

Serial vs Parallel vs Distributed

Architectures ~ threads

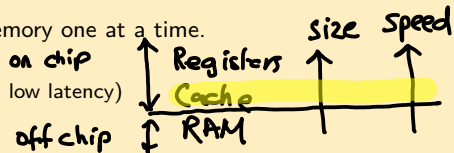


(programming)

Serial Computing Systems

Sequential.
Serial

- Single control mechanism, determines the next instruction to be executed.
- All data operations are fetched from memory one at a time.
- Speedup can be introduced using:
 - Instruction Caching (large caches with low latency)
 - Pipelining (super-scalar architectures)
 - Overclocking
 - Overlapping of I/O and computation using Direct Memory Access (DMA)

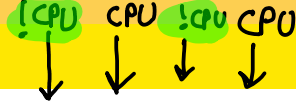


- **Note:** Two serial programs can execute concurrently (multi-tasking).
- **Multi-Processing:** Multiple CPU cores executing more than 1 program at a given time
- **Multi-Tasking:** A single CPU core “appearing” to be executing more than 1 program at a given time
- **Multi-Threading:** The presence of more than 1 “threads” in a single program.

Serial vs Parallel vs Distributed

- 50%

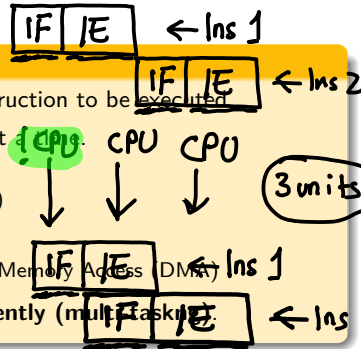
4 units



Serial Computing Systems

- Single control mechanism, determines the next instruction to be executed
- All data operations are fetched from memory one at a time
- Speedup can be introduced using:
 - Instruction Caching (large caches with low latency)
 - Pipelining (super-scalar architectures)
 - Overclocking
 - Overlapping of I/O and computation using Direct Memory Access (DMA)

- **Note:** Two serial programs can execute concurrently (multi-tasking).



3 units

- **Multi-Processing:** Multiple CPU cores executing more than 1 program at a given time
- **Multi-Tasking:** A single CPU core “appearing” to be executing more than 1 program at a given time
- **Multi-Threading:** The presence of more than 1 “threads” in a single program.

66%

Serial vs Parallel vs Distributed



Serial Computing Systems

 $4 \times 10^8 \text{ Hz}$
 $4 \times 10^8 \text{ Flop/s.}$

- Single control mechanism, determines the next instruction to be executed.
- All data operations are fetched from memory one at a time.
- Speedup can be introduced using:

H.W {
 AW {
 • Instruction Caching (large caches with low latency)
 • Pipelining (super-scalar architectures)
 • Overclocking
 • Overlapping of I/O and computation using Direct Memory Access (DMA)

→ 4 GHz
 → Clock Frequency.

- Note: Two serial programs can execute concurrently (multi-tasking).



H.W / dual core / core 2 duo /

- Multi-Processing: Multiple CPU cores executing more than 1 program at a given time
 FPU₁ → FPU₂ for each core → dedicated Logic unit
 " " " f. point unit
- Multi-Tasking: A single CPU core "appearing" to be executing more than 1 program at a given time
- Multi-Threading: The presence of more than 1 "threads" in a single program.



Serial vs Parallel vs Distributed



bulbs

60 Hz

eye not noticeable
recording - visible.

Serial Computing Systems

- Single control mechanism, determines the next instruction to be executed.
- All data operations are fetched from memory one at a time.
- Speedup can be introduced using:
 - Instruction Caching (large caches with low latency)
 - Pipelining (super-scalar architectures)
 - Overclocking
 - Overlapping of I/O and computation using Direct Memory Access (DMA)

$$\frac{1}{4 \times 10^{-6}} = 4,000,000$$

- **Note:** Two serial programs can execute concurrently (multi-tasking)

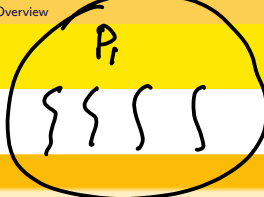
- **Multi-Processing:** Multiple CPU cores executing more than 1 program at a given time
- **Multi-Tasking:** A single CPU core "appearing" to be executing more than 1 program at a given time \rightarrow OS - (time slice)
- **Multi-Threading:** The presence of more than 1 "threads" in a single program.

$$4 \times 10^{-6}$$

$$4 \mu s \text{ (ms)}$$

Serial vs Parallel vs Distributed

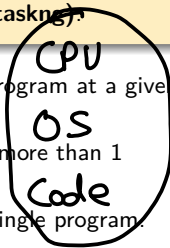
multiple threads



Serial Computing Systems

- Single control mechanism, determines the next instruction to be executed.
- All data operations are fetched from memory one at a time.
- Speedup can be introduced using:
 - Instruction Caching (large caches with low latency)
 - Pipelining (super-scalar architectures)
 - Overclocking
 - Overlapping of I/O and computation using Direct Memory Access (DMA)
- **Note: Two serial programs can execute concurrently (multi-tasking).**

- **Multi-Processing:** Multiple CPU cores executing more than 1 program at a given time
- **Multi-Tasking:** A single CPU core “appearing” to be executing more than 1 program at a given time
- **Multi-Threading:** The presence of more than 1 “threads” in a single program.



User space

Struct a
{
// members.
}

Waiting

kernel space

1:1

1:1

Waiting.

hardware space

1:1

1:1

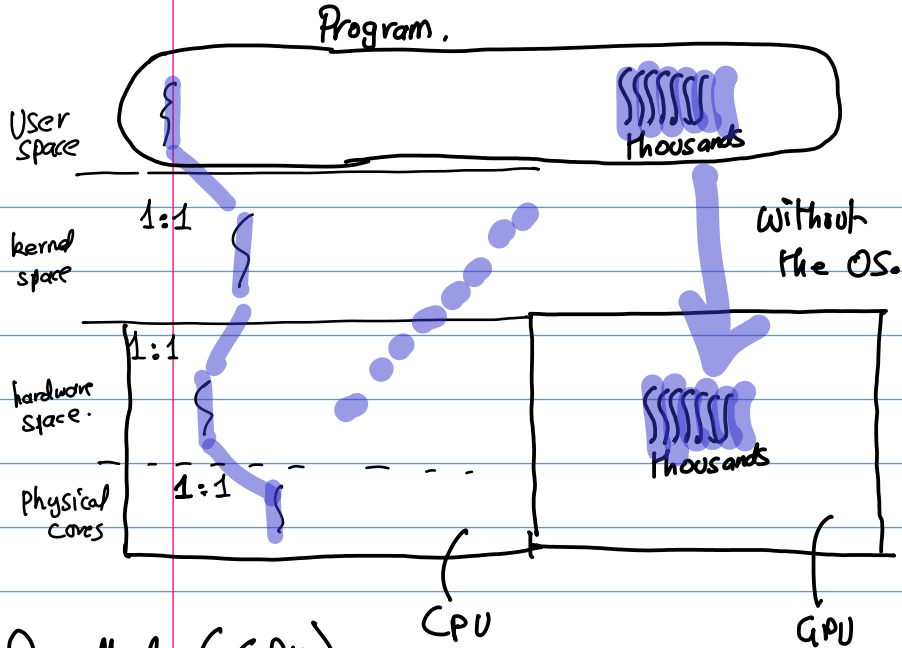
Waiting.

Physical cores

M:1

[SMP]
Parallel (CPU)

M:N
1:1
M:1
1:M



Parallel (GPU)

Distributed

Program.

User space

kernel space

hardware space.

Physical cores

1:1

1:1

1:1

CPU

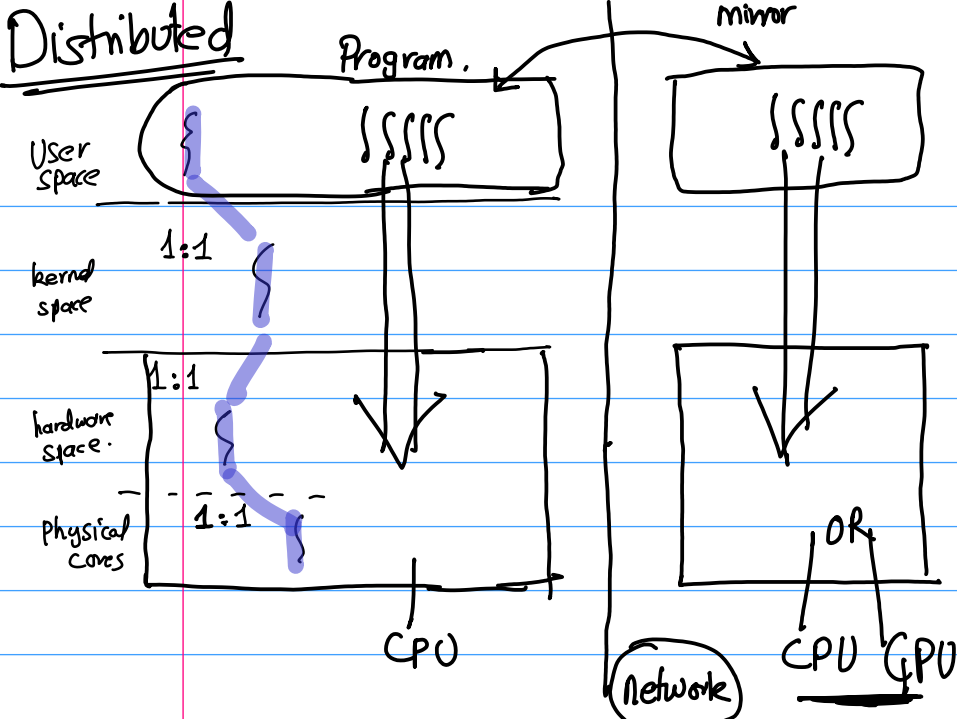
mirror

Network

OR

CPU

GPU



Speedup (OS)

new

Ready

new process

write
read
printf
scanf

syscalls
related
to
I/O

Running

CPU

waiting
I/O

moved
here
due to
I/O sys
calls

OS

I/O	CPU	I/O	CPU
-----	-----	-----	-----

I/O	CPU	
	I/O	CPU

Speedup (compilers support)

L → gcc → code

→ alphabet
-O0
↖ Int
↗ numerical

-O1

-O2

-O3

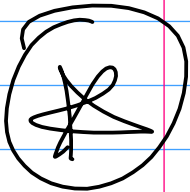
Coding style.

processor
compatible

30_60

⋮

enable certain
comp. optimizations.



Serial vs Parallel vs Distributed (cont.)

Microprocessor

Multiple Cores.

may or may not communicate.

Parallel Computing Systems

- Several processors that are located within a small distance of each other
- Their main purpose is to perform a computation task jointly
- Communication between processors, if required, is reliable, fast, and predictable.

12 = Communications possible

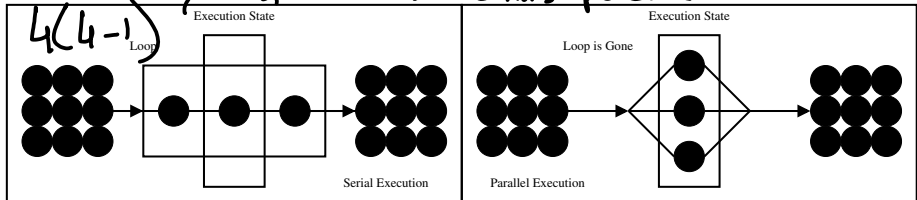
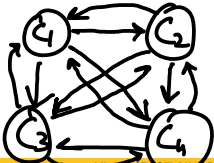


Figure 6: Difference between serial and parallel execution



Complete Graph

$$\frac{n(n-1)}{2}$$

communications possible

$$O(n^2)$$

Serial vs Parallel vs Distributed (cont.)

Parallel Computing Systems

- Several processors that are located within a small distance of each other
- Their main purpose is to perform a computation task jointly
- Communication between processors, if required, is reliable, fast, and predictable.

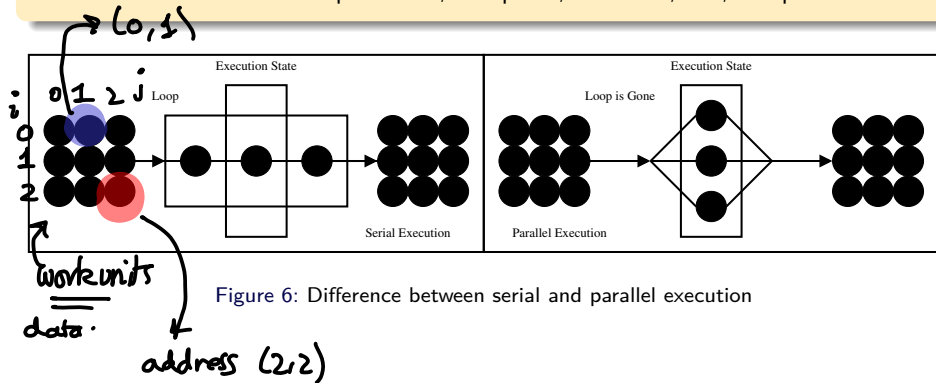


Figure 6: Difference between serial and parallel execution

Serial vs Parallel vs Distributed (cont.)

Parallel Computing Systems

- Several processors that are located within a small distance of each other
- Their main purpose is to perform a computation task jointly
- Communication between processors, if required, is reliable, fast, and predictable

total work
= 9

total thread
= 3

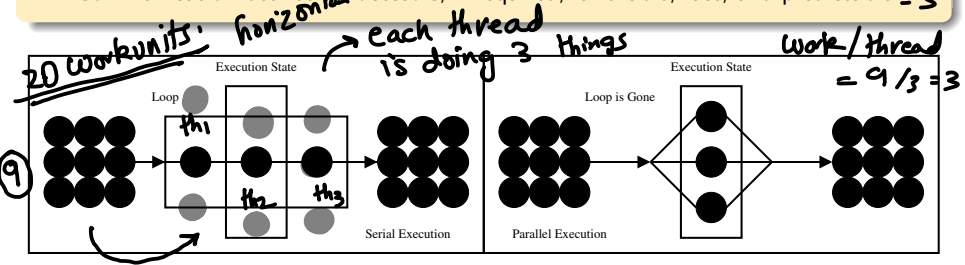


Figure 6: Difference between serial and parallel execution

~~for(i)~~
~~for(j)~~] get rid of one of these loops.

if
a
loop

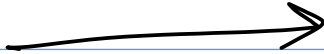
in

prog.

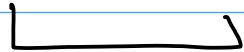


n iterations

10 iterations



Replace
with
threads.



n threads

n/p threads

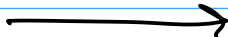
threads

work done

n/p threads			
2	5	10	1
5	2	1	10



parallel
on single
CPU



single loop.

parallel
on gpus
distributed
threads



multiple
loops

Serial vs Parallel vs Distributed (cont.)

Parallel Computing Systems

- Several processors that are located within a small distance of each other
- Their main purpose is to perform a computation task jointly
- Communication between processors, if required, is reliable, fast, and predictable.

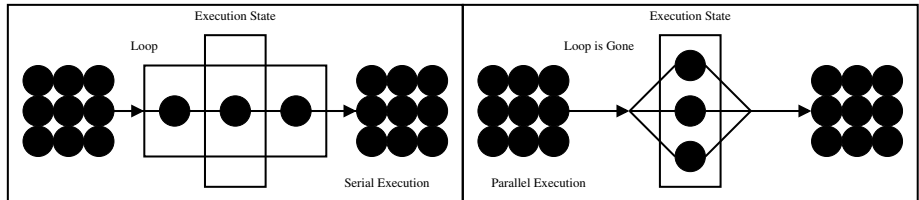


Figure 6: Difference between serial and parallel execution

vertical arrangement

① ① eye communicate
Parallel .

Distributed .
Communication over
a network

→ Avoid'd Communication | min
at all costs

You must engineer | min
data migration.

Communication is on a single chip.