Sequence Coordination

`eventcount \rightarrow integer`

`wait(eventcount, value)`

`notify(eventcount)`

`if ec \leq value`

`wait`
Performance
Concurrency
Caching
Scheduling
Performance Metrics

Capacity: Amt of resource
Utilization: % of capacity using
Latency: time for a req. to complete
Throughput: req/sec

size in GBs
instrs/sec
Performance Bottlenecks
I/O Bottleneck

10^8 = 100 MX

111 ms

NET 100 ms
HTML 1 ms
Disk 10 ms
Concurrency

- between modules
- within a module

pipelining

R1 R2 R3

R1 ... Rn

hide latency

tput – 10/sec

0 11ms 111ms 211 311

+R1 -R1,+R2 -R2,+R3
Race Condition

Outcomes – 1) OK

T₁

→page ← buf[first]
→first ← first + 1
→ret. page

T₂

→page ← buf[first]
→first ← first + 1
→return
Isolation Primitives

atomic

isolate $\rightarrow$ locks

lock $\rightarrow$ set, unset

ACQUIRE

RELEASE

RSL – read + set lock

\[
\begin{align*}
\text{held} &= \text{false} \\
\text{while}(!\text{held}) & \\
& \quad \text{held} = \text{RSL}(\text{tl}) \\
\text{end}
\end{align*}
\]

lock $\text{tl}$

\[
\begin{align*}
\text{ACQ}(\text{tl}) & \\
\text{page} & \leftarrow \text{buf}[\text{first}] \\
\text{first} & \leftarrow \text{first} + 1 \\
\text{REL}(\text{tl}) &
\end{align*}
\]
Caching