CSE 4733/6733 - Operating System 1

Stephen A. Torri, Ph.D.

Mississippi State University

CSE 4733/6733 - Operating System 1

Stephen A. Torri, Ph.D.

Mississippi State University

Introduction[1]

- Operating System started with scheduling processes with a single core.
- Higher-speed CPUs generate too much heat.
- Most computer systems use one CPU.
  So how can a computer run faster

Introduction[1]
- Operating System started with scheduling processes with a single core.
- Higher-speed CPUs generate too much heat.
- Most computer systems use one CPU.
  So how can a computer run faster

Single Core Limitations

Single Core Limitations

Barriers to faster CPUs

I have a need. A need for speed.

There are three barriers to speed:

- Memory bandwidth between CPU and system memory.
- Instruction level parallelism
- Power demand and heat

Time vs. Cost vs. Quality

FAST and CHEAP, but it won't be good quality
CHEAP and GOOD QUALITY, but it won't be quick or on time
ON TIME and GOOD QUALITY, but it cannot be CHEAP

Barriers to faster CPUs
I have a need. A need for speed.
There are three barriers to speed:
Memory bandwidth between CPU and system memory.
Instruction level parallelism
Power demand and heat
Time vs. Cost vs. Quality
FAST and CHEAP, but it won't be good quality
CHEAP and GOOD QUALITY, but it won't be quick or on time
ON TIME and GOOD QUALITY, but it cannot be CHEAP

1. [1]: Instruction level parallelism (ILP) is the availability of enough discrete parallel instructions for a multi-core chip
2. [1]: The limit of power consumption of computers cause chip manufacturers to switch from single-core architecture to multiple-core architecture.
3. [1]: The way to conquer the power demand and heat problem is to focus on reducing the watts per instruction while speed is limited by heat.
4. [1]: If power demand goes down, then the core speed can go up to maintain the same heat load

How to go faster?

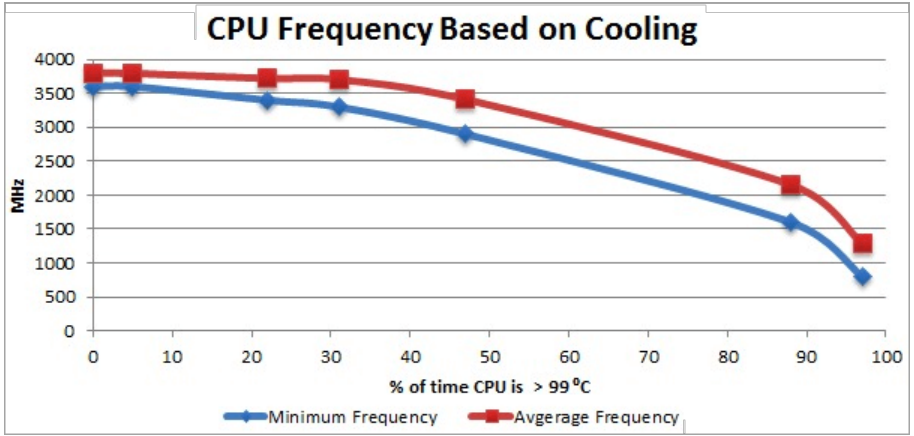
Clock Frequency vs. Throughput vs Memory Bandwidth

- CLOCK and THROUGHPUT, the CPU will run extremely hot.
- CLOCK and BANDWIDTH, the motherboard uses more power and costs more
- BANDWIDTH and THROUGHPUT, CPU cannot consume memory fast enough.

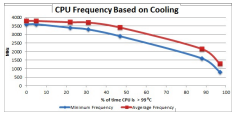
How to go faster?
Clock Frequency vs. Throughput vs Memory Bandwidth
CLOCK and THROUGHPUT, the CPU will run extremely hot.
CLOCK and BANDWIDTH, the motherboard uses more power and costs more
BANDWIDTH and THROUGHPUT, CPU cannot consume memory fast enough.

1. All of these issues assume a single CPU with a single core. What happens if we add more CPUs? Motherboards are more complex, and upgrades to better CPUs cost more.
2. So if a CPU core can handle a set number of instructions, then a CPU can scale by adding cores instead of making them run faster.

CPU Frequency Based on Cooling[2]

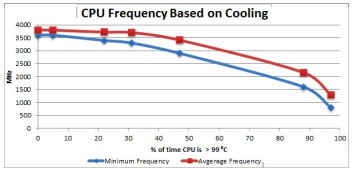


CPU Frequency Based on Cooling[2]



1. As the CPU load increases, more heat is generated.
2. The biggest limitation of air-cooled systems is thermal performance.

Air Cooling Limitations[3]



- As speed increases, CPUs will require forced convection (e.g., bigger heat sinks and larger fans).
- The biggest limitation to air-cooling is that air does not have the same ability as liquid cooling.
- Liquid cooling is more expensive, complicated, and requires more maintenance.

If one will not do it, then add more.

Air Cooling Limitations[3]
As speed increases, CPUs will require forced convection (e.g., bigger heat sinks and larger fans).
The biggest limitation to air-cooling is that air does not have the same ability as liquid cooling.
Liquid cooling is more expensive, complicated, and requires more maintenance.
If one will not do it, then add more.

1. As the CPU load increases, more heat is generated.
2. The biggest limitation of air-cooled systems is thermal performance.

Multi-core Scheduling

Multi-Core/CPU Scheduling

The Rise of the Threads

- Software engineers will need to rewrite software to run in parallel using threads.
- Multiple threads allow for better CPU utilization by spreading the work across multiple cores and CPUs.
- Threads began as an operating system construct to add concurrency to applications.
- Recent advances in chip design now have threads on the CPU that is a virtual version of a CPU core.

Multi-Core/CPU Scheduling
The Rise of the Threads
Software engineers will need to rewrite software to run in parallel using threads.
Multiple threads allow for better CPU utilization by spreading the work across multiple cores and CPUs.
Threads began as an operating system construct to add concurrency to applications.
Recent advances in chip design now have threads on the CPU that is a virtual version of a CPU core.

Multi-Core Scheduling

Multiplle CPU with Cache

Caches are an additional tool to make CPUs faster.

- A hardware cache helps CPUs run faster by keeping recently used data close.
- Spatial locality: instructions near the current instruction are probably next.
- Temporal locality: instructions just executed have a high chance of execution again.
- Testing and code reviews are essential to developing resilient applications.

Multi-Core Scheduling
Multiplle CPU with Cache
Caches are an additional tool to make CPUs faster.
A hardware cache helps CPUs run faster by keeping recently used data close.
Spatial locality: instructions near the current instruction are probably next.
Temporal locality: instructions just executed have a high chance of execution again.
Testing and code reviews are essential to developing resilient applications.

Multi-Core Scheduling

Synchronization: how to make everyone play nice.

At some point, multiple threads will require access to the same memory. This can cause problems.

- Race condition
- Deadlock

Multi-Core Scheduling
Synchronization: how to make everyone play nice.
At some point, multiple threads will require access to the same memory. This can cause problems.
Race condition
Deadlock

1. Race Condition: An undesirable situation occurs when a device or system attempts to perform two or more operations simultaneously.
2. Deadlock: Situation in which more than one process is blocked because it is holding a resource and also requires some resource that is acquired by some other process.[4]

Necessary conditions to create deadlock[4]

Four conditions must exist for a deadlock to occur:

- Mutual Exclusion: Only one process can use a resource at any time.
- Hold and wait: A process is holding at least one resource at a time and is waiting to acquire other resources held by some other process.
- No preemption: The resource can be released by a process voluntarily.
- Circular Wait: A set of processes are circularly waiting for each other.

Necessary conditions to create deadlock[4]
Four conditions must exist for a deadlock to occur:
- Mutual Exclusion: Only one process can use a resource at any time.
- Hold and wait: A process is holding at least one resource at a time and is waiting to acquire other resources held by some other process.
- No preemption: The resource can be released by a process voluntarily.
- Circular Wait: A set of processes are circularly waiting for each other.

1. Mutual Exclusion: the resources are non-sharable.
2. No preemption: after execution of the process

Scheduling Algorithms

Scheduling Algorithms

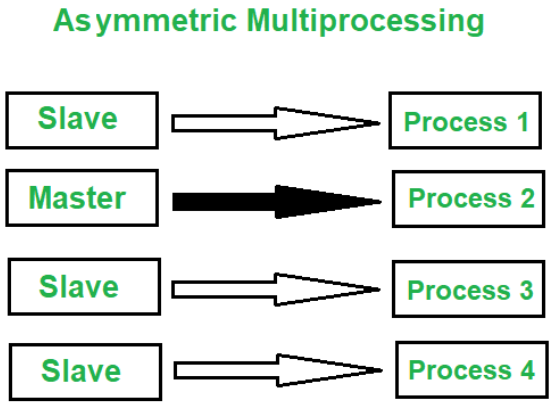
Symmetric versus Asymmetric MP

Asymmetric Multiprocessing

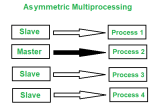
An Asymmetric Multiprocessing system is a multiprocessor computer system where not all interconnected central processing units (CPUs) are treated equally. Only a master processor runs the tasks of the operating system.

Symmetric versus Asymmetric MP
Asymmetric Multiprocessing
An Asymmetric Multiprocessing system is a multiprocessor computer system where not all interconnected central processing units (CPUs) are treated equally. Only a master processor runs the tasks of the operating system.

Asymmetric Multiprocessing Diagram



Asymmetric Multiprocessing Diagram



Symmetric Multiprocessing

Symmetric Multiprocessing

It involves a multiprocessor computer hardware and software architecture where two or more identical processors are connected to a single, shared main memory and have full access to all input and output devices. In other words, Symmetric Multiprocessing is a type of multiprocessing where each processor is self-scheduling.

Symmetric Multiprocessing
Symmetric Multiprocessing
It involves a multiprocessor computer hardware and software architecture where two or more identical processors are connected to a single, shared main memory and have full access to all input and output devices. In other words, Symmetric Multiprocessing is a type of multiprocessing where each processor is self-scheduling.

Asymmetric MP versus Symmetric MP

No.	Asymmetric MP	Symmetric MP
1.	In asymmetric multiprocessing, the processors are not treated equally.	In symmetric multiprocessing, all the processors are treated equally.
2.	The master processor does tasks of the operating system.	Individual processors do tasks of the operating system.
3.	No Communication between Processors as the master processor controls them.	All processors communicate with another processor by shared memory.
4.	Process scheduling approach used is master-slave	the process is taken from the ready queue.

Asymmetric MP versus Symmetric MP
No. Asymmetric MP Symmetric MP
1. In asymmetric multiprocessing, the processors are not treated equally. In symmetric multiprocessing, all the processors are treated equally.
2. The master processor does tasks of the operating system. Individual processors do tasks of the operating system.
3. No Communication between Processors as the master processor controls them. All processors communicate with another processor by shared memory.
4. Process scheduling approach used is master-slave the process is taken from the ready queue.

Asymmetric MP versus Symmetric MP

No.	Asymmetric MP	Symmetric MP
5.	Asymmetric MP systems are cheaper.	Symmetric multiprocessing systems are costlier.
6.	Asymmetric multiprocessing systems are easier to design.	Symmetric multiprocessing systems are complex to design.
7.	All processors can exhibit different architecture.	The architecture of each processor is the same.
8.	It is simple; here, the master processor can access the data, etc.	It is complex as the processors require synchronization to maintain the load balance.

Asymmetric MP versus Symmetric MP

No.	Asymmetric MP	Symmetric MP
5.	Asymmetric MP systems are cheaper.	Symmetric multiprocessing systems are costlier.
6.	Asymmetric multiprocessing systems are easier to design.	Symmetric multiprocessing systems are complex to design.
7.	All processors can exhibit different architecture.	The architecture of each processor is the same.
8.	It is simple; here, the master processor can access the data, etc.	It is complex as the processors require synchronization to maintain the load balance.

Asymmetric MP versus Symmetric MP

No.	Asymmetric MP	Symmetric MP
9.	If a master processor malfunctions, the slave processor continues the execution, which is turned to the master processor. When a slave processor fails, then other processors take over the task.	In case of processor failure, there is a reduction in the system's computing capacity.
10.	It is suitable for homogeneous or heterogeneous cores.	It is suitable for homogeneous cores.

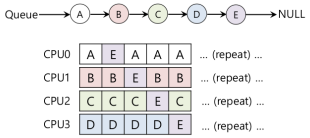
Asymmetric MP versus Symmetric MP

No.	Asymmetric MP	Symmetric MP
9.	If a master processor malfunctions, the slave processor continues the execution, which is turned to the master processor. When a slave processor fails, then other processors take over the task.	In case of processor failure, there is a reduction in the system's computing capacity.
10.	It is suitable for homogeneous or heterogeneous cores.	It is suitable for homogeneous cores.

Process Affinity

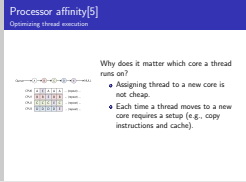
Processor affinity[5]

Optimizing thread execution



Why does it matter which core a thread runs on?

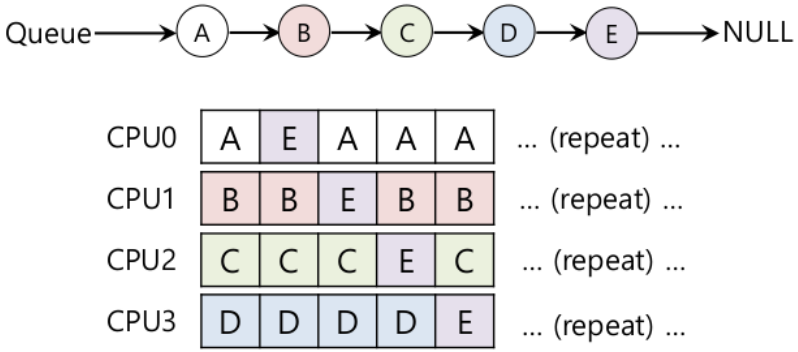
- Assigning thread to a new core is not cheap.
- Each time a thread moves to a new core requires a setup (e.g., copy instructions and cache).



1. A preemptive multitasking operating system consistently reschedules jobs on a multiple-core processor for optimal system performance.
2. The core to which a given thread or process is assigned can differ each time. Each time a thread is assigned to a core, the processor must copy a thread's relevant data and instructions into its cache if the data is not already in the cache.

Processor affinity[5]

Processor affinity image



Processor affinity[5]
Processor affinity image

Processor affinity[5]

Optimizing thread execution

Processor affinity

Processor affinity or CPU pinning enables applications to bind or unbind a process or a thread to a specific core or a range of cores or CPUs. The operating system ensures that a given thread executes only on the assigned core(s) or CPU(s) each time it is scheduled if it was pinned to a core.

Processor affinity[5]
Optimizing thread execution
Processor affinity
Processor affinity or CPU pinning enables applications to bind or unbind a process or a thread to a specific core or a range of cores or CPUs. The operating system ensures that a given thread executes only on the assigned core(s) or CPU(s) each time it is scheduled if it was pinned to a core.

1. Processor affinity takes advantage of the fact that the remnants of a process execution remain valid when the same process or thread is scheduled a second time on the same processor
2. the cache may still be useful when a thread execution is rescheduled after being preempted. This helps scale the performance on multiple-core processor architectures that share the same global memory and have local caches (UMA Architecture).

Load Balancing[6]

Load Balancing

Load balancing attempts to keep the workload evenly distributed across all processors and cores in a system.

- Load balancing is typically only necessary on systems where each processor has a private queue of eligible processes to execute.
- On systems with a common run queue, load balancing is often unnecessary because once a processor becomes idle, it immediately extracts a runnable process from the common run queue.

Load Balancing[6]
Load Balancing
Load balancing attempts to keep the workload evenly distributed across all processors and cores in a system.
- Load balancing is typically only necessary on systems where each processor has a private queue of eligible processes to execute.
- On systems with a common run queue, load balancing is often unnecessary because once a processor becomes idle, it immediately extracts a runnable process from the common run queue.

1. It is important to keep the workload balanced among all processors and cores to utilize the CPU fully. Otherwise, one or more processors may sit idle while other processors have high workloads along with lists of processes awaiting the CPU.

Load Balancing[6]

Push Migration vs. Pull Migration

There are two general approaches to load balancing: push migration and pull migration.

- **Push migration**: a specific task periodically checks the load on each processor and if it finds an imbalance evenly distributes the load by moving (or pushing) processes from overloaded to idle or less-busy processors.
- **Pull migration** occurs when an idle processor pulls a waiting task from a busy processor.

Load Balancing[6]
Push Migration vs. Pull Migration
There are two general approaches to load balancing: push migration and pull migration.
- **Push migration**: a specific task periodically checks the load on each processor and if it finds an imbalance evenly distributes the load by moving (or pushing) processes from overloaded to idle or less-busy processors.
- **Pull migration** occurs when an idle processor pulls a waiting task from a busy processor.

1. Push and pull migration need not be mutually exclusive and are often implemented in parallel on load-balancing systems.

Single-queue Scheduling

- Example:

- Possible job scheduler across CPUs:

CPU0	A	E	D	C	B	... (repeat) ...
CPU1	B	A	E	D	C	... (repeat) ...
CPU2	C	B	A	E	D	... (repeat) ...
CPU3	D	C	B	A	E	... (repeat) ...

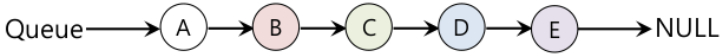
Features:

- Put all jobs that need to be scheduled into a single queue.
- Requires synchronization for multiple cores to pick jobs from the queue.

Single-queue Scheduling
Features:
- Put all jobs that need to be scheduled into a single queue.
- Requires synchronization for multiple cores to pick jobs from the queue.

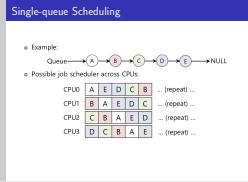
Single-queue Scheduling

- Example:

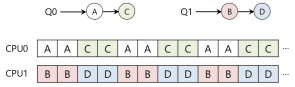


- Possible job scheduler across CPUs:

CPU0	A	E	D	C	B	... (repeat) ...
CPU1	B	A	E	D	C	... (repeat) ...
CPU2	C	B	A	E	D	... (repeat) ...
CPU3	D	C	B	A	E	... (repeat) ...



Multi-queue Scheduling[7]

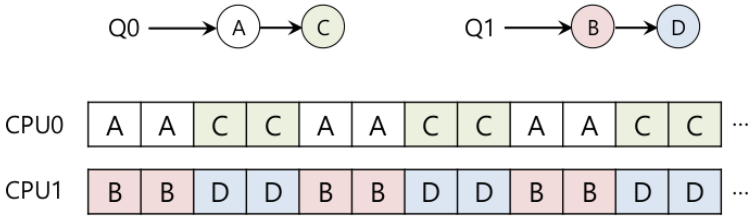


Multiple scheduling properties:

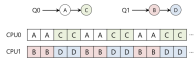
- Each queue will follow a particular scheduling discipline.
- When a job enters the system, it is placed on exactly one scheduling queue.
- Avoid the problems of information sharing and synchronization.

Multi-queue Scheduling[7]
Multiple scheduling properties:
- Each queue will follow a particular scheduling discipline.
- When a job enters the system, it is placed on exactly one scheduling queue.
- Avoid the problems of information sharing and synchronization.

Multi-queue Scheduling[7]



Multi-queue Scheduling[7]



References I

C. Mims.
Why cpus aren't getting any faster, 2010.
https://www.technologyreview.com/2010/10/12/199966/why-cpus-arent-getting-any-faster/.

Software Testing Help.
11 best cpu temp monitor tools: Pc temperature monitor 2023, 2022.
https://www.softwaretestinghelp.com/top-cpu-temp-monitor/.

BOYD.
Air vs. liquid cooling: Advancements in thermal management for power electronics, 2023.
https://www.boydcorp.com/resources/resource-center/technical-papers/air-vs-liquid-cooling-advancements-in-thermal-management.html.

I. Yadav.
Deadlock in operating system, 2022.
https://www.scaler.com/topics/operating-system/deadlock-in-os/.

Intel.
1.3.1 process affinity or cpu pinning, 2023.
https://www.intel.com/content/www/us/en/docs/programmable/683013/current/processor-affinity-or-cpu-pinning.html.

C. Ozdogan.
Load balancing, 2011.
http://boron.physics.metu.edu.tr/ozdogan/OperatingSystems/ceng328/node130.html.

Unknown.
Lecture notes from g667 at university of cantabria - multiprocessor scheduling (advanced), 2022.
https://ceunican.github.io/aos/10.Multiprocessor_Scheduling(Advaned).pdf.