Threads , Assignment 5

Result.

(I) T_1, T_2 ... $\begin{bmatrix} - & - & - & - \\ - & - & - & - \end{bmatrix}_{N \times N} \times \begin{bmatrix} \vdots & \vdots \\ \vdots & \vdots \end{bmatrix} = \begin{bmatrix} - & - & - & - \end{bmatrix}_{N \times N}$

- Create Threads 1 per row } 'N' threads.

$N = 100000$
$N = 1000$
$N = 20$

(1) Sequential - Matrix Multiplication with 1 thread

(2) Parallel - 'N' threads

100000

Sequential exec: T_{seq}

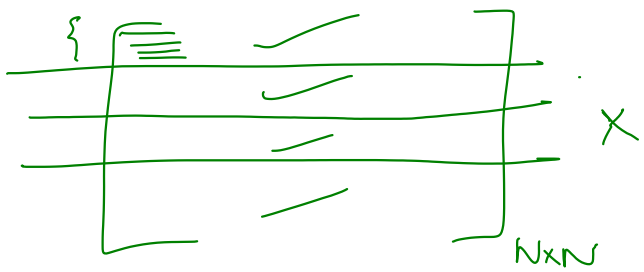
Parallel " : T_{par}

Speedup : $\frac{T_{seq}}{T_{par}}$

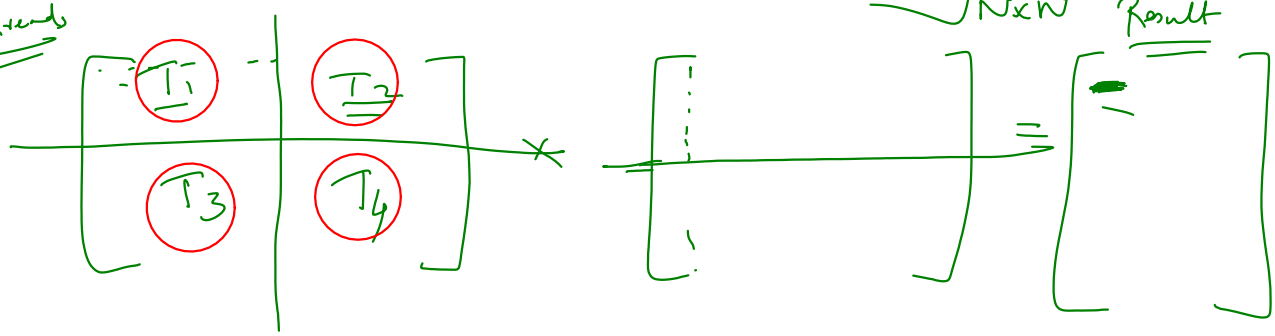
(II) 4 threads.

Result.

"N/4 rows"



(II) 4 threads



$Result_{i,j} =$

$[\ Result_{1,1} = \overline{result_{1,1}} + Result_{1,1}\ ] = \boxed{Locks}$

Main program

```
int main( - - ) {
    Threads t1, t2, ...
    thread_create ( t1, fn, arg )
                  ( t2, fn, arg )
    thread_join (t1);
    exit ( );
}
```

parent

Midsem → Theory + code/debug = Viva

Labs

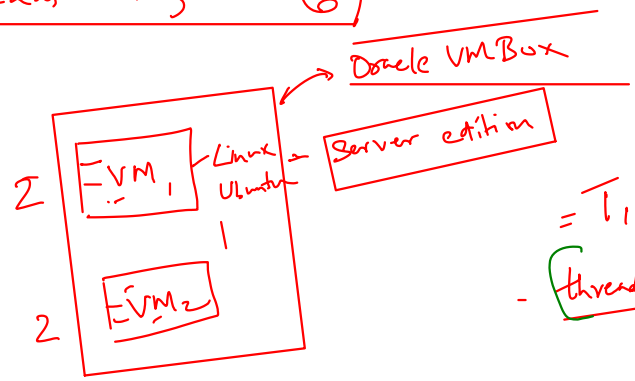
{ 25 + 35 + 40 } / 100

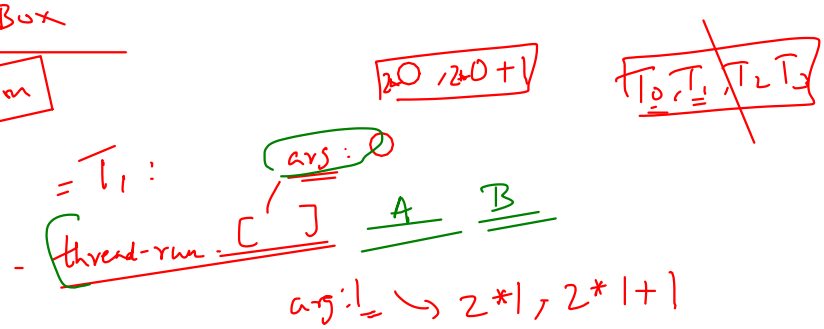
Oct 5: 2 - 5 pm

: Video on

- xv6, gcc --

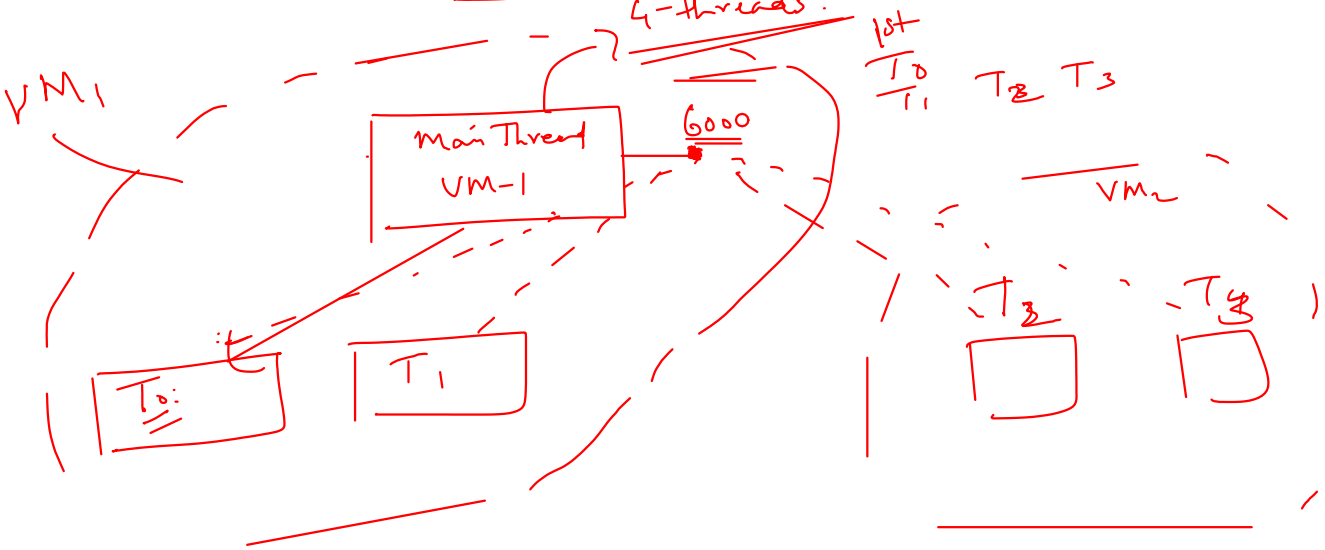
Threads assignment 6



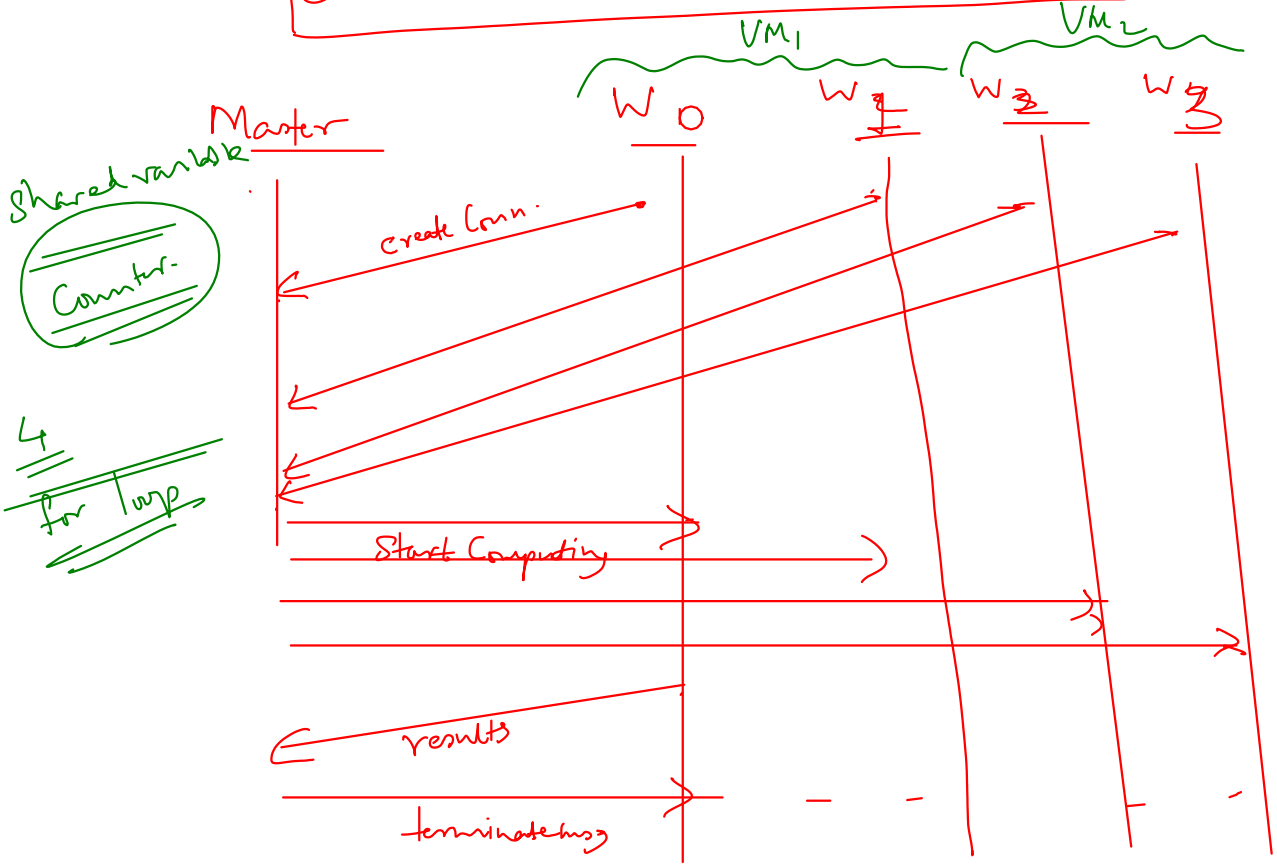


Main Thread on VM1
↳ Result.

Server on some port: 6000



Socket for communicating the results



Master

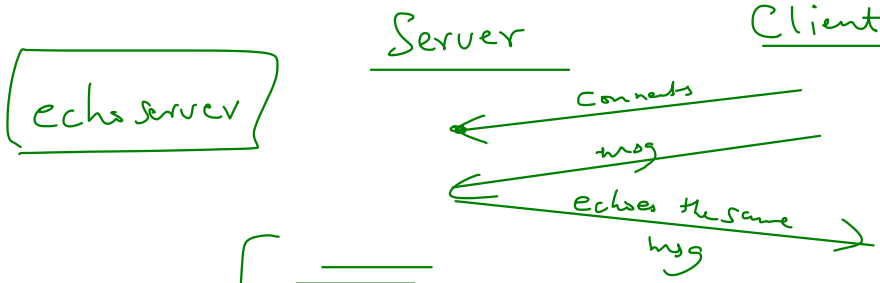
- create Server Socket
- Listen for conn. req.
- When a conn. req. arrives, it'll spawn a thread to handle that conn. req.

Thread — [receive updates from the workers]

[Computing
{ get Lock
updating result }]

Step 1) Create the VMs, get gcc -

Step 2) Create a program — [Socket comm. — chat app^n]



thread-run arg:

VM-1

Main

(Server) : IP addr. — hard-codg

Clients as threads — Computing

VM-2

Clients as threads

Step 3) A, B

- Multi-threaded Server
- Socket comm. — protocol