Concurrency

A

B

start

end

Parallel

C Concurrent

both work now → 2+ process uncle then

Speed

UX

- move now
- see wide
- web bro
dependency
dep. graph

\[ x = a(\ldots) \]
\[ y = b(\ldots) \]
\[ z = c(x, y) \]
\[ w = d(x, z) \]
\[ q = e(z, x) \]

avg do of concurrently
work \[ \frac{23}{\text{len(crit Path)}} \rightarrow 15 \]

bina

Critical Path

Core 1

core 2
Shared memory

\[ \text{comm} = \text{memory} \]

Core_1

\[ x \rightarrow 1 \]
add \$1, x

Race condition

Core_2

\[ x \rightarrow 1 \]
add \$1, x

true

send

receive

buffer
Mutual exclusion (broken)

\[
\text{occupied} = \text{False}
\]

\[
\text{if} \ (\neg \text{occupied}) \Rightarrow \\
\text{occupied} = \text{true} \quad \text{update}
\]

\[
\text{work} \leq \text{race condition}
\]

\[
\text{occupied} = \text{false}
\]

\[
\text{else}
\]

\[
\text{data race}
\]

\[
\text{data (variable)}
\]

\[
\text{read} \quad \text{dep}
\]

\[
\text{control}
\]
$$y = b(\;\)$$
```
syscall b is done
```

$$x = a(\;\)$$
```
syscall a is done
```

---

$c_1$:
```
y = b(\ ... \)
while (a is not done) {
  \text{do nothing}
}
```