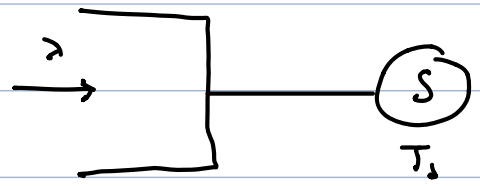
Queuing Theory



$$s = \lambda T_s = \frac{\lambda}{\mu} \quad 0 \le \rho < 1$$

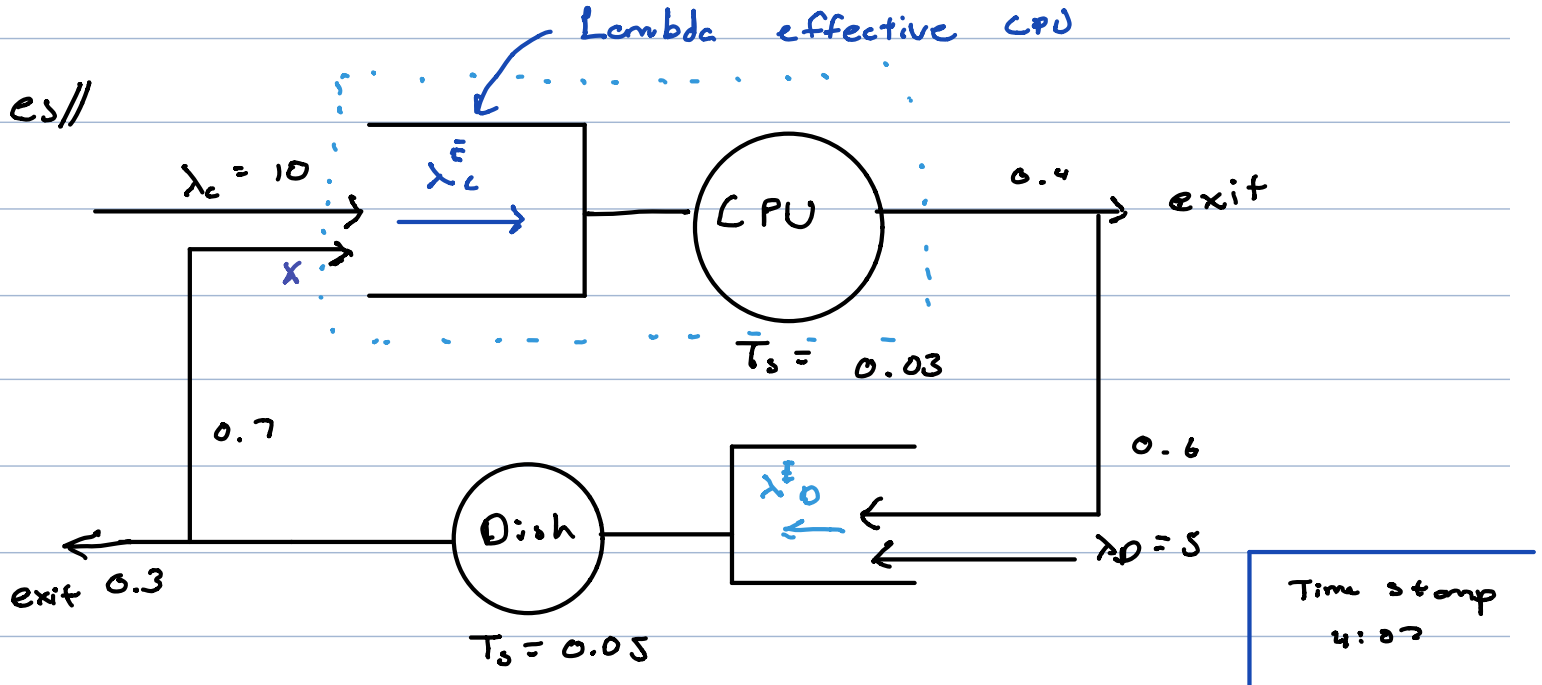
$$q = \lambda T_q$$

$$T_q = T_w + T_s$$

$$w = \lambda T_w$$

$$q = w + s$$

$$q = \frac{\rho}{1-\rho}$$



Time stamp
4:07

1 ρ_{CPU} ρ_{Disk}

$$\lambda_c^E = 10 + X$$

$$\lambda_D^E = 5 + 0.6 * \lambda_c^E$$

$$X = 0.7 \lambda_D^E$$

process / CPU cycle

$$\lambda_c^E = 10 + 0.7 \lambda_D^E$$

$$\lambda_D^E = 5 + 0.6 \lambda_c^E$$

$$\lambda_c^E = 10 + 0.7(5 + 0.6)\lambda_c^E$$

$$\lambda_c^E = 23.2$$

$$\lambda_D^E = 18.9$$

$\rho_{CPU} = \lambda_C^E * T_{s\,CPU} = 23.2 * .03 = 0.69$

$\rho_{Disk} = \lambda_D^E * T_{s\,pin} = 18.9 * 0.05 = 0.945$

Disk is the bottleneck b/c it's closer to 1 (resource)

q_{Disk} q_{CPU}

$$\rho_{CPU} = \frac{\rho_{CPU}}{1 - \rho_{CPU}} = \frac{0.69}{1 - 0.69} = 2.22 \text{ processes}$$

$$q_{disk} = \frac{0.945}{1 - 0.945} = 17.18$$

$$T_{q_{CPU}} = \frac{q_{CPU}}{\lambda^E_{CPU}} = \frac{2.22}{23.2} = \_\_\_\_\_ \text{ seconds.}$$

What is the probability that the process arrives and finds the CPU idle.

$1 - 0.69 = 0.31$ or 31% chance

$T_{q\,total}$ = Average turnaround time for a process submitted to the whole system.

$$T_{q\ total} = \frac{q_t}{\lambda_t}$$

5 + 10

$$\frac{q\ CPU + q\ Disk}{15}$$

$$= \frac{2.22 + 17.18}{15} = \text{seconds}$$

What arrival would break the system?

Disk would due to it being closer to 1

λ_D^* = break the system

$$S_{Disk} = 1$$

$$20 = \lambda_D^* + 24 * 0.6$$

$$\lambda_D^* = 20 - \left(\frac{24 * 6}{10}\right)$$

$20 - 14.4 = 5.6$ pr/s breaks the system

λ_c^* = breaks the system

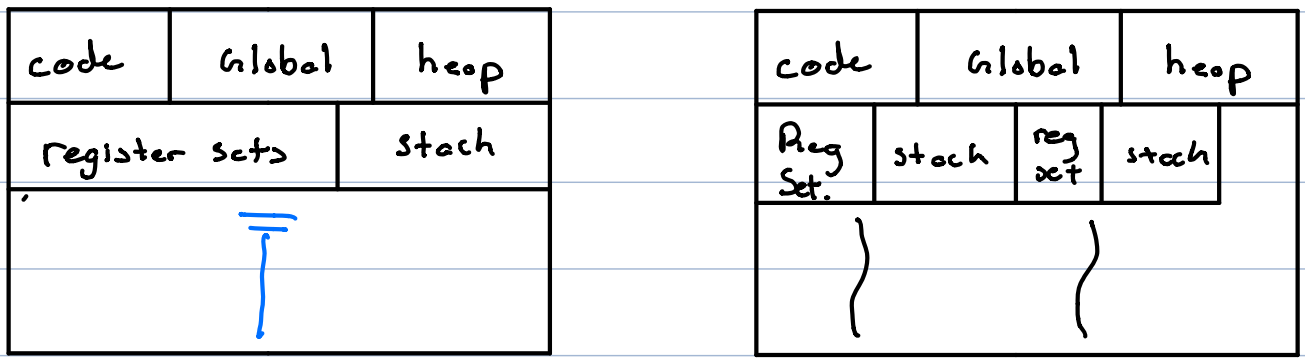
$\rho_{disk} = 1$ (bottleneck)

$\lambda_c^* = 11$

Threads

Goal: Support multiple concurrent paths of execution within a single process.

before



Why?

1. Scalability: Increased speedup in multi-processor systems.

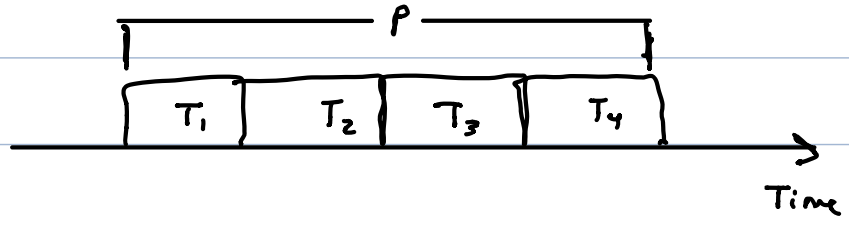
2. Responsiveness: An application can continue to run/execute, even if part of the program being blocked.

[3] Economy : thread creation is faster + efficient than process creation

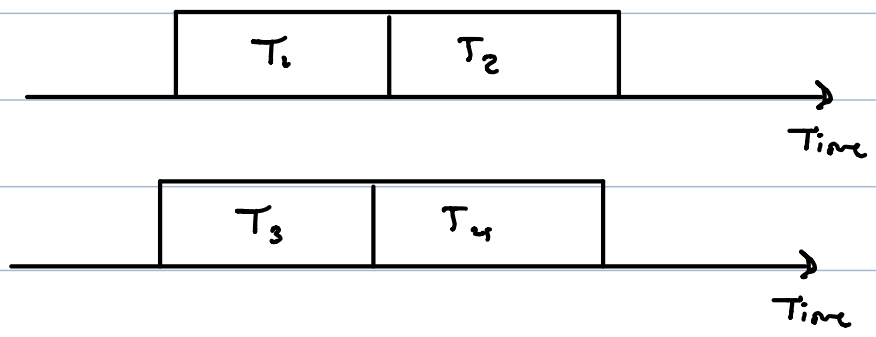
[4] Resource Sharing : By default threads share resources (no need for shared mem / message passing)

T_1 ——— T_4

Single Processor



Multiple Processors

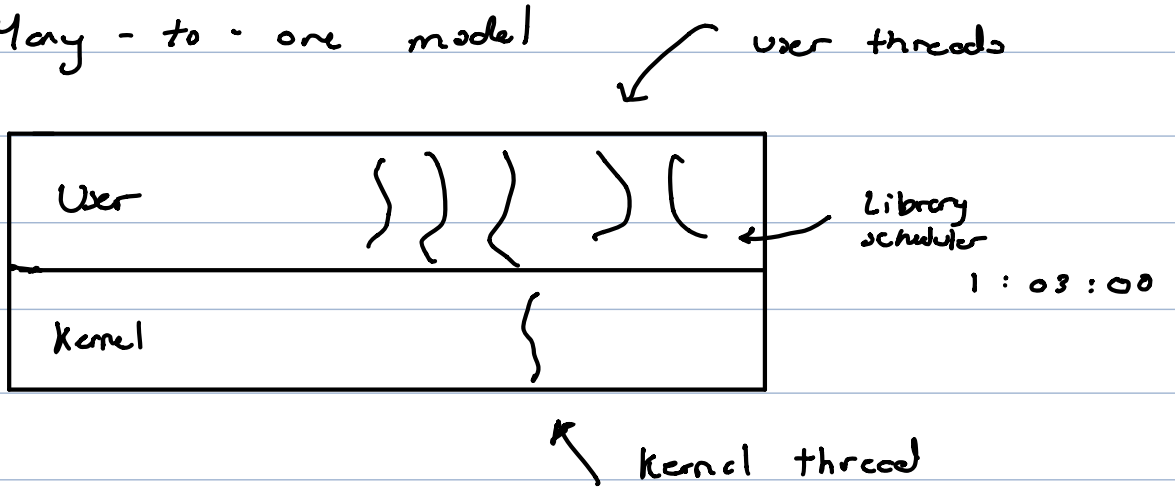


Many to one } are some of the types
one to one

Threading Models

:

1 Many - to - one model



Thread management done by a thread library e.g p threads

Create threads
Kill threads + synchronize
schedule threads

- When one thread blocks then the entire application blocks

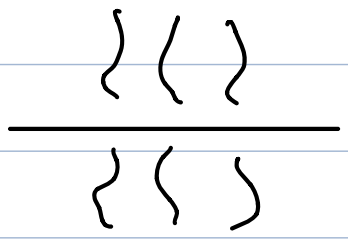
Drawbacks

* process blocks if 1 thread blocks
* no speedup on multi processor systems

2 One - to - one Model

run in parallel

Open MP



3 May - to - May

Interleaved to work with the influx