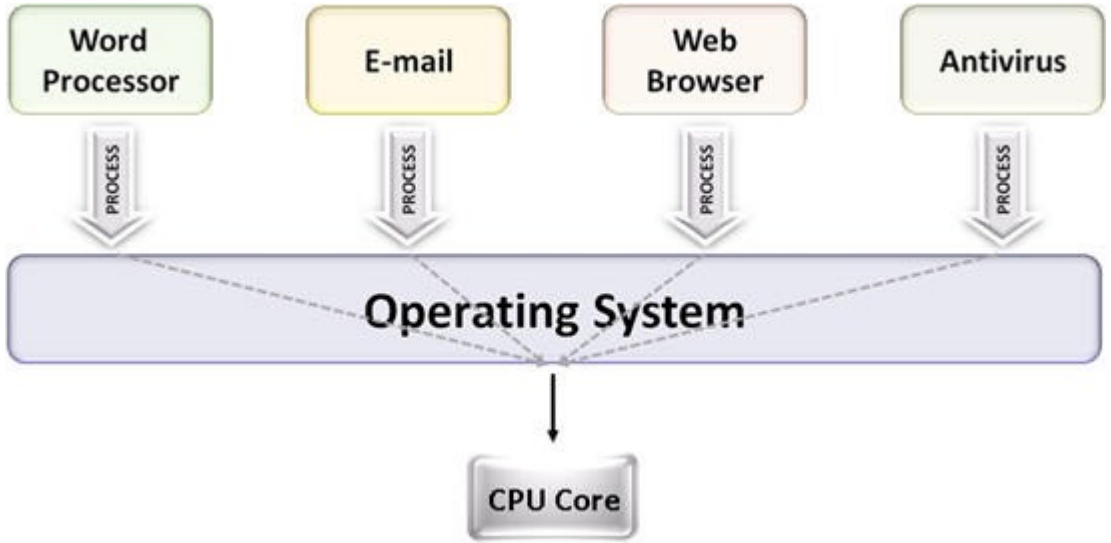


user space

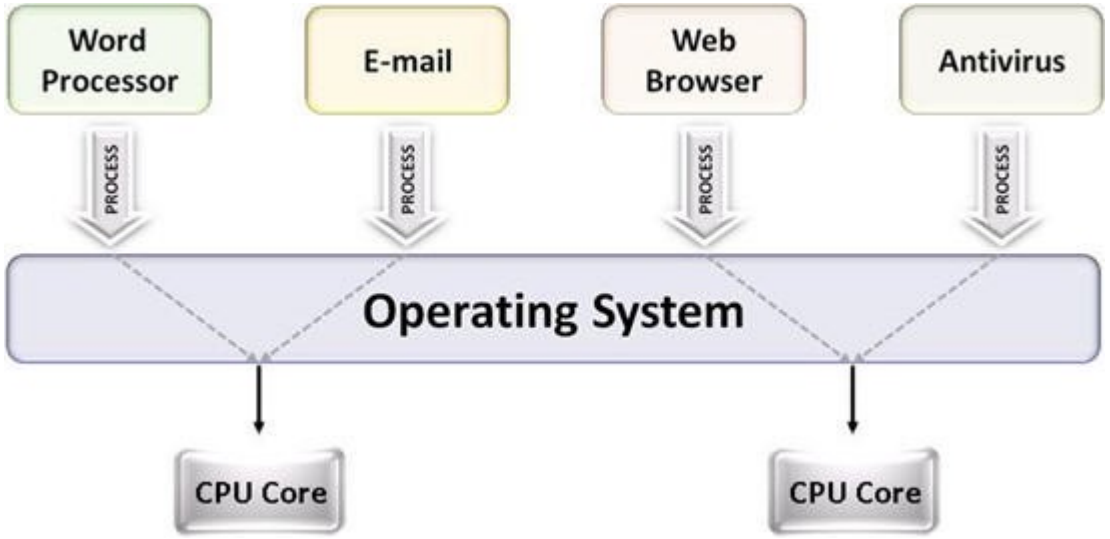
kernel space

Many to many, ex: ThreadFiber Windows

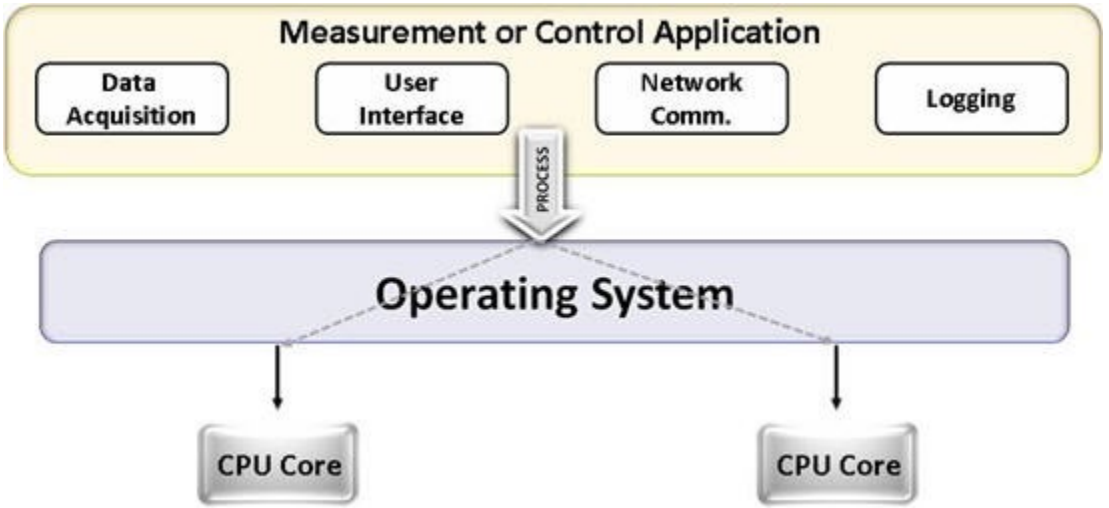
Multitasking



Single-core systems schedule tasks on 1 CPU to multitask



Dual-core systems enable multitasking OS to execute two tasks simultaneously



Dual-core system enables multithreading

Ví dụ tạo thread trên Linux

```
#include <pthread.h>
#include <stdio.h>

#include <stdlib.h>

int sum; /* this data is shared by the thread(s) */
void *runner(void *param); /* threads call this function */

int main(int argc, char *argv[])
{
   pthread_t tid; /* the thread identifier */
   pthread_attr_t attr; /* set of thread attributes */

   /* set the default attributes of the thread */
   pthread_attr_init(&attr);
   /* create the thread */
   pthread_create(&tid, &attr, runner, argv[1]);
   /* wait for the thread to exit */
   pthread_join(tid,NULL);

   printf("sum = %d\n",sum);
}
```

```
/* The thread will execute in this function */
void *runner(void *param)
{
   int i, upper = atoi(param);
   sum = 0;

   for (i = 1; i <= upper; i++)
      sum += i;

   pthread_exit(0);
}
```

Tham khảo

- Operating System Concepts, 10th edition
- https://www.geeksforgeeks.org/thread-in-operating-system/
- https://www.geeksforgeeks.org/difference-between-user-level-thread-and-kernel-level-thread/