Parallelism 2

Note Title 12/10/2008

Return marked up proj. 3 to envelope on by door by 12/19 5pm

Performance test case on website tonight.

Course Evals

Multicore is wave of future.

Multi. threading

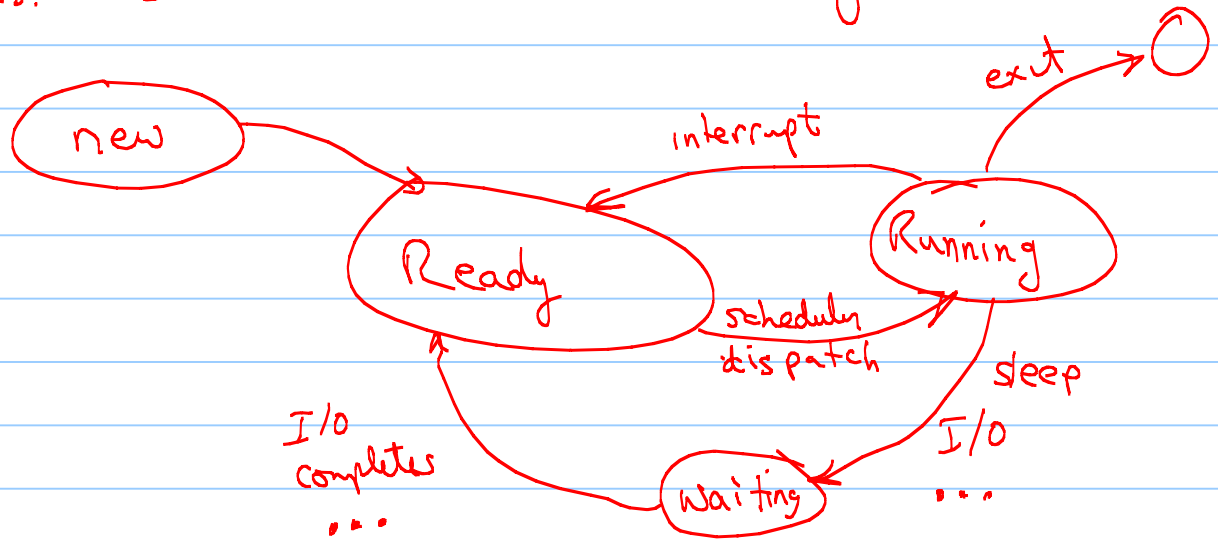
- Goal is to make a single app use multi procs

A Thread (of execution) is a sequential computation w/ its own PC, stack, registers, but not its own memory address space.

Contrast: a process is the same but w/ own A.S.

Idea: each thread can run on a different core.

Many more threads than cores. How does O.S. manage threads. State Machine for scheduling



Threads share address space – can access some locations.

T1	T2
read x	read x
inc x (in private reg)	inc x (in private reg)
write x	write x

x: 10

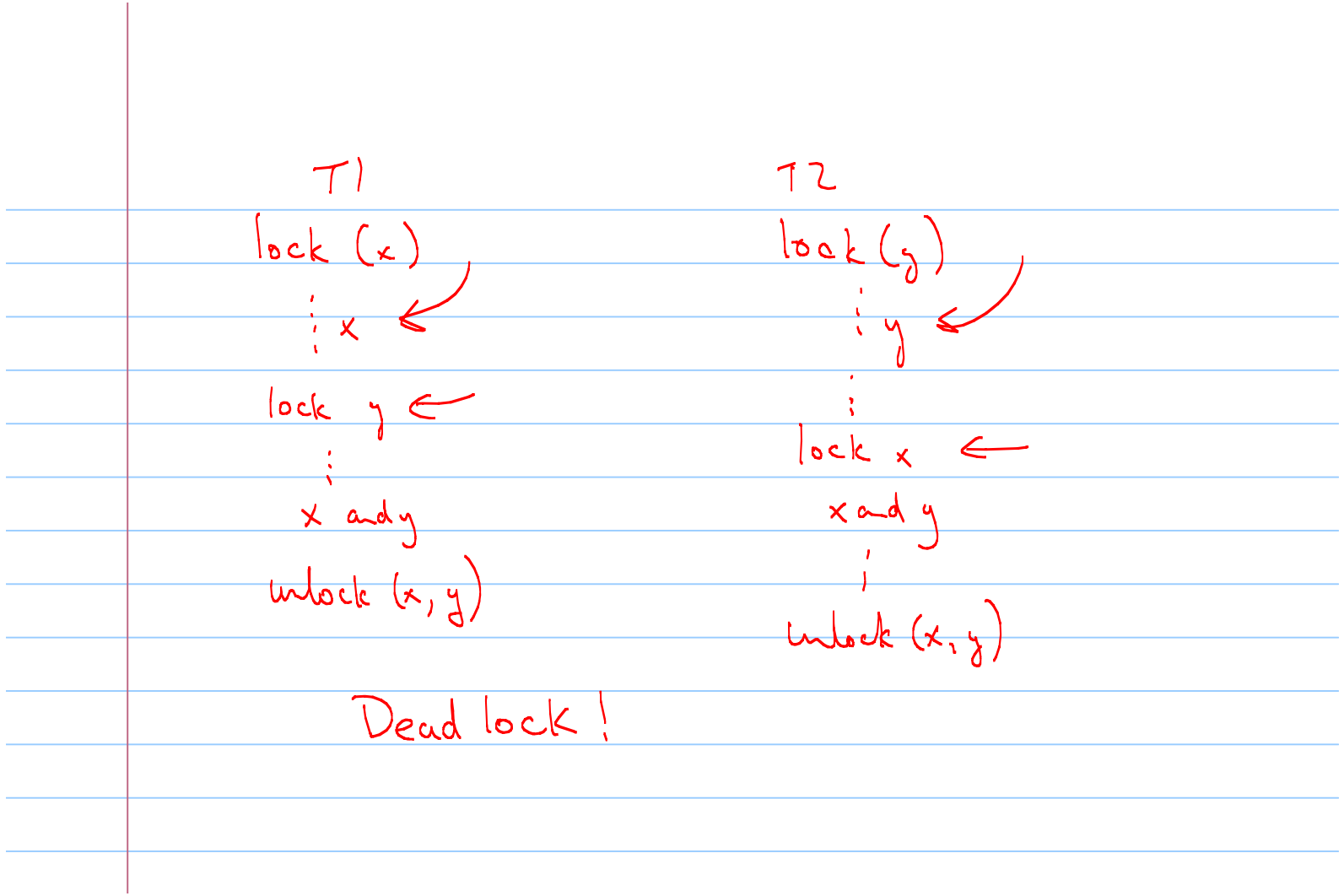
Race-condition.

Programing w/ threads have to use synchronization to prevent race conditions.

A lock (mutual exclusion device, mutex) is a guard on a critical section.

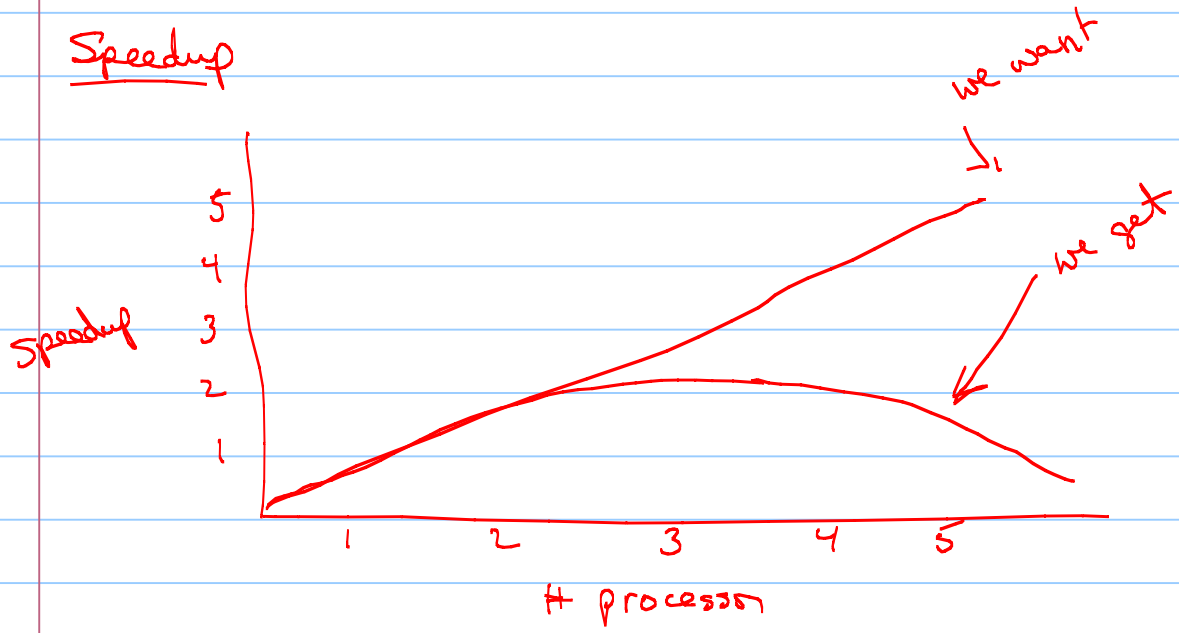
T1	T2
lock (x)	=
read x	=
inc x	=
write x	=
unlock (x)	=

When T2 tries to lock x and T1 has already locked x, T2 goes to wait state.

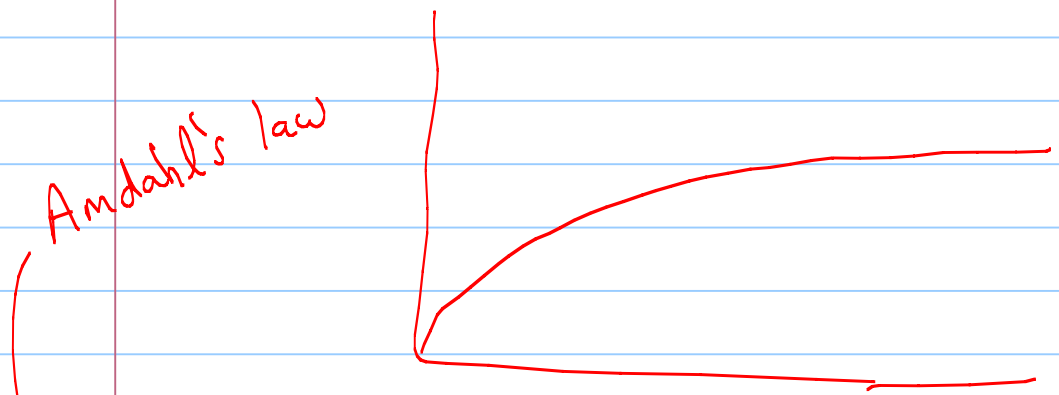

T1
lock (x)
x
lock y
x and y
unlock (x, y)
T2
lock (y)
y
lock x
x and y
unlock (x, y)
Dead lock !

End result: programing w/ threads is not at all like programing sequentially

Speedup



Gene Amdahl :



Every program has some sequential fraction = S

Speedup obtainable w/ P processors =

Time (1 proc) / Time (P procs) ≤

→ 1 / (S + ((1-S)/P)) as P → ∞ : 1/S