แก้ไขครั้งล่าสุดเมื่อวันที่ 4 กันยายน 2567

Thread outline

What is thread?

User-level thread / Kernel thread / Hardware thread

Simultaneous multi-treading (SMT or HT), Multi-core

Multi-threading model: many-to-one, one-to-one, many-to-many

Thread libraries: Pthreads, Win32, Java, .NET

Thread pools

Windows XP / Linux threads

Homework: server/client, matrix multiplication (optional)

Threads

Thread is light-weight process.

Single-threaded process vs. multi-threaded process

Benefits: responsiveness, resource sharing, economy, scalability

Multi-core programming

Challenges: dividing activities, balance, data splitting, data dependency, testing and debugging

Multi-threading models:

- **user-level vs. kernel-level threads**
- **M:1, 1:1, M:N**

Kernel vs. User-level threads

Process

หมด **time slice** ↓ **I/O block** ↓

Kernel thread

ใช้ CPU scheduling ของ OS เราไม่สามารถควบคุมได้ 100% ว่าจะให้ทำ thread ไหนก่อน แต่อาจจะกำหนด priority คร่าวๆ ได้

กินทรัพยากรระบบมาก และ 1 thread อาจจะได้ time slice เท่า 1 process

ถ้า spawn thread ออกมามาก ก็ยิ่งโหลด CPU มาก เครื่องอาจจะ hang

I/O block ↓

Process P1 ←——→

User-level thread

เอา time slice ของ process มาแบ่งใช้เอง

ใช้ CPU scheduling ของ thread library

I/O block จะทำให้เกิด context switch ไป process ต่อไปทันที

ระวัง thread ในบางภาษาเป็น user-level thread เช่น Python

I/O block ↓

ข้อดีของ user-level thread คือใช้ได้กับทุก OS แต่จะไม่ได้ performance เพิ่มขึ้น

Global Interpreter Lock

What is GIL ?

Wiki Definition: Global interpreter lock (GIL) is a mechanism used in computer language interpreters to synchronize the execution of threads so that only one native thread can execute at a time.

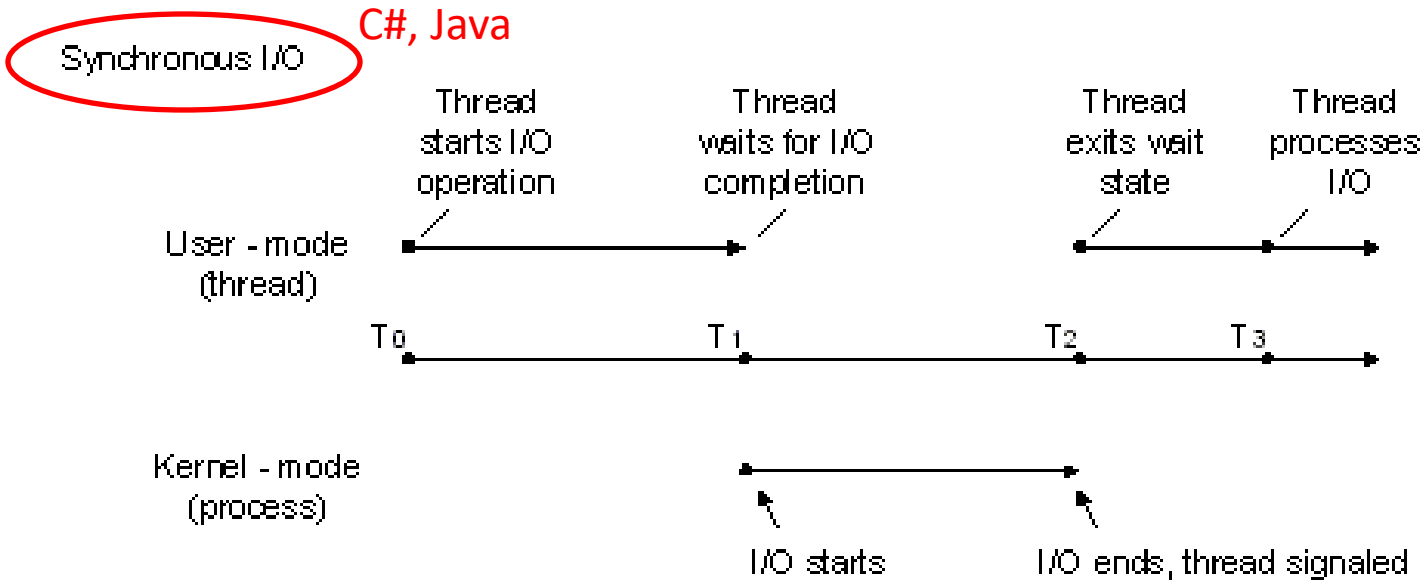
Why python uses GIL ?

In CPython, the global interpreter lock, or GIL, is a mutex that prevents multiple native threads from executing Python bytecodes at once. This lock is necessary mainly because CPython's memory management is not thread-safe.

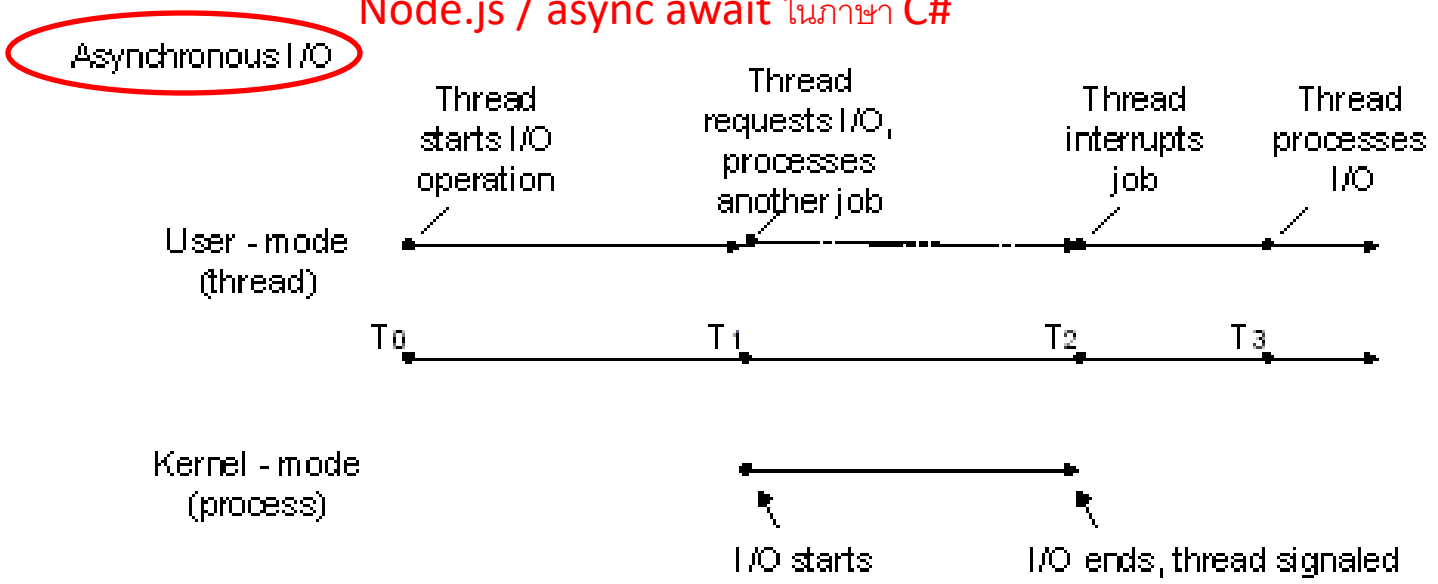
http://anuragjain67.github.io/writing/2016/01/15/problem-with-multithreading-in-python

http://msdn.microsoft.com/en-us/library/aa365683(VS.85).aspx

C#, Java



Node.js / async await ในภาษา C#



User-level thread หรือ Green thread example (obsolete!)

- **Java 1.1 on Solaris (ใช้ multi-core ไม่ได้ และอาจจะทำ thread อื่นรวมถึง JVM หยุดไปด้วย ถ้าเจอ I/O block)**
- **Ruby < 1.9**
- **Cpython (standard interpreter ของภาษา Python)**
- http://en.wikipedia.org/wiki/Green_threads

Web browsers (ปัจจุบันอาจจะเปลี่ยนไปแล้ว)

- **Firefox, Safari** — **one process (many threads)**
- **Chrome** — **many processes (per site instance)**

โปรแกรม **chrome.exe** สามารถตั้ง **parameter** ได้ **(ปัจจุบันอาจจะเปลี่ยนไปแล้ว)**

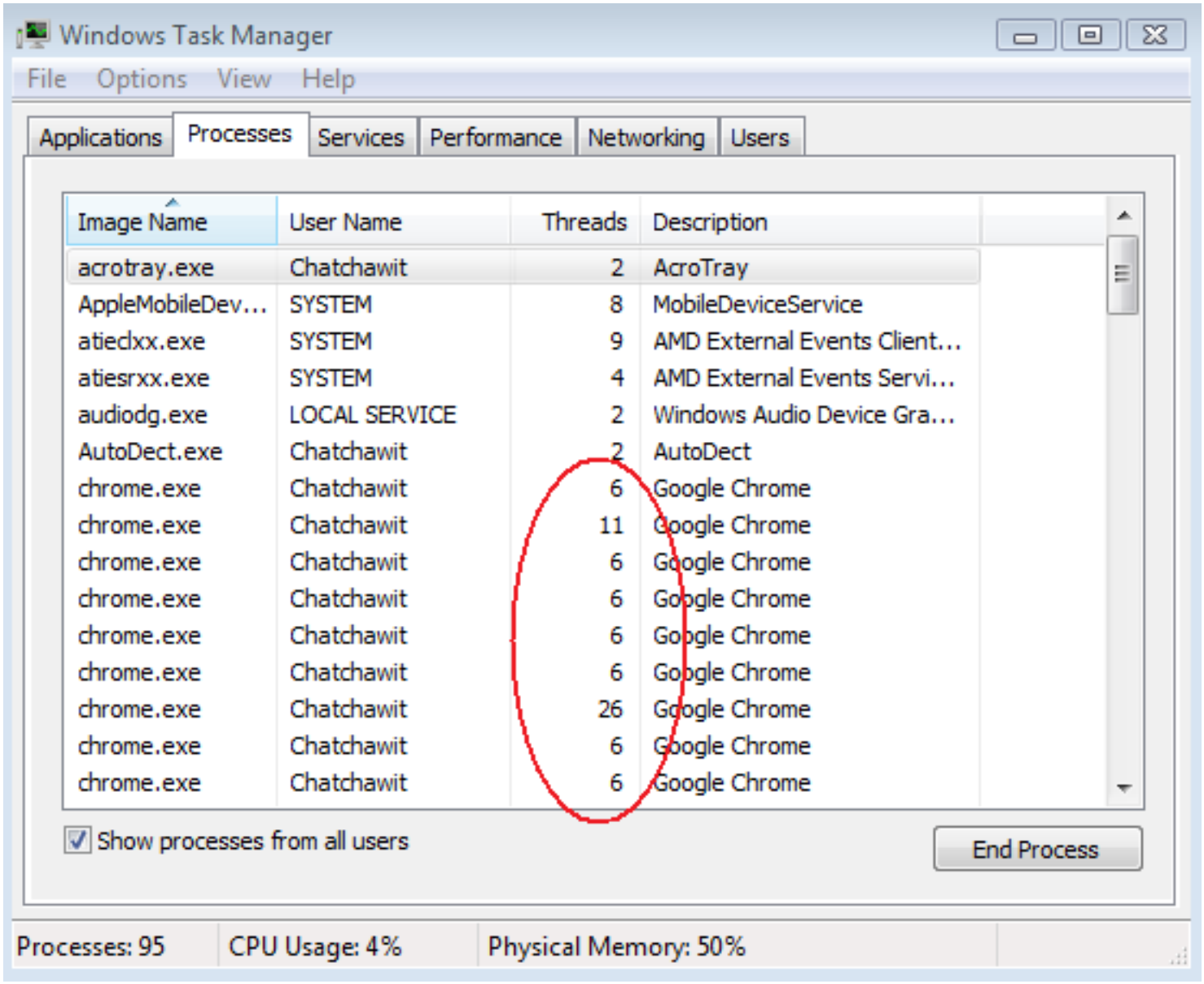
4 process models

ใน 1 site instance
เช่น เป็น parent/child tabs กัน

Process-per-site-instance	(default) 1 site instance มีหลาย tabs ได้
Process-per-site	--process-per-site
Process-per-tab	--process-per-tab
Single process	--single-process เหมือน Firefox, Safari

http://www.chromium.org/developers/design-documents/

เพิ่มคอลัมน์ Threads ใน Task Manager (เพื่อดูจำนวน kernel thread ของแต่ละ process)

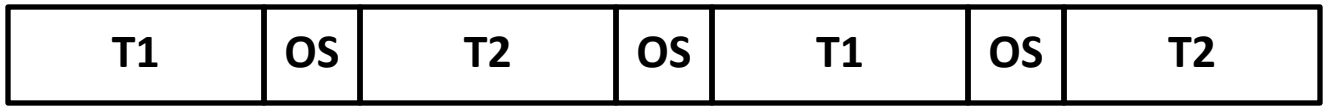


Hyperthreading vs. Multi-core

Physical vs. Logical processor

เพิ่มฮาร์ดแวร์เพื่อหลอกให้ OS เข้าใจว่ามี processor มากกว่า 1 ตัว (จริงๆ มีตัวเดียว) เช่น เพิ่ม pc และ registers เป็น 2 ชุด CPU scheduling ต้องหา thread มาเติม ตามจำนวน logical processor ที่มี

Single core



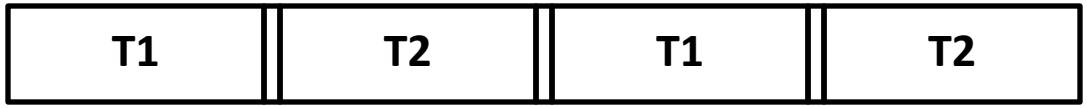
ใช้ SW (OS) ในการทำ context switch

1 physical processor
2 logical processors

Hyperthreading

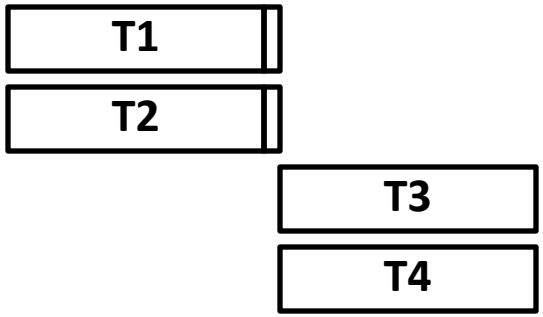
รอดู slide บทถัดไป จะละเอียดกว่านี้

thread มักจะติด I/O block แล้วก็ต้องทำ context switch ดังนั้นใส่ thread เข้าไป 2 ตัวเลย (แต่มี core เดียว) พอติด I/O block ก็ switch เร็วๆ เพื่อไปทำอีก thread นึง

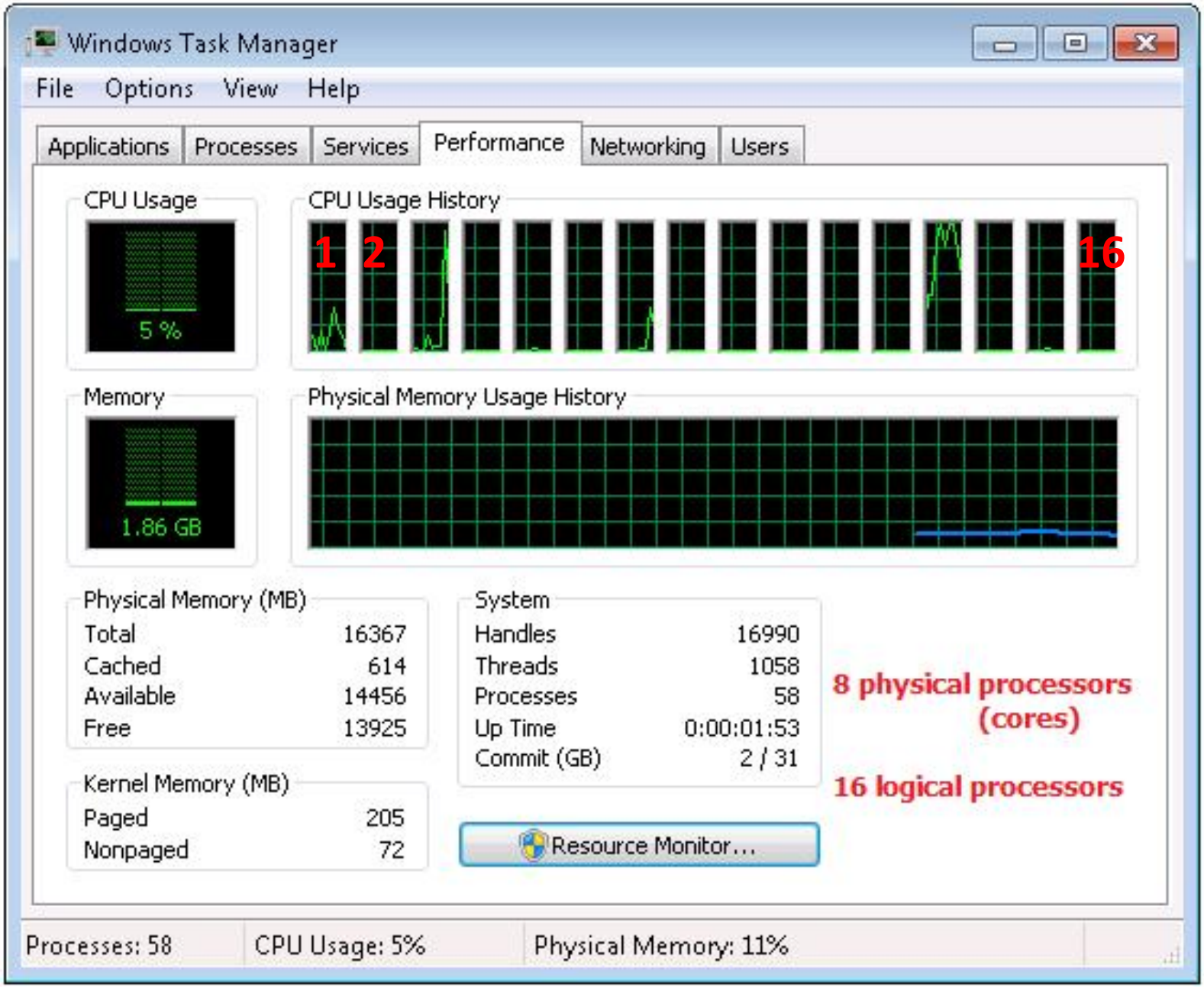


ใช้ HW ในการทำ context switch ระหว่าง 2 thread ทำให้โปรแกรมโดยรวมเร็วขึ้น 20-30% (เนื่องจากมี pc และ registers 2 ชุด ทำให้ switch ได้เกือบจะทันที)

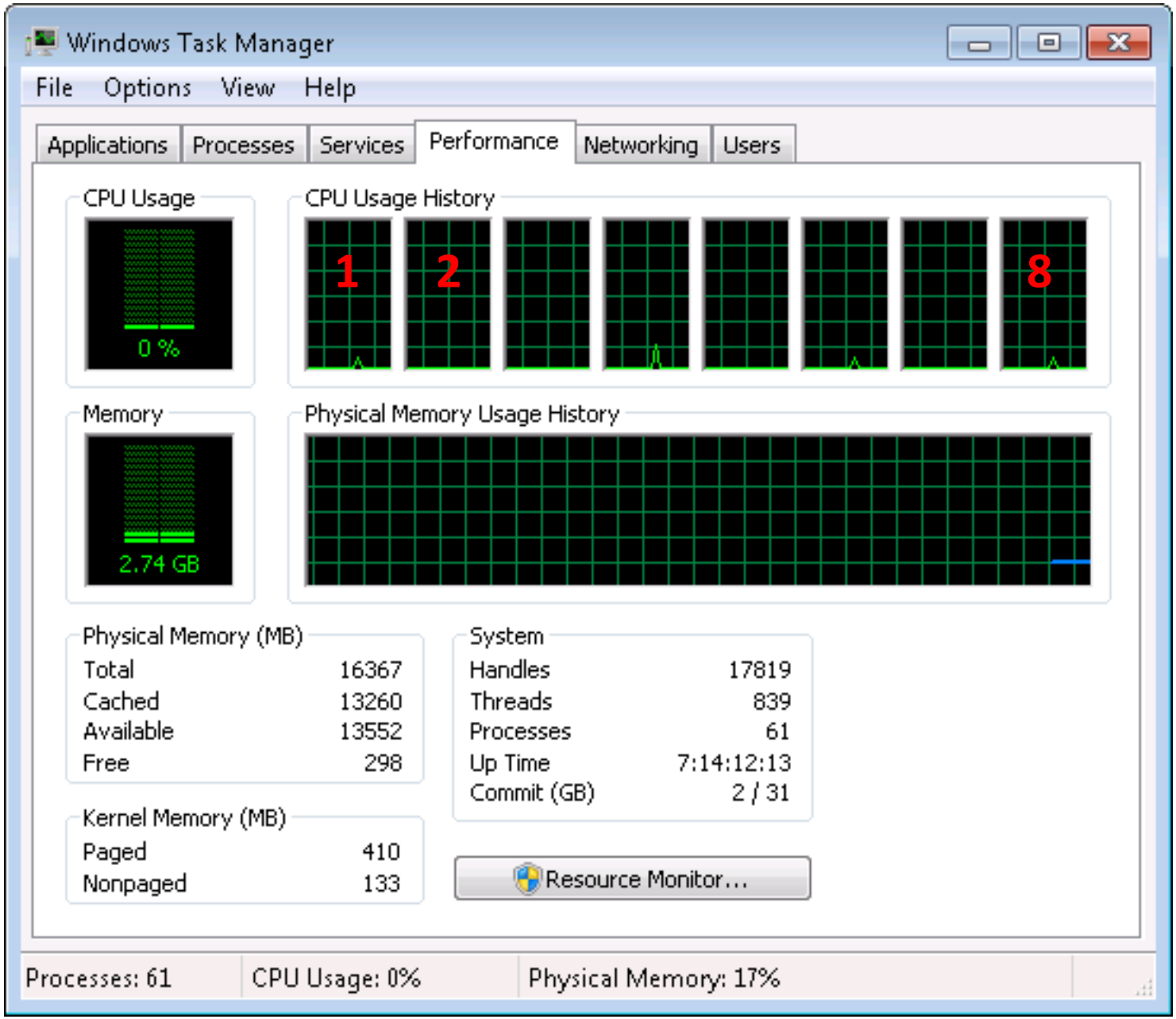
Multi-core + hyperthreading



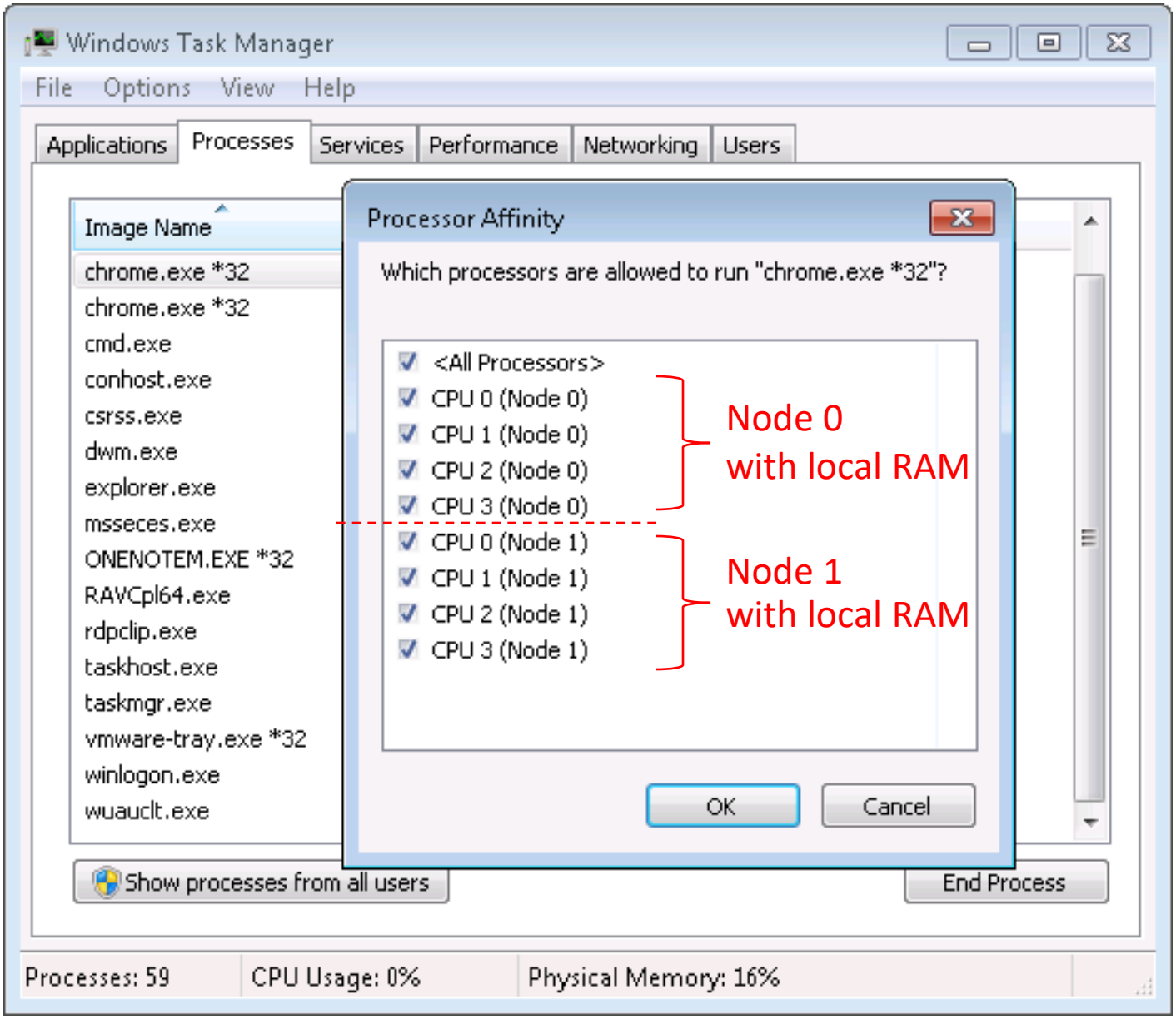
CPU 2 ตัว เป็น quad-core แต่ละ core เป็น hyper-threading (set ใน BIOS)



Disable hyper-threading



Processor Affinity



Thread libraries

Pthreads, Win32, Java

pros and cons 

Java thread API is generally implemented using a thread library available on the host system (Windows / Linux / macOS).

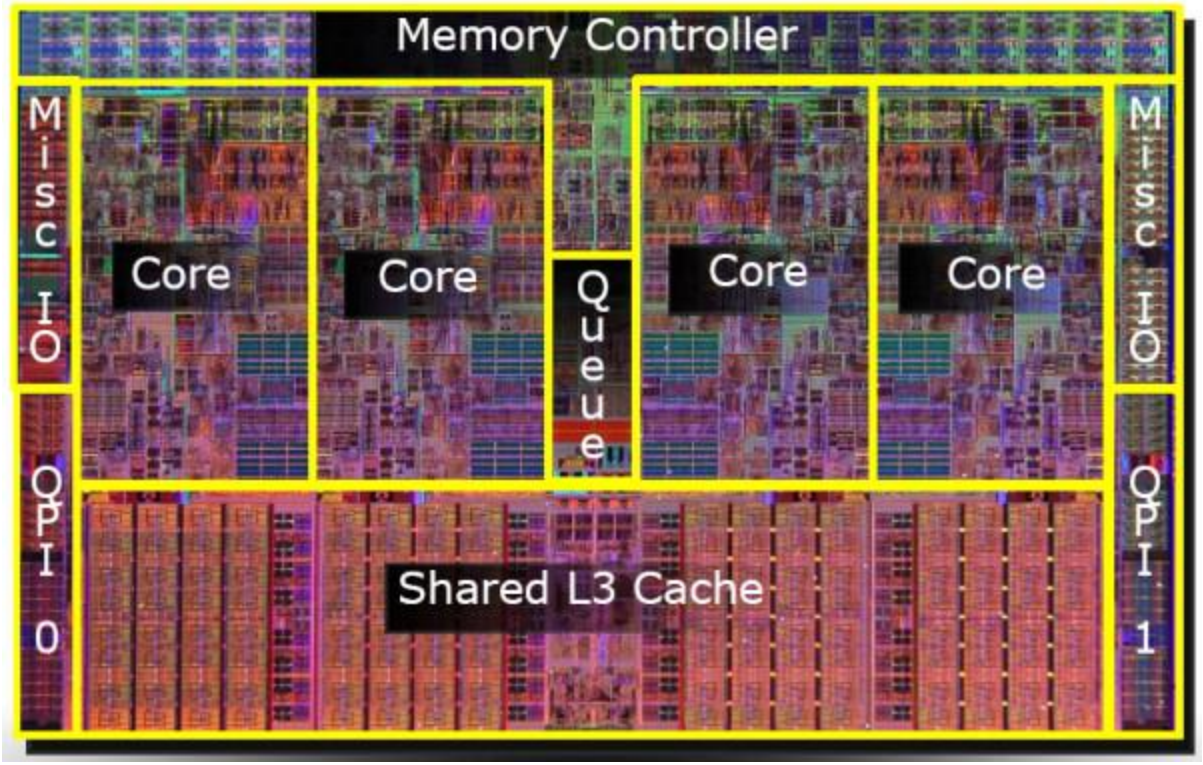
Problem: $$sum = \sum_{i=0}^{N} i$$

Pthreads 

Win32 

Java threads 

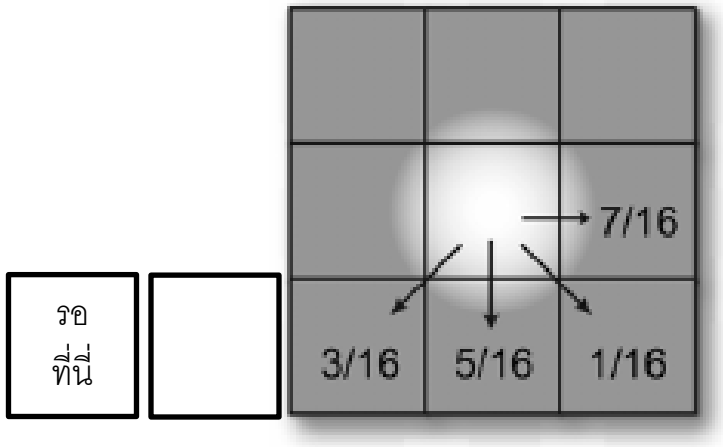
Multi-core programming

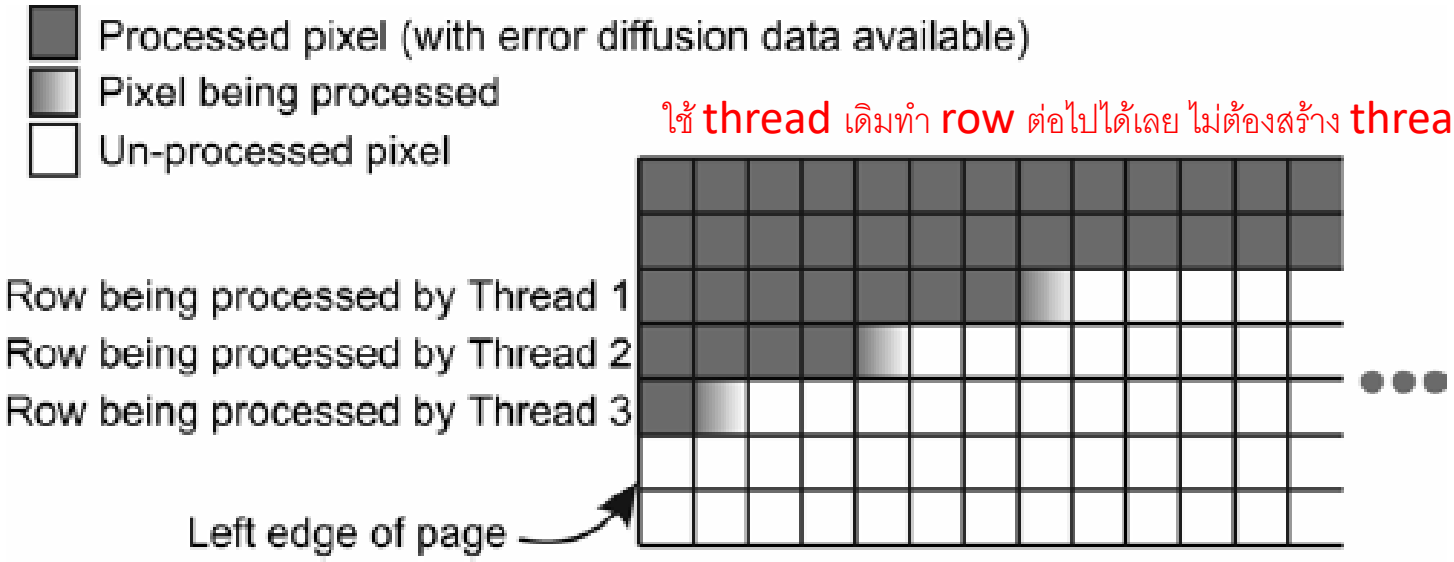


Error diffusion

Input: 8-bit grayscale image (1 million pixels, 10K x 10K)
Output: black & white image (same resolution)

Lena, 1972



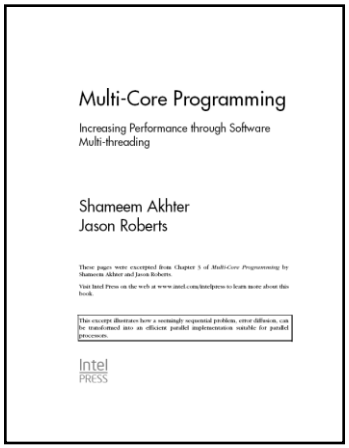


Multiple threads are able to process multiple rows simultaneously.

Error diffusion

Hint:

1. put space on the edge
2. use “byte” to save memory space
3. use integer programming
4. use n threads, n < #cores (next row = row + n)
5. create n threads and reuse them
6. ใช้ C# หรือ Java


Multi-Core Programming

Increasing Performance through Software Multi-threading

Shameem Akhter
Jason Roberts

These pages were excerpted from Chapter 3 of *Multi-Core Programming* by Shameem Akhter and Jason Roberts.

Visit Intel Press on the web at www.intel.com/intelpress to learn more about this book.

This excerpt illustrates how a seemingly sequential problem, error diffusion, can be transformed into an efficient parallel implementation suitable for parallel processors.

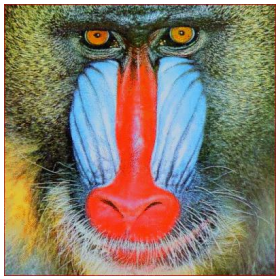
Intel PRESS

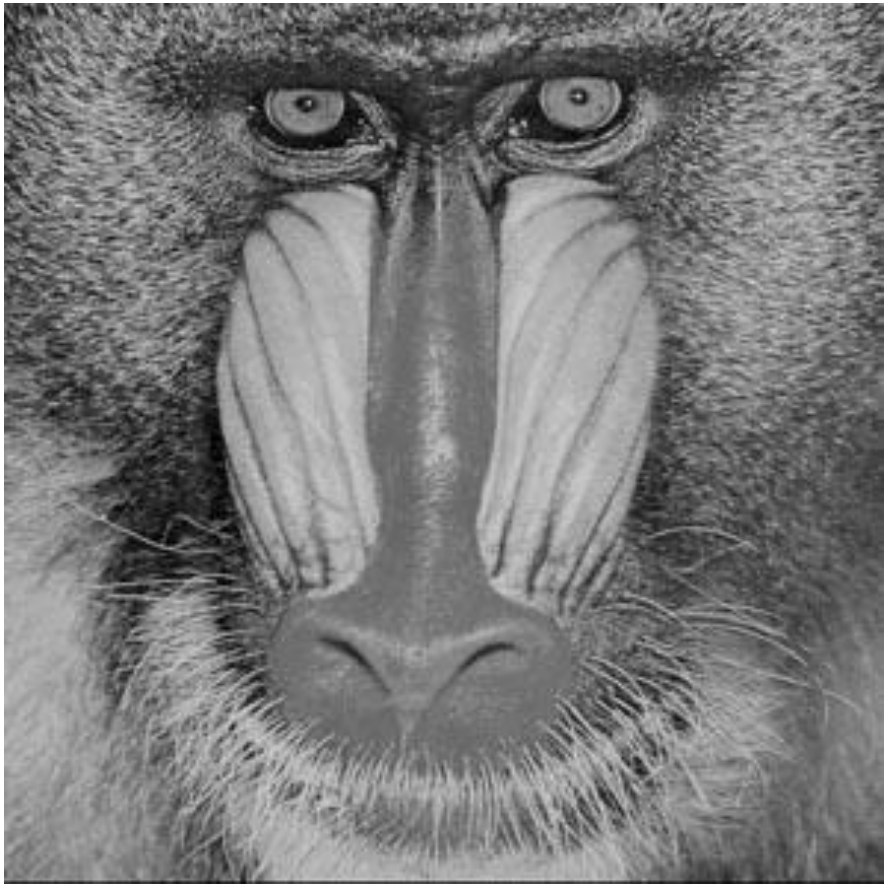
```
for (i = 0; i < HEIGHT; i++) {
        for (j = 0; j < WIDTH; j++) {

                OutputImage[i][j] = (InputImage[i][j] < 128 ? 0 : 1);

                err = InputImage[i][j] - (255 * OutputImage[i][j]);

                InputImage[i][j+1] += (err * 7) / 16;
                InputImage[i+1][j-1] += (err * 3) / 16;
                InputImage[i+1][j] += (err * 5) / 16;
                InputImage[i+1][j+1] += (err * 1) / 16;
        }
}
```

00 = black, FF = white

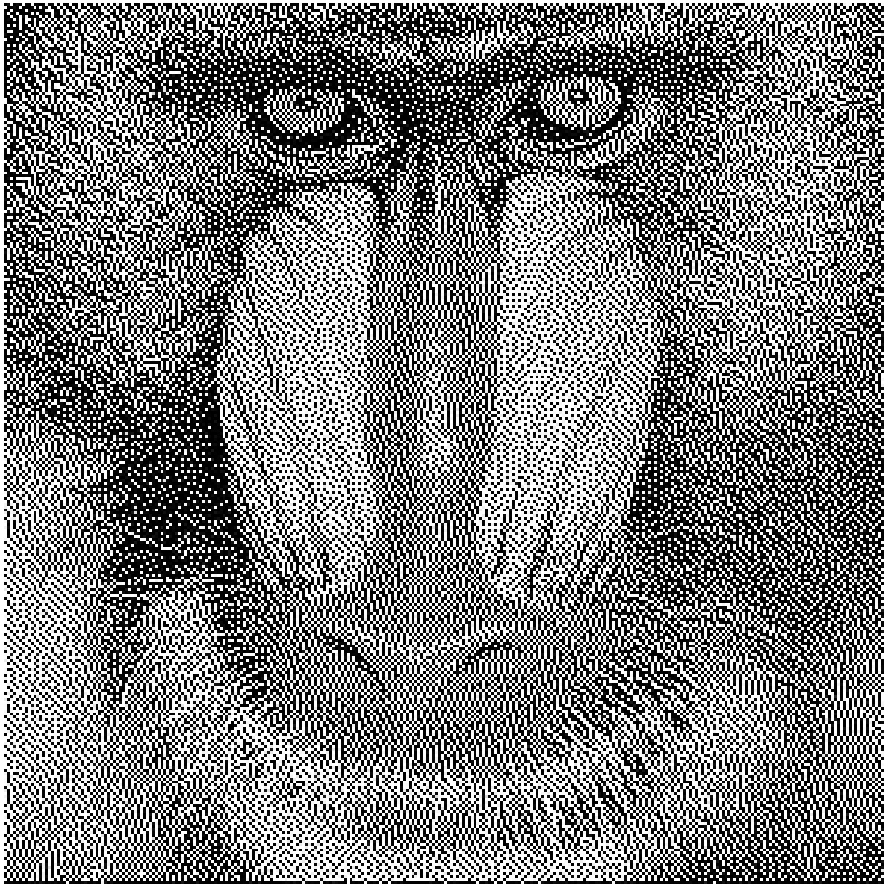
เช็คคำตอบ sequential vs. parallel

Example: Baboon



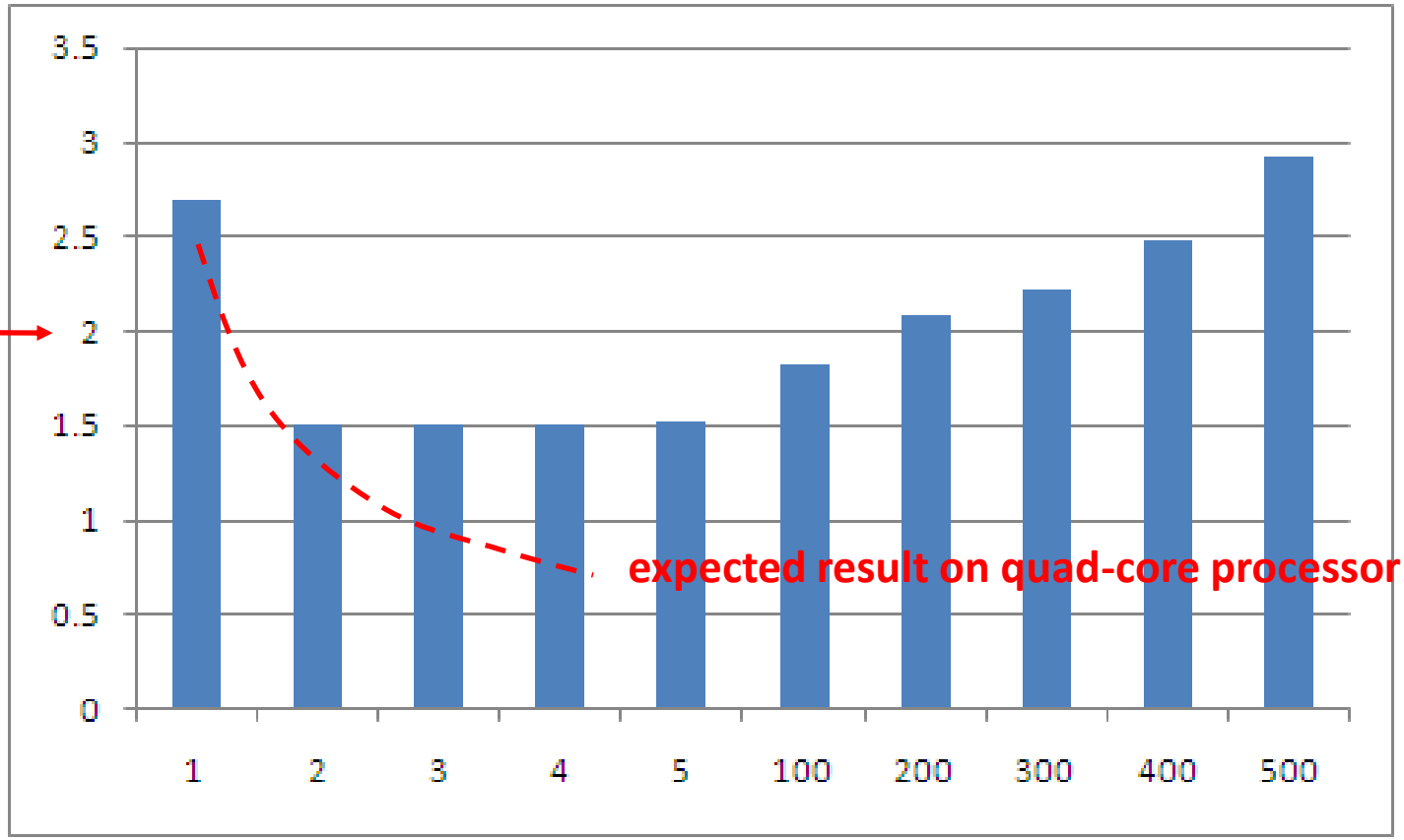


Grayscale
(298 x 298)



Black & white
(298 x 298)

เริ่มจับเวลาหลังสร้าง thread และหยุดจับเวลาหลัง join

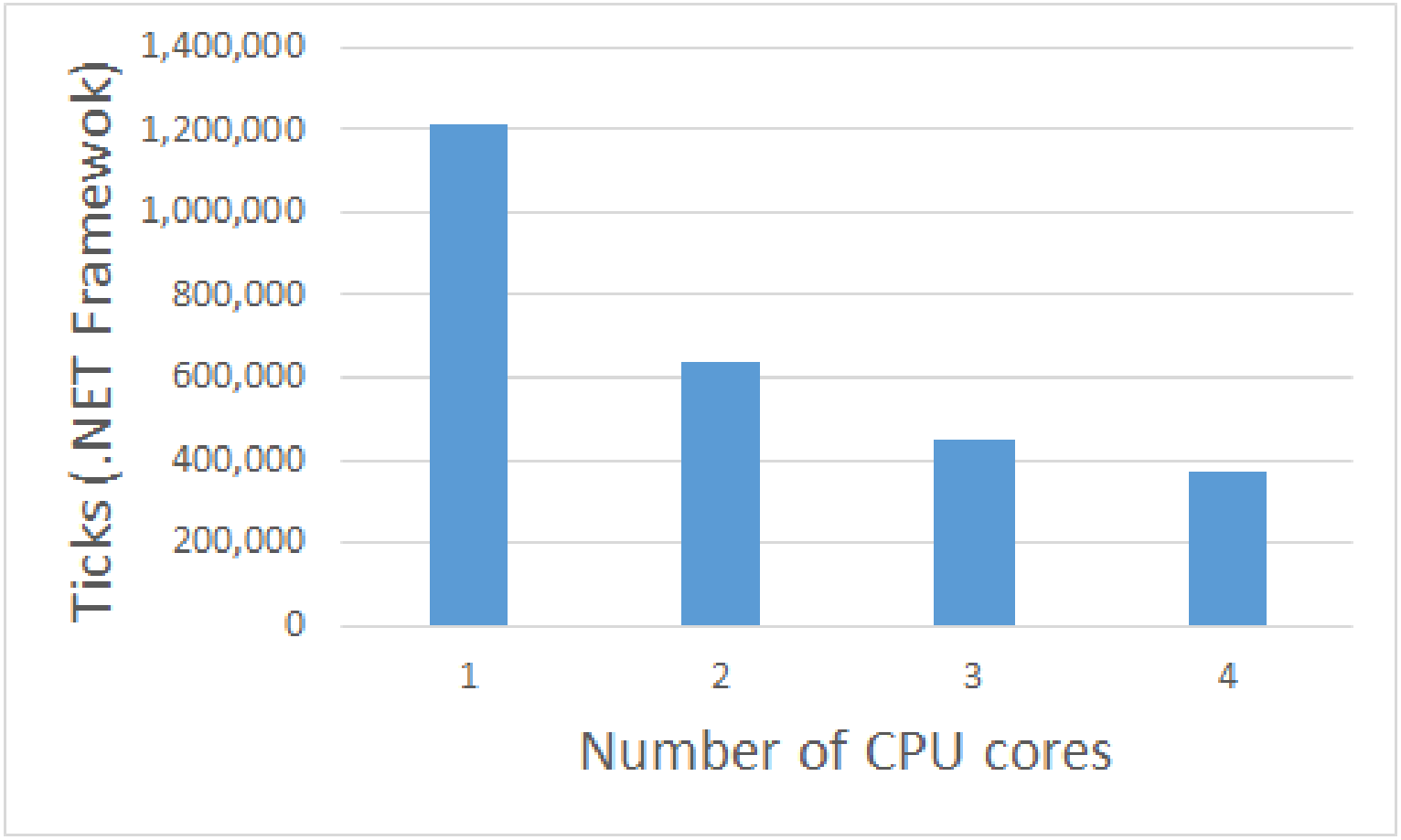


Number of threads (X-axis) vs. Wallclock time (Y-axis)

Intel CoreTM2 CPU T5500 1.66 GHz

10,000 x 10,000 pixels

Speedup on a quad-core processor

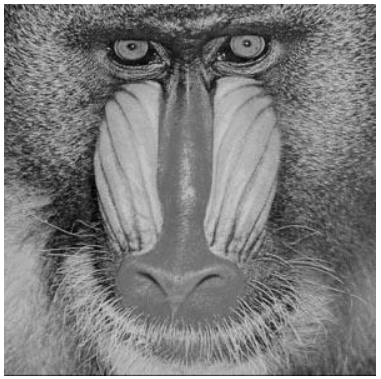


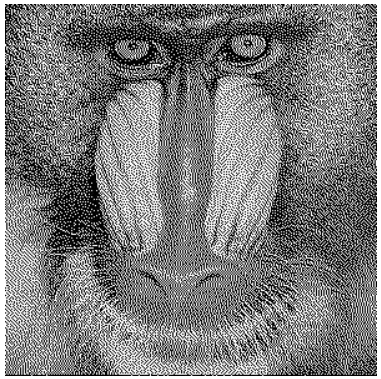
กิจกรรม: Error Diffusion

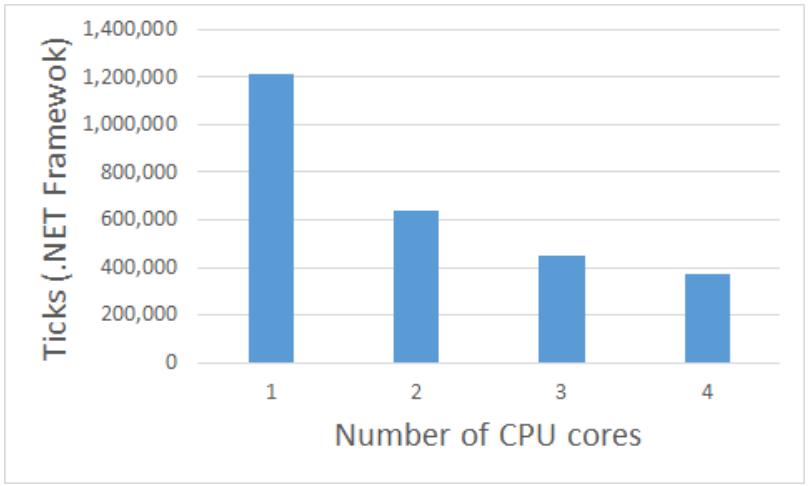
ทำบน Host OS เท่านั้น
ไม่ทำใน virtual machine

ส่ง

- รูป 8-bit grey scale image (ก่อน)
- รูป 8-bit grey scale image (หลัง)
- กราฟแท่ง แกน x มีอย่างน้อยเท่าจำนวน core
- อธิบายโค้ด ทำยังไงไม่ให้ thread วิ่งแซงกัน







lock Statement (C# Reference)

- โปรแกรมนี้ถ้าไม่ lock จะทำงานผิดพลาดอย่างไร
- ทำไมถ้ามี lock แล้วทำงานได้ถูกต้อง
- ถ้าตัวแปร lock เป็นแบบ static จะเกิดอะไรขึ้น
- Error Diffusion ไม่ต้องใช้ lock

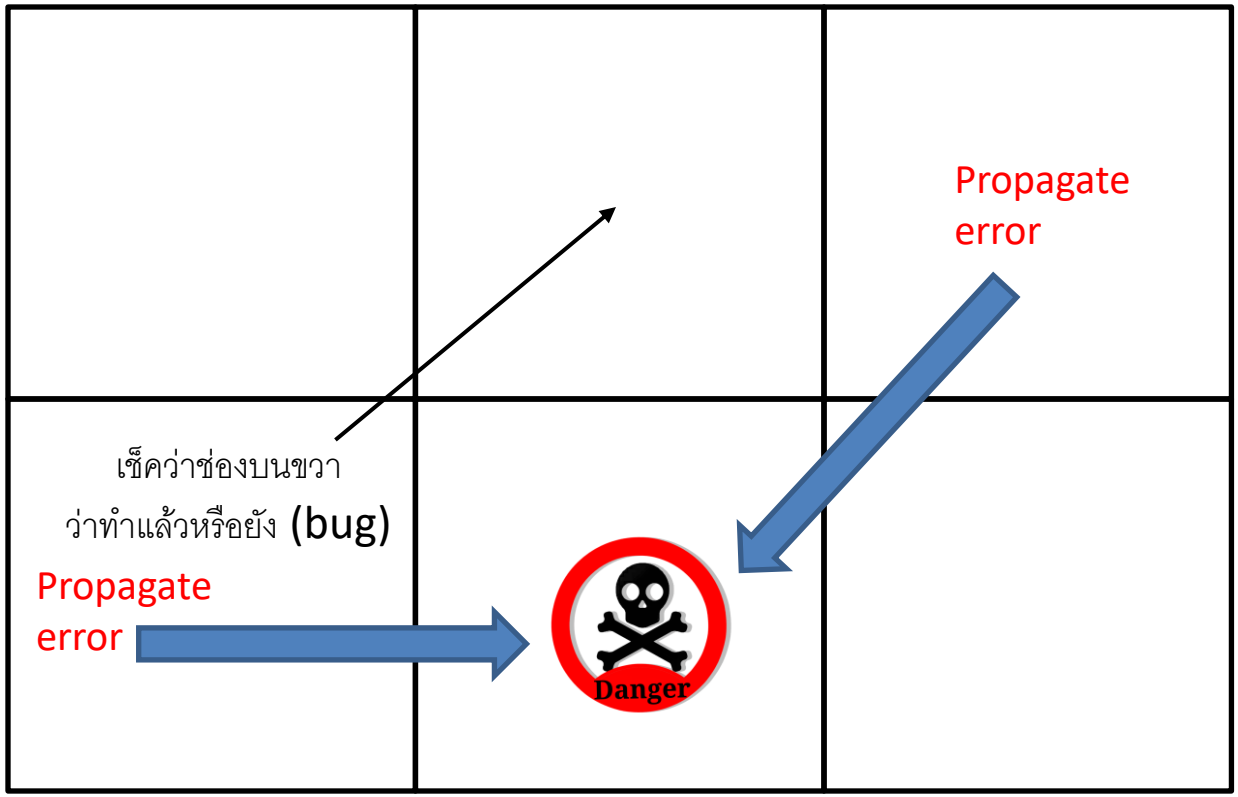
```
class Account
{
    decimal balance;
    private Object thisLock = new Object();

    public void Withdraw(decimal amount)
    {
        lock (thisLock)
        {
            if (amount > balance)
            {
                throw new Exception("Insufficient funds");
            }
            balance -= amount;
        }
    }
}
```

ถ้า lock แล้ว thread อื่นจะ lock ซ้ำอีกไม่ได้ ต้องหยุดรอ

https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/lock-statement

Error Diffusion ไม่จำเป็นต้องใช้ lock

เช็คว่าช่องบนขวาทำไปแล้วหรือยัง (bug)



Thread cancellation

- **Asynchronous cancellation**
- **Deferred cancellation**

Signal Handling

Thread pool (Win32 API, Java), IIS performance tuning

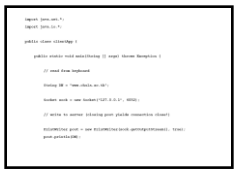
Scheduler activations

Windows XP threads

Linux threads

Homework

Client-server



Write a Java program for server and client.
Server ทำหน้าที่แปลง **domain name** เป็น **ip address**

Matrix multiplication

Write a C/C++ program to perform matrix multiplication.
(option: fix number of threads, or using thread pool)