COLLEGE OF ARTS AND SCIENCES
UNIVERSITY OF THE PHILIPPINES LOS BAÑOS
1/F Wing C Physical Sciences Bldg., Harold Cuzner Royal Palm Ave.
College 4031, Laguna, Philippines
Phone +63 49 536 2313
ics.uplb@up.edu.ph www.ics.uplb.edu.ph
INSTITUTE OF COMPUTER SCIENCE

CMSC 180: Introduction to Parallel Computing

Threads with Core-Affinity

Clinton E. Poserio

Introduction

In order to make truly parallel programs, you will need to review familiar terms and concepts related to parallel computing.

Contents

Learning Objectives

At the end of this discussion, the student must be able to:

- describe the basic components of a parallel processor;
- understand the concept of multicore systems; and
- differentiate concurrency and parallelism.

Discussion

CPUs, Cores, and Threads

Let us review the most important part of a parallel system, the processing unit. The most common processing unit installed in our computer is a CPU, an integrated circuit (IC) chip, usually of rectangular shape, which can be seen embedded or installed in the motherboard. From our basic IT literacy course, we know that a CPU has several internal components such as the arithmetic logic unit (ALU) and the control unit (CU). CPUs or processors are often advertised with their number of *cores* and their number of *threads*.

Let's begin with threads. You might remember from your CMSC125 that a thread is a basic unit of computation i.e. it is a virtual or software concept describing a task to be executed by the CPU. During the old days, computers existed with only a single CPU core, which means (at the lowest level) instructions are only executed one at a time. Basic computing tasks were not as fast as they are today. In 2002, Intel introduced their Hyper-Threading Technology (HTT) on their

CMSC 180: Introduction to Parallel Computing

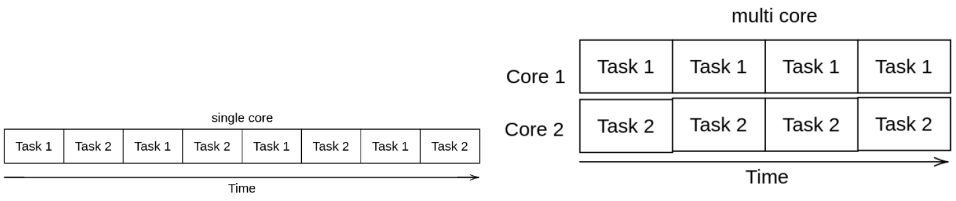
processor models. Without loss of generality, the HTT tricks the operating system; a single physical CPU core is seen as two virtual cores (aka *logical cores* or *vCPUs*) by the operating system using the concept of threads. The HTT allowed the operating system to execute instructions from multiple processes and increase the parallelization of executions [1].

Due to the advancement in processor design and manufacturing, modern processors now contain multiple CPUs in a single IC chip. These individual CPUs are often referred to as *processor cores* (or *CPU cores* or simply cores). This means that a CPU with multiple cores also has multiple sets of ALU, CU, and other CPU execution units.

The individual physical CPU cores in combination with the hyper-threading technology gives as double the processing power (assuming that each CPU core is still virtually two threads). This is why our computer is considered as a *one-device parallel system* - each virtual CPU is one parallel processor.

Concurrency vs Parallelism

Concurrency and parallelism are two terms that we commonly encounter when tackling parallelization. They are often used interchangeably but here are the key differences.



(a) Single Core. (b) Multi Core.

Figure 1: Task execution in single and multi core systems.

As illustrated in Figure 1, concurrency refers to interleaved execution of multiple tasks. This is usually demonstrated in single processor systems wherein tasks are progressing and executed partially until all tasks are completed. On the other hand, parallelism refers to execution of multiple tasks at the same time. Each task is executed by one processor.

Concurrency may be considered as *dealing many things at once* and **parallelism** as *doing many things at once*.

COLLEGE OF ARTS AND SCIENCES
UNIVERSITY OF THE PHILIPPINES LOS BAÑOS
1/F Wing C Physical Sciences Bldg., Harold Cuzner Royal Palm Ave.
College 4031, Laguna, Philippines
Phone +63 49 536 2313
ics.uplb@up.edu.ph www.ics.uplb.edu.ph
INSTITUTE OF COMPUTER SCIENCE

CMSC 180: Introduction to Parallel Computing

Suggested Learning Activities

1. Enumerate three examples of concurrent tasks you do at home.
2. Enumerate three examples of parallel tasks you do at home.
3. Find the total number of your logical processors.
4. Search the current maximum number of logical processors in a consumer processor.
5. Search the current maximum number of logical processors in a server processor.
6. Find the supercomputer with the greatest number of processors.

Questions?

If you have any questions or comments regarding this material, please contact your instructor.

CMSC 180: Introduction to Parallel Computing

References

[1] G. G. Abraham Silberschatz Peter B. Galvin, Operating system concepts, 8th ed. Wiley, 2008.

[2] D. Marr et al., "Hyper-Threading Technology Architecture and Microarchitecture." vol. 6, Art. no. 1, 2002.

Online References in Core Affinity

- C/C++: Set Affinity to process thread: https://bytefreaks.net/programming-2/c/cc-set-affinity-to-process-thread-example-code
- C++11 Threads, Affinity, and Hyperthreading: https://dzone.com/articles/c11-threads-affinity-and-hyperthreading-1
- Python | os.sched_setaffinity() method: https://www.geeksforgeeks.org/python-os-sched_setaffinity-method/
- Sys::CpuAffinity - Set CPU affinity for processes: https://metacpan.org/pod/Sys::CpuAffinity
- Processor affinity: https://en.wikipedia.org/wiki/Processor_affinity
- Limiting execution to certain CPUs: https://www.gnu.org/software/libc/manual/html_node/CPU-Affinity.html

Other notes:

- You may set the core affinity of a POSIX thread(pthreads in C/C++) using the function pthread_setaffinity_np .
- You may use the multiprocessing library in Python to automatically assign/allocate CPUs to your tasks.