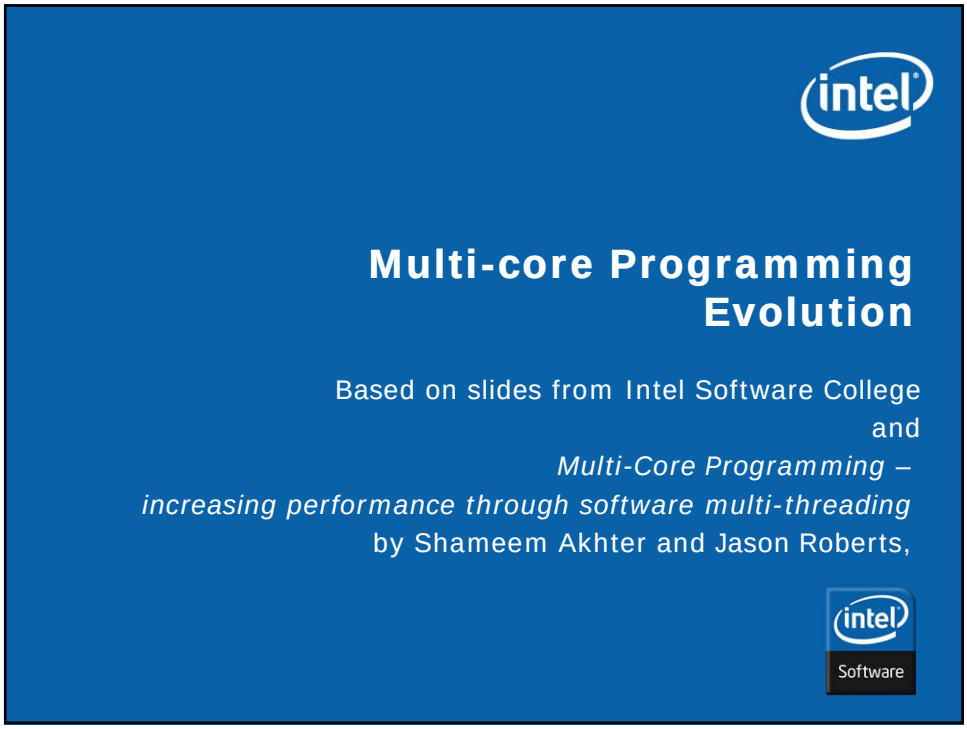


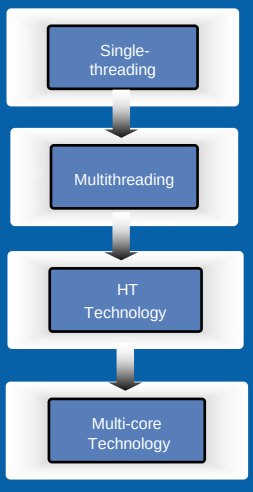
Evolution of Multi-Core Technology

Single-threading:
Only one task processes at one time.

Multitasking and Multithreading:
Two or more tasks execute at one time by using content switching (Functionality)).

HT Technology:
Two single threads execute simultaneously on the same processor core.

Multi-core Technology:
Computational work of an application is divided and spread over multiple execution cores (Performance).



Evolution of Multi-Core Technology

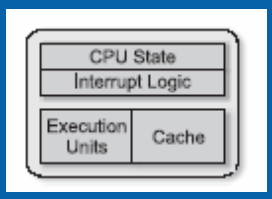
2008/1/17 2

Formal Definition of Threads, Processors, and Hyper-Threading in Terms of Resources

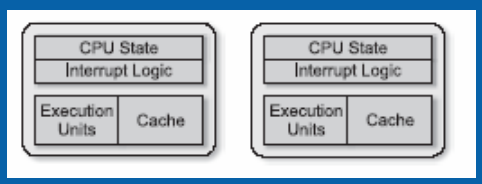
- Thread
  - Basic unit of CPU utilization
  - Program counter, CPU state information, stack
- Processor
  - Architecture state: general purpose CPU registers, interrupt controller registers, caches, buses, execution units, branch prediction logic
- Logical Processor
  - Duplicates the architecture space of processor
  - Execution can be shared among different processors
- HyperThreading
  - Intel version of Simultaneous Multi-Threading (SMT)
  - Makes single physical processor appear as multiple logical processors
  - OS can schedule to logical processors

HyperThreading and Multicore

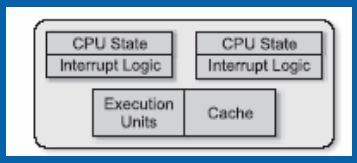
- HyperThreading
  - Instructions from logical processors are persistent and execute on shared execution resources
  - Up to microarchitecture how and when to interleave the instructions
  - When one thread stalls, another can proceed
    - This includes cache misses, and branch mispredictions
- Multicore
  - Chip multiprocessing
    - 2 or more execution cores in a single processor
    - 2 or more processors on a single die
    - Can share an on-chip cache
    - Can be combined with SMT



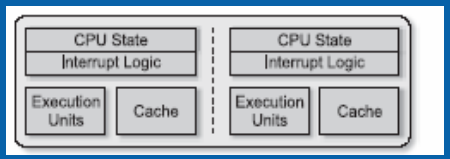
- Single Processor



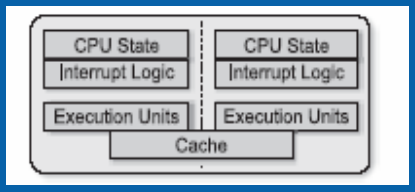
- Multiprocessor



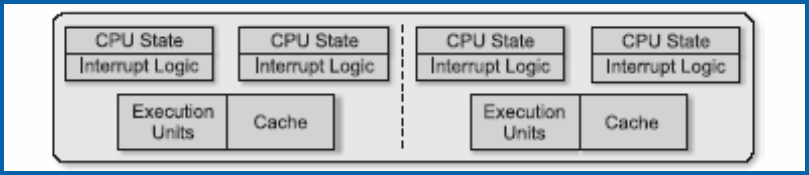
- Hyper-Threading



- Multi-core Processor



- Multicore Shared Cache



- Multicore with HypeThreading Technology

2008/1/17 7

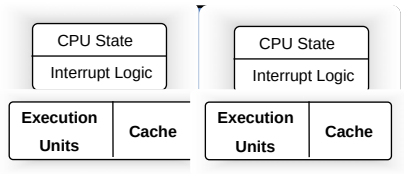
Defining Multi-Core Technology

With Multi-core technology:

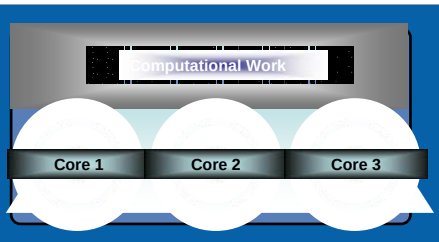
- Two or more complete computational engines are placed in a single processor.

- Computers split the computational work of a threaded application and spread it over multiple execution cores.

- More tasks get completed in less time increasing the performance and responsiveness of the system.



Processor with Multi-Core Technology



Computational Work Distributed Between Multiple Cores

2008/1/17 8

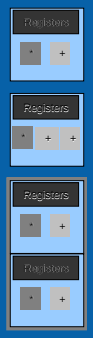
Evolution to Dual Core

Basic Processor
Pipelined Execution
1 Instr. / Cycle

Evolution to Dual Core

Superscalar
Parallel Functional Units

Evolution to Dual Core



Dual Core

Double the transistors

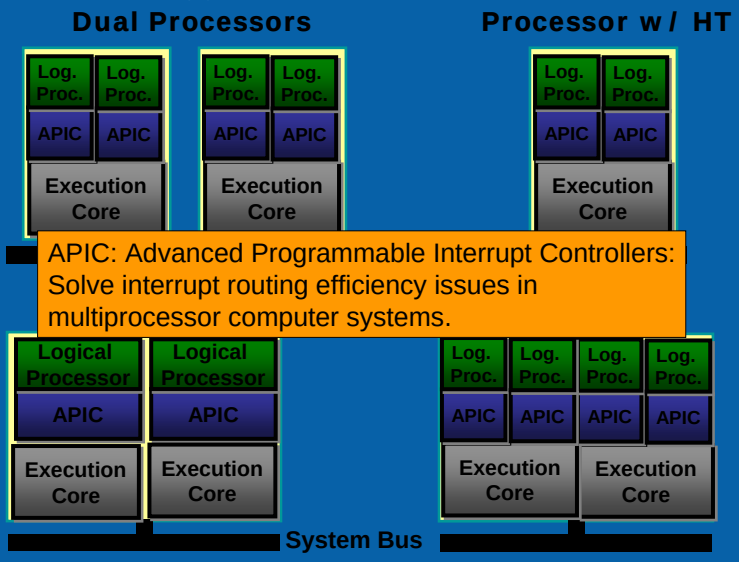
Scope

Instruction Level – encoding/pipelined execution – increase throughput

Instruction Level – scheduling, superscalar

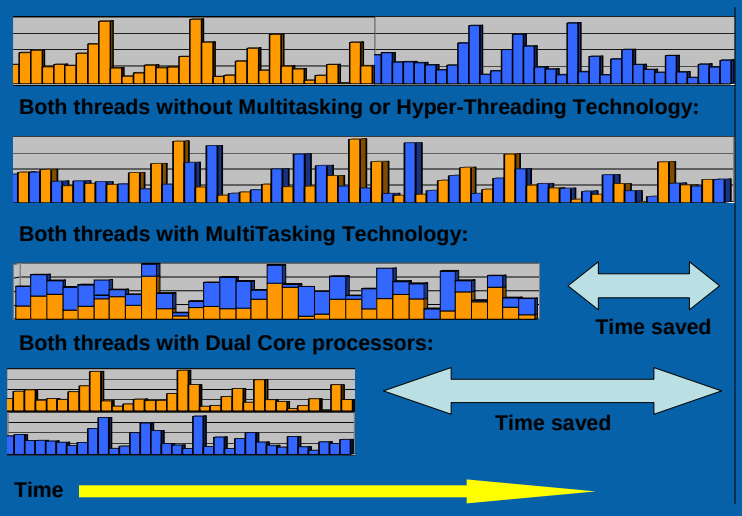
Outermost Loops, whole program – data/functional

HT Technology and Dual-Core



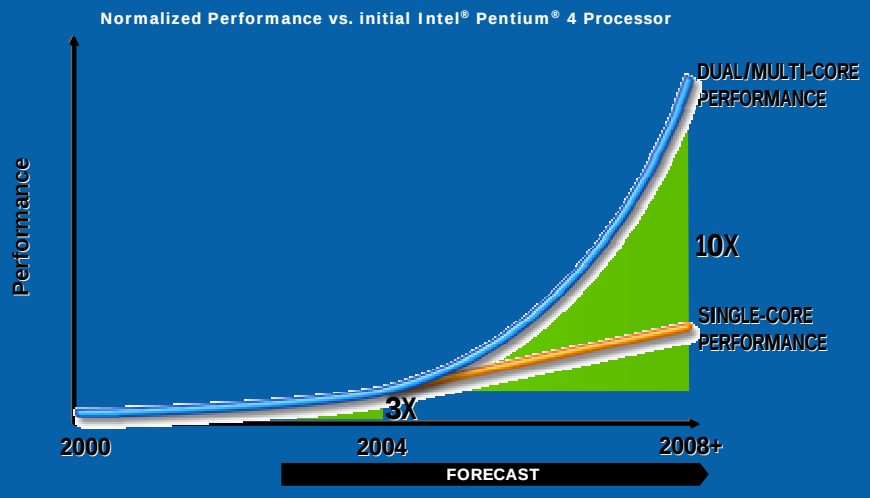
2008/1/17 13

Hyper-Threading vs. Dual Core



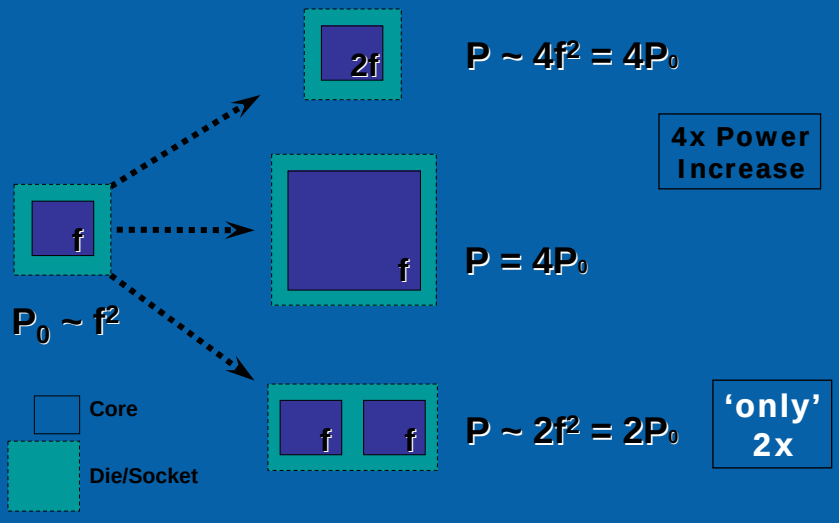
2008/1/17 14

Driving Greater Parallelism



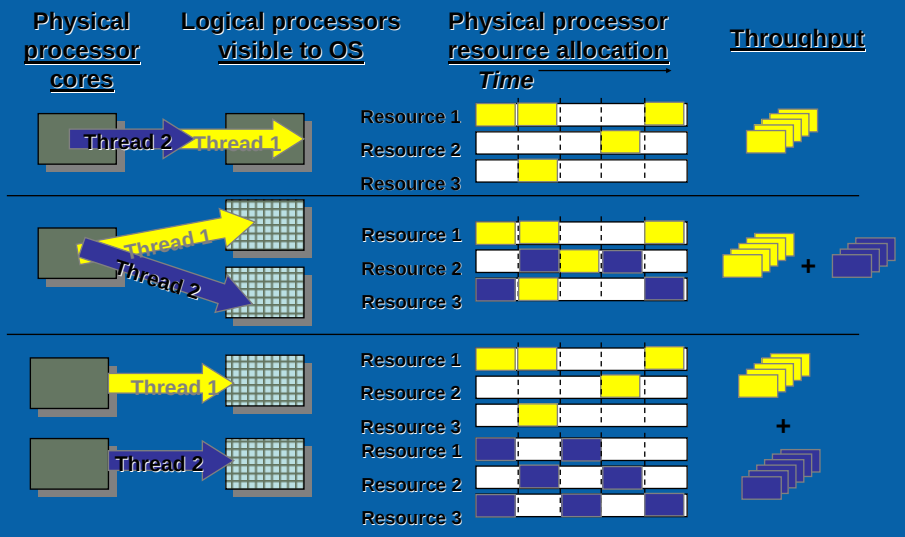
2008/1/17 15

Power Vs. Frequency



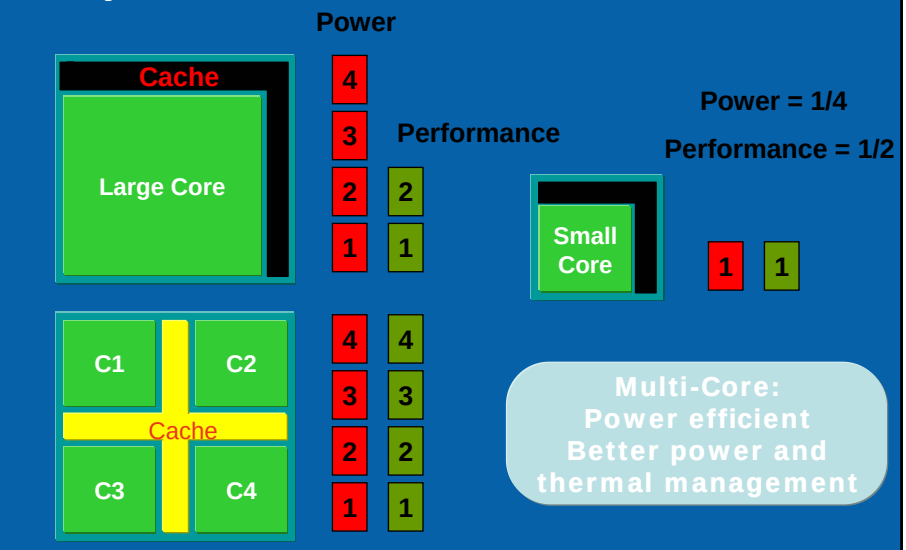
2008/1/17 16

How HT Technology and Dual-Core Work



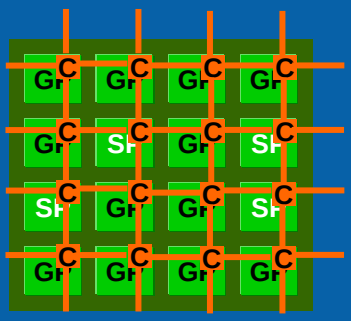
2008/1/17 17

Today? … Multi-Core



2008/1/17 18

The Future?* … Many-core



General Purpose Cores

Special Purpose HW

Interconnect fabric

Heterogeneous Multi-Core Platform

2008/1/17 19

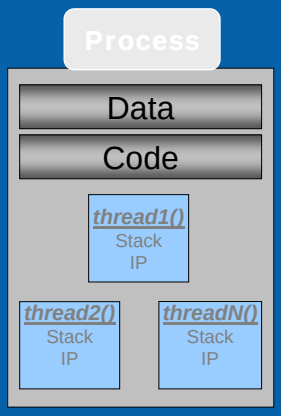
Taking full advantage of Multi-Core requires multi-threaded software

- **Increased responsiveness and worker productivity**
  - Increased application responsiveness when different tasks run in parallel
- **Improved performance in parallel environments**
  - When running computations on multiple processors
- **More computation per cubic foot of data center**
  - Web-based apps are often multi-threaded by nature

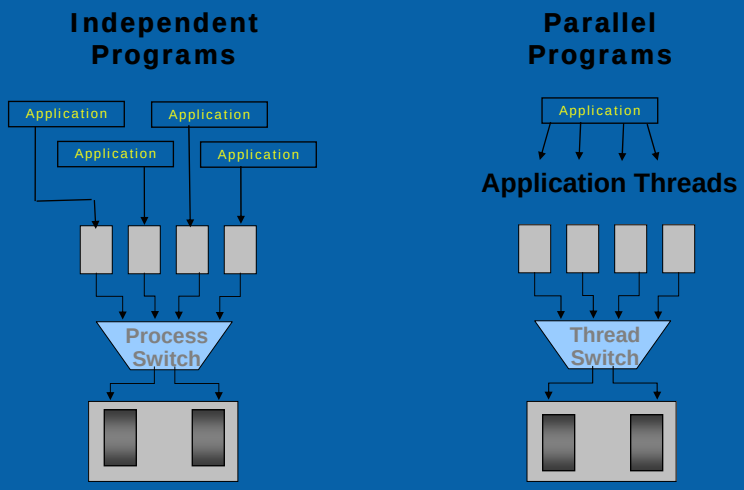
2008/1/17 20

Threading - the most common parallel technique

- OS creates process for each program loaded
- Additional threads can be created within the process
  - Each thread has its own Stack and Instruction Pointer
  - All threads share code and data
  - Single shared address space
  - Thread local storage if required



Contrast with Distributed address parallelism: Message Passing Interface (MPI)



Threads allows one application to utilize the power of multiple processors

Multi-Threading on Single-Core v Multi-Core

- *Optimal application performance on multi-core architectures will be achieved by effectively using threads to partition software workloads*
  - On single core, multi-threading can only be used to hide latency
    - Ex: rather than block UI when printing spawn a thread, free UI thread to respond to user
- *Multi-threaded applications running on multi-core platforms have different design considerations than do multi-threaded applications running on single-core platforms*
  - On single core can make assumptions to simplify writing and debugging
  - These may not be valid on multi-core platforms e.g areas like *memory caching* and *thread priority*

Memory Caching Example

- Each core may have own cache
  - May be out of sync
  - Ex: 2 thread on dual-core reading and writing neighboring memory locations
    - Data values may be in same cache line because of data locality
    - Memory system may mark the line as invalid due to write
    - Result is cache miss even though the data in cache being read is valid
  - Not an issue on single-core since only one cache

Thread Priorities Example

- Applications with two threads of different priorities
  - To improve performance developer assumes that higher thread will run without interference from lower priority thread
  - On single core valid since scheduler will not yield CPU to lower priority thread
  - On multi-core may schedule 2 threads on different cores, and they may run simultaneously making code unstable