Serial vs Parallel vs Distributed

Architectures ~ threads (programming)

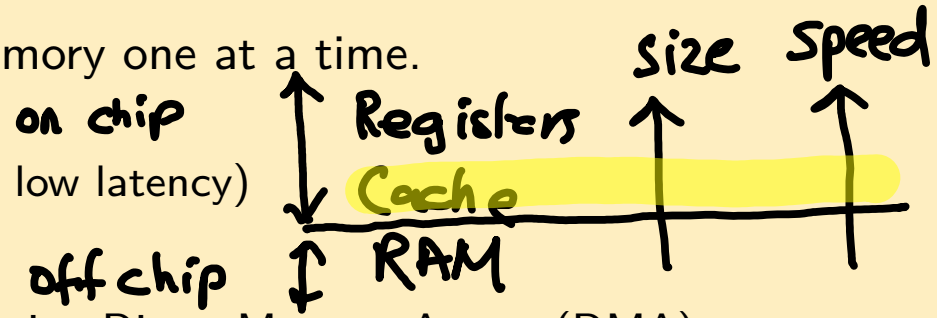
1 CPU

st- end

Sequential, Serial

Serial Computing Systems

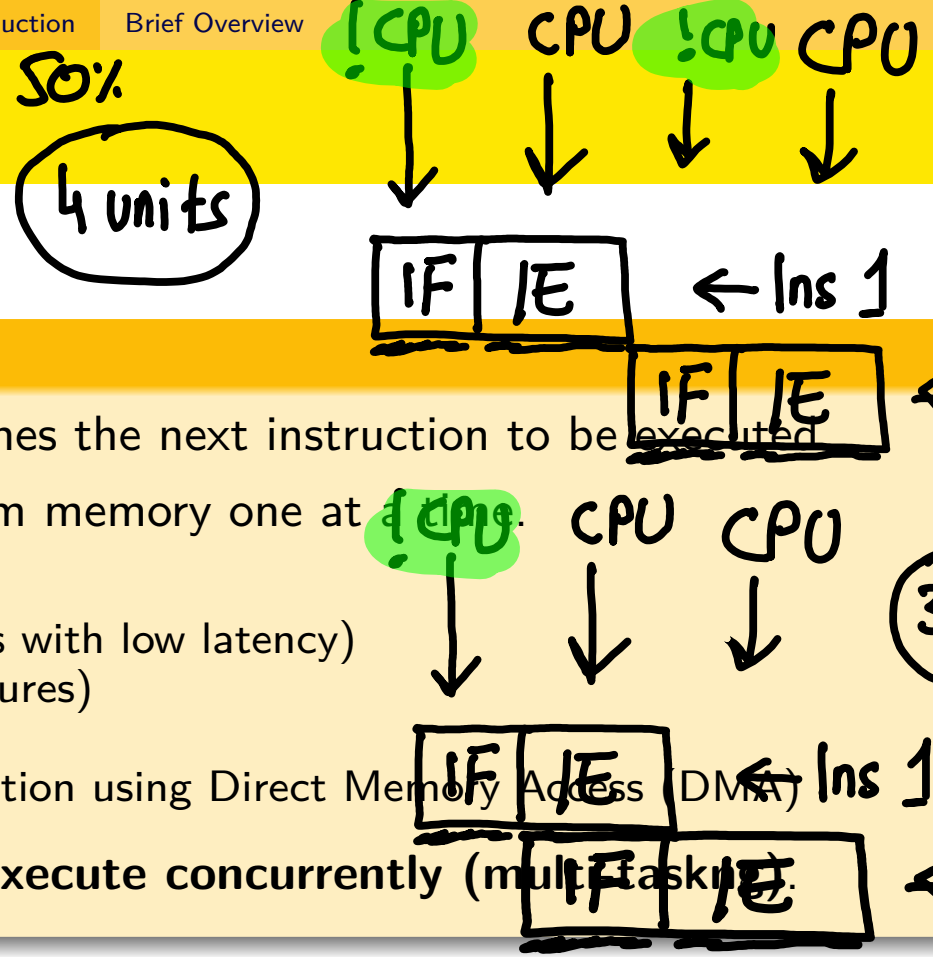
- Single control mechanism, determines the next instruction to be executed.
- All data operations are fetched from memory one at a time.
- Speedup can be introduced using:
  - Instruction Caching (large caches with low latency)
  - Pipelining (super-scalar architectures)
  - Overclocking
  - Overlapping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



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

- **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



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.

Serial vs Parallel vs Distributed

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-taskng).**
- **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

bulbs → 60 Hz
↳ eye noticeable
↳ recording – visible.

Serial Computing Systems

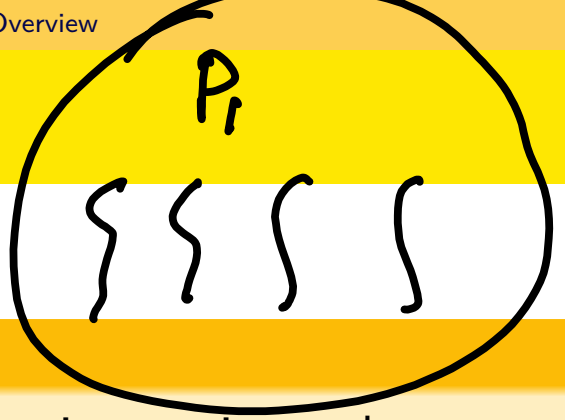
- Single control mechanism, determines the next instruction to be executed.
- All data operations are fetched from memory one at a time.
- Speedup can be introduced using:
  - Instruction Caching (large caches with low latency)
  - Pipelining (super-scalar architectures)
  - Overclocking
  - Overlapping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

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

4×10^{-6}

- **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 → OS – (time slice → 4 ms (ms))
- **Multi-Threading**: The presence of more than 1 "threads" in a single program.

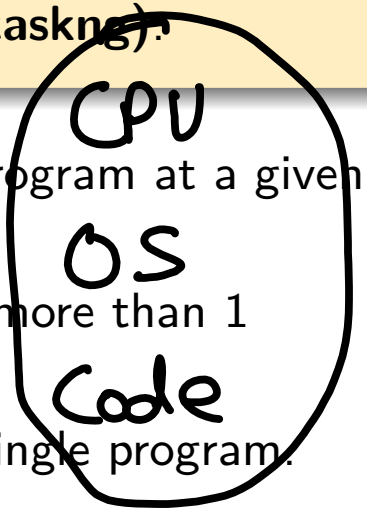
Serial vs Parallel vs Distributed



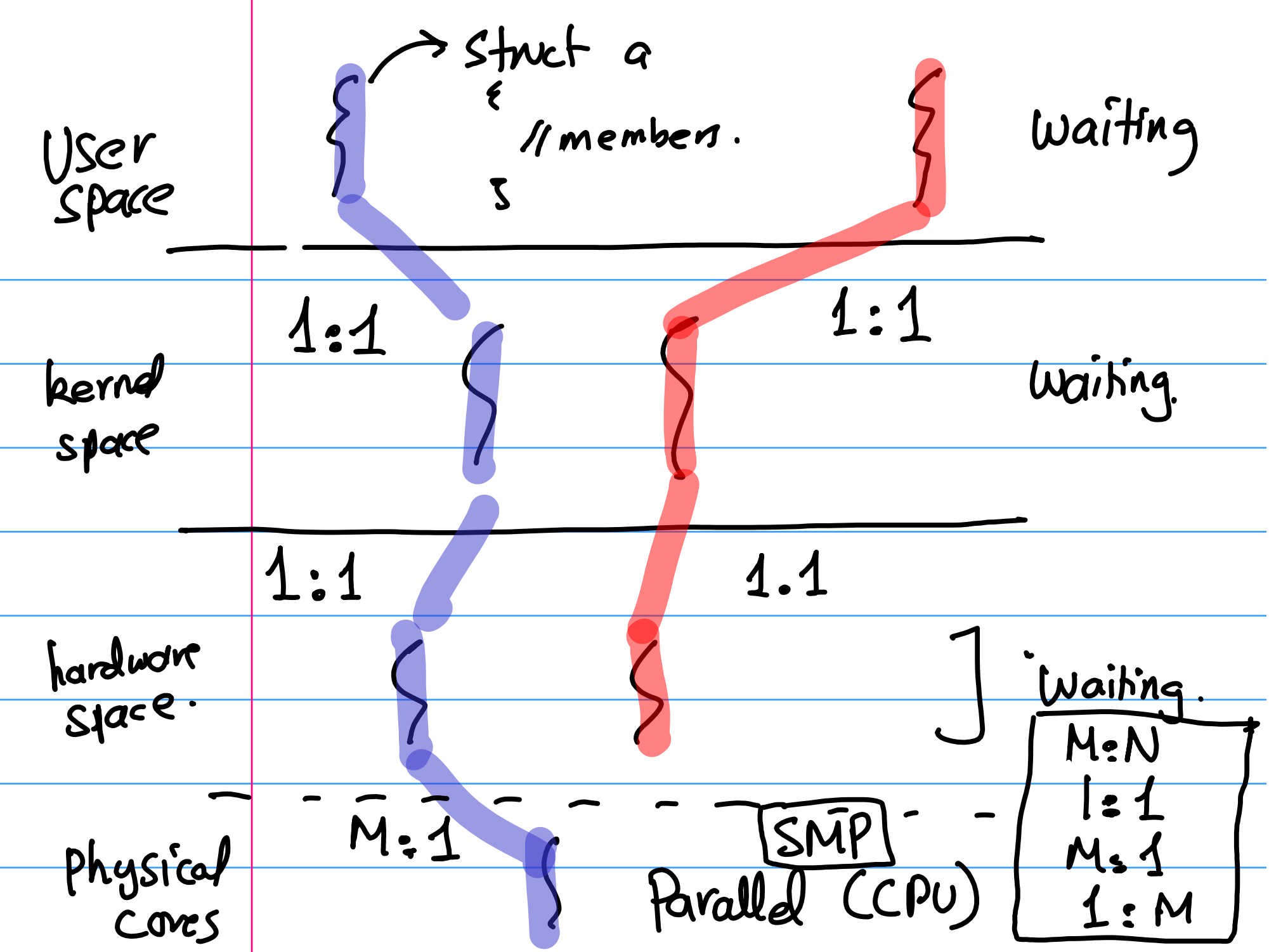
multiple. threads

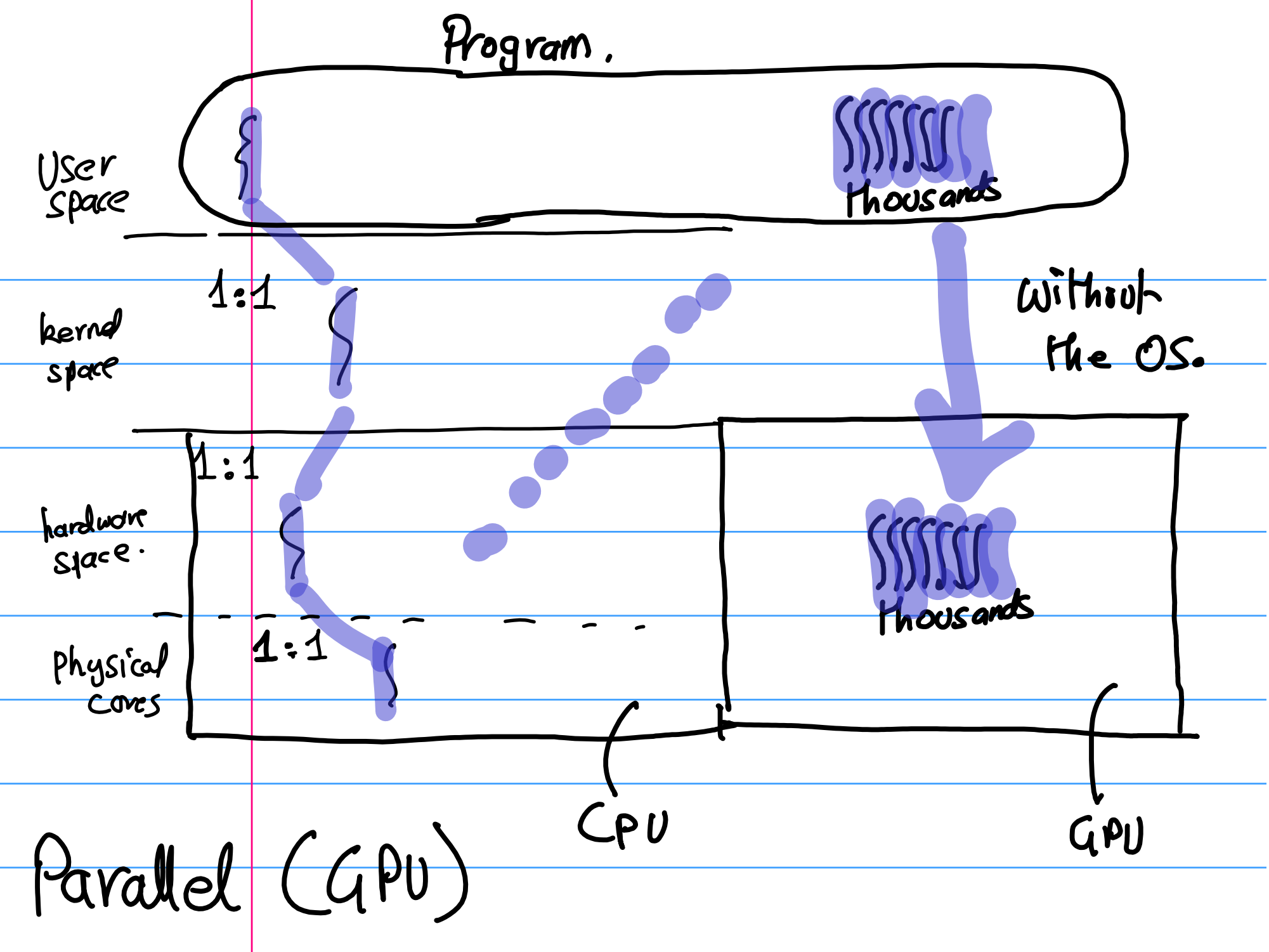
Serial Computing Systems

- 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).**

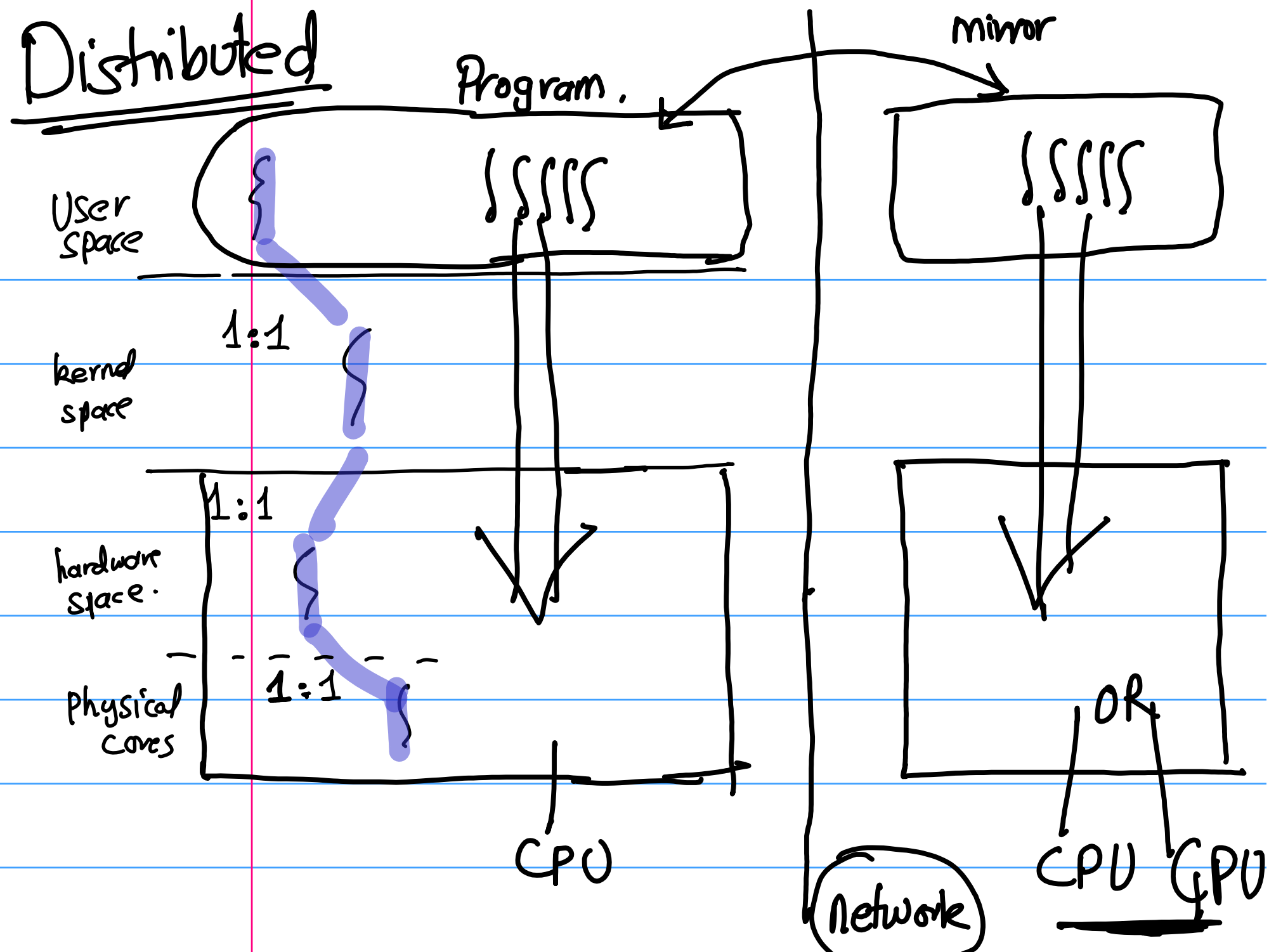


- **Multi-Processing**: Multiple CPU cores executing more than 1 program at a given time
- **Multi-Tasking**: A single CPU core "appearing" to be executing more than 1 program at a given time
- **Multi-Threading**: The presence of more than 1 "threads" in a single program.

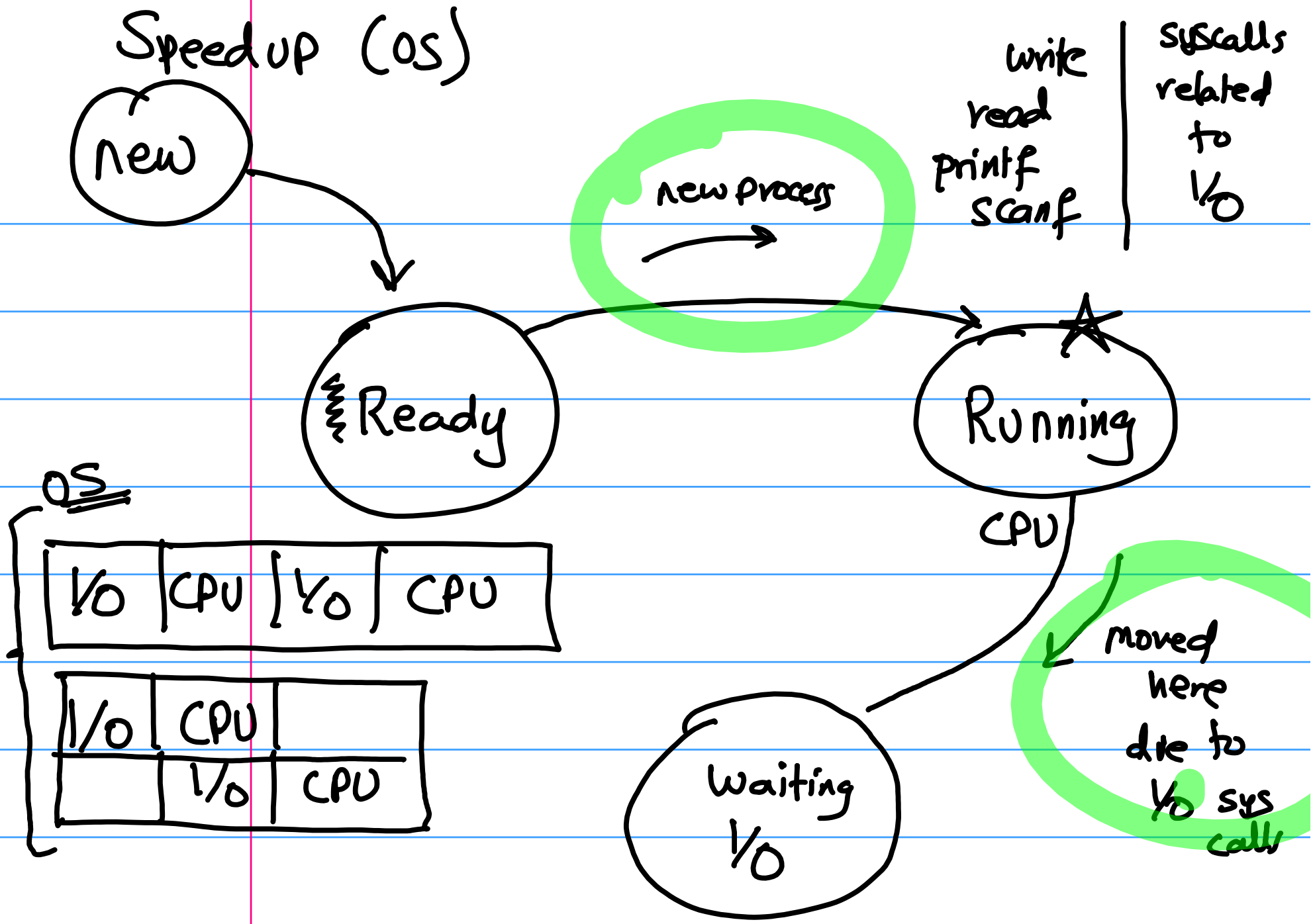

struct a
{
//members.
}
User space
Waiting
1:1
1:1
kernel space
Waiting.
1:1
1.1
hardware space.
Waiting.
M:N
1:1
M:1
1:M
M:1
SMP
Parallel (CPU)
Physical cores



Parallel (GPU)


Distributed
Program.
mirror
User space
kernel space
hardware space.
physical cores
1:1
1:1
1:1
CPU
network
OR
CPU
CPU

Speedup (OS)

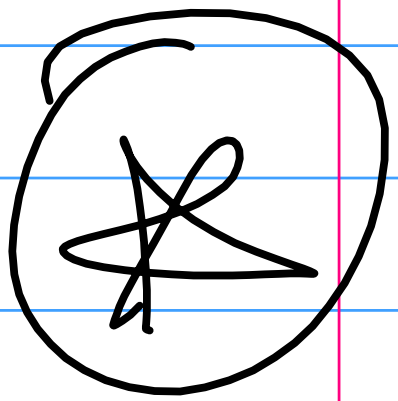

new
Ready
new process
Running
CPU
waiting
I/O
moved here due to I/O sys call
OS
I/O
CPU
I/O
CPU
I/O
CPU
I/O
CPU
write
read
printf
scanf
syscalls related to I/O

Speedup (compilers support)

→ gcc ⌊ code ⌋ -O∅

∅ → alphabet

int numerical



-O1

-O2

-O3

Coding style.

processor compatible

⋮

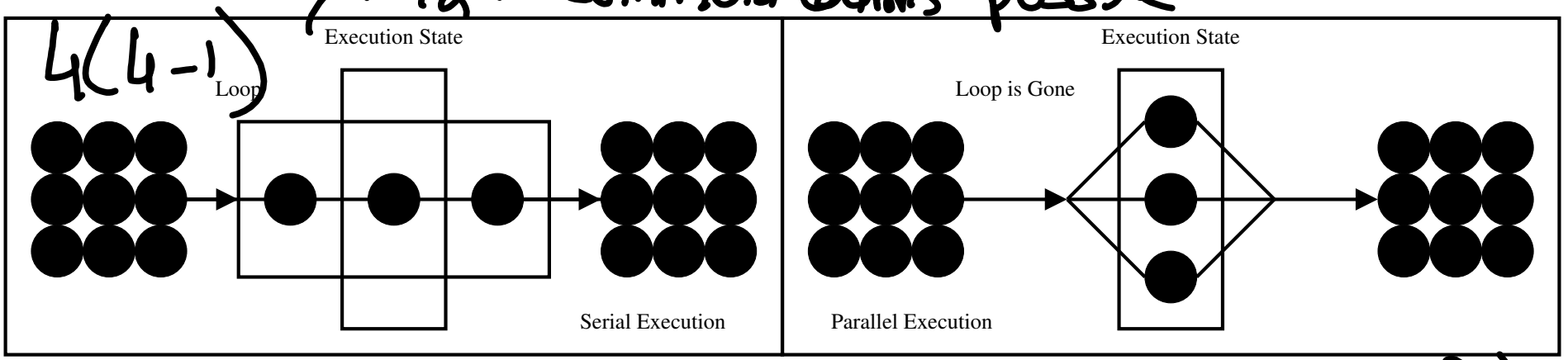
-Ofast

enable certain comp. optimizations.

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.



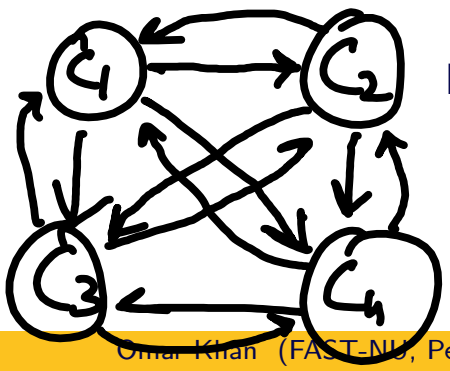


Figure 6: Difference between serial and parallel execution

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.

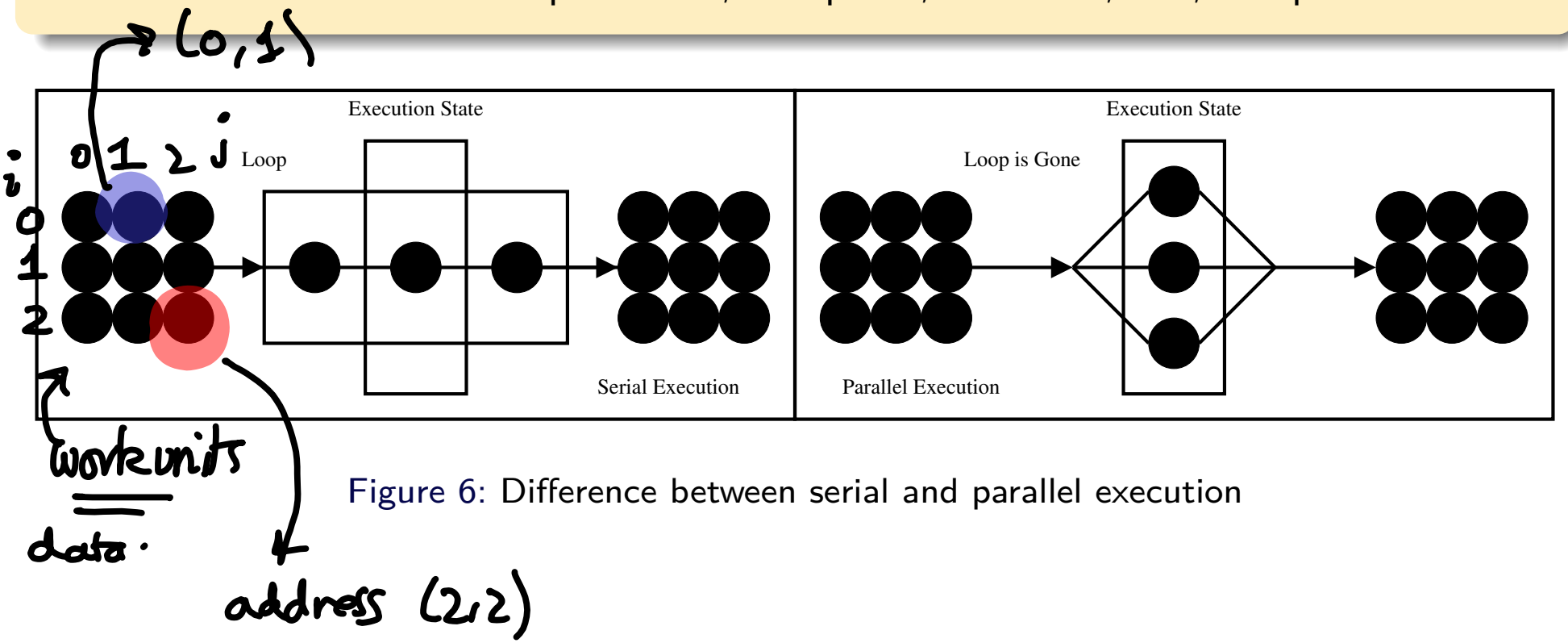


Figure 6: Difference between serial and parallel execution

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

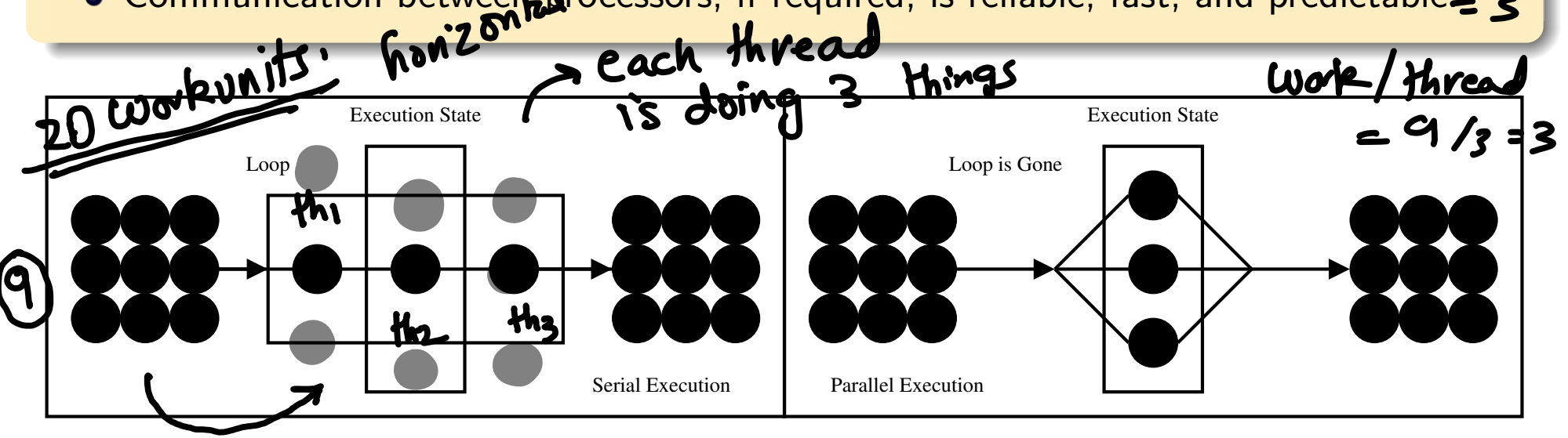


Figure 6: Difference between serial and parallel execution

if a loop in prog.

Replace with threads.

n iterations

10 iterations

n threads

n/p threads

threads	2	5	10	1
work done	5	2	1	10

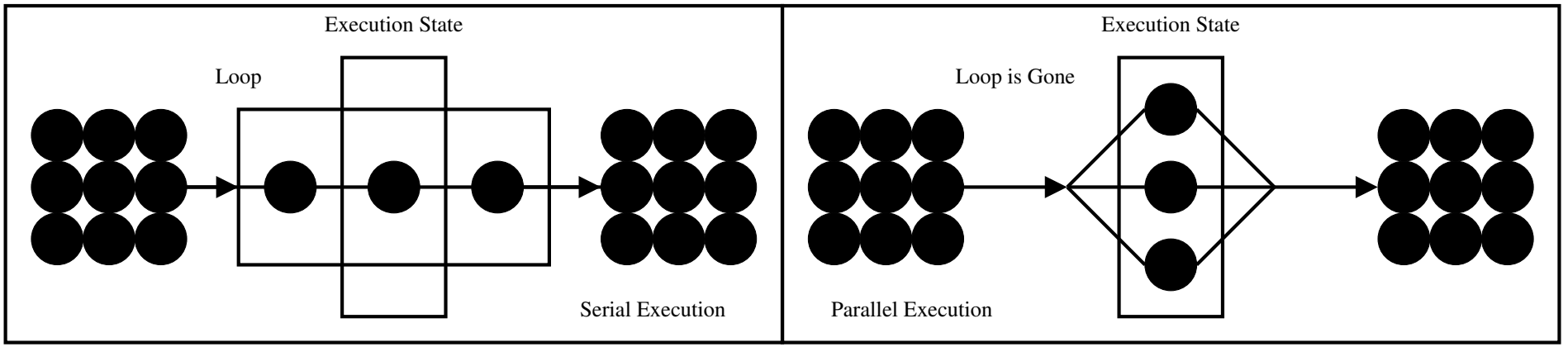
parallel on single CPU → single loop.

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.



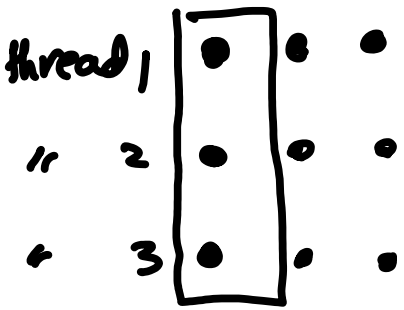


Figure 6: Difference between serial and parallel execution

vertical arrangement

Parallel.

↳ Avoid Communication | Min
at all costs

&

You must engineer | Min
data migration.

Communication is on a single chip.