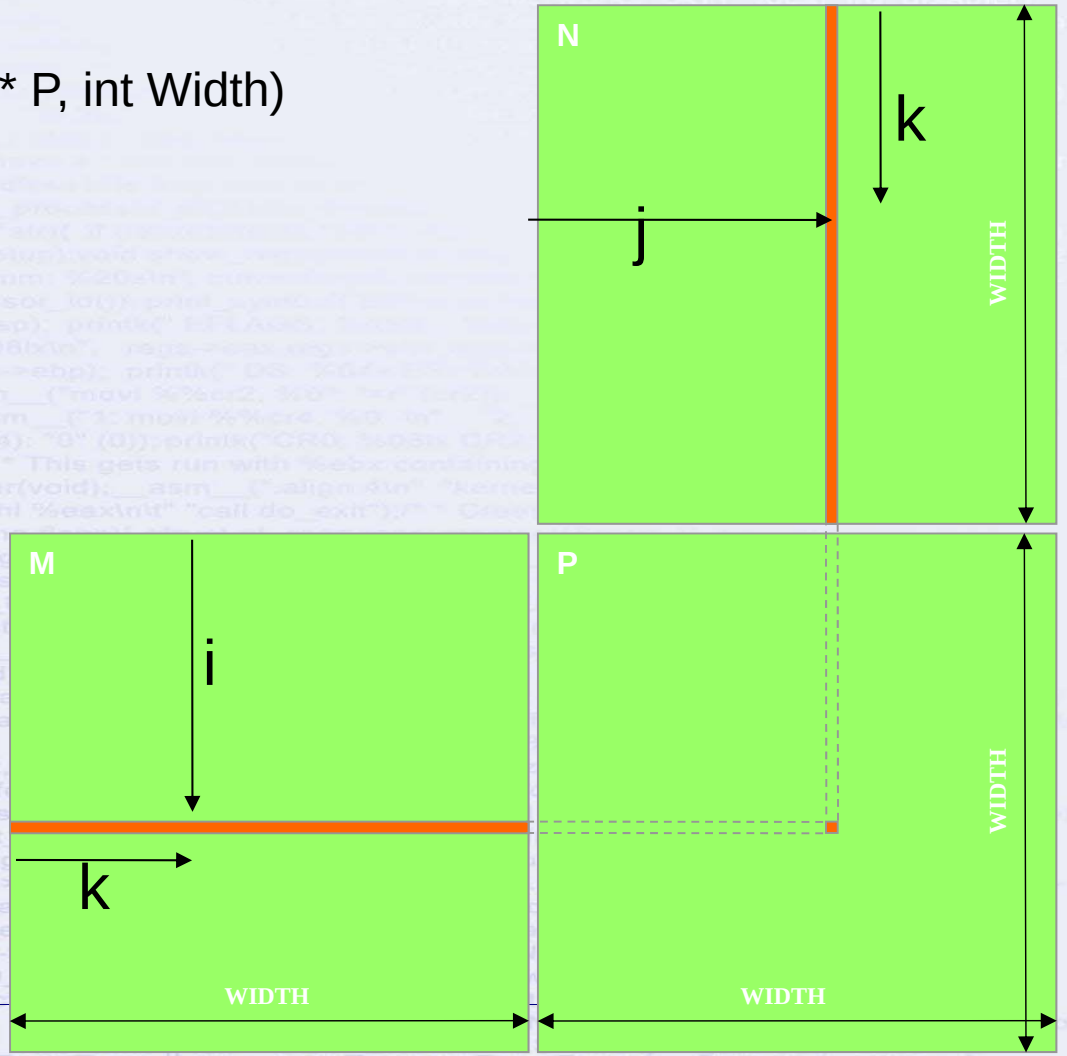
EPL372
Lab Exercise 3:
pThreads, 2D Matrix Multiplication

Reference: https://computing.llnl.gov/tutorials/pthreads/

Matrix Multiplication
A Simple Host Version in C

// Sequential Matrix multiplication Code in C

```
void MatrixMul (float* M, float* N, float* P, int Width)
{
  for (int i = 0; i < Width; ++i)
    for (int j = 0; j < Width; ++j) {
      double sum = 0;
      for (int k = 0; k < Width; ++k) {
        double a = M[i * width + k];
        double b = N[k * width + j];
        sum += a * b;
      }
      P[i * Width + j] = sum;
    }
}
```



Example 1: **matrix_serial.c**

```
65  void mult(int size,
66        int row,
67        int column,
68        matrix_t MA,
69        matrix_t MB,
70        matrix_t MC)
71  {
72    int position;
73
74    MC[row][column] = 0;
75    for(position = 0; position < size; position++) {
76      MC[row][column] = MC[row][column] +
77        ( MA[row][position]  *  MB[position][column] ) ;
78    }
79  }
```

```
93    startTime();
94    /* Process Matrix, by row, column. */
95    for(row = 0; row < size; row++) {
96      for (column = 0; column < size; column++) {
97        mult(size, row, column, MA, MB, MC);
98      }
99    }
00    stopTime();
01    elapsedTime();
```

gcc -Wall -Werror matrix_serial.c -o matrix_serial

Example 2: **matrix_threads.c**

```
85  void mult(int size,
86        int row,
87        matrix_t MA,
88        matrix_t MB,
89        matrix_t MC)
90  {
91    int position, column;
92    for (column = 0; column < size; column++){
93        MC[row][column] = 0;
94            for(position = 0; position < size; position++) {
95              MC[row][column] += ( MA[row][position]   * MB[position][column] ) ;
96            }
97    }
98  }
```

```
104 void *mult_worker(void *arg)
105 {
106   package_t *p=(package_t *)arg;
107      #ifdef DEBUG
108         printf("MATRIX THREAD %d: processing A row %d, B col %d\n", p->id, p->Arow, p->Bcol );
109     #endif
110     mult(p->size, p->Arow, *(p->MA), *(p->MB), *(p->MC));
111     #ifdef DEBUG
112      printf("MATRIX THREAD %d: complete\n", p->id);
113     #endif
114   free(p);
115
116   return(NULL);
117 }
```

For cs6472.in.cs.ucy.ac.cy use -pthread

gcc -lpthread -Wall -Werror matrix_threads.c -o matrix_threads

Try on octa/cs6472 machine (ssh octa (8 cores) ή ssh CS6472 (12 cores))

Locking and Unlocking Mutexes

```
pthread_mutex_t mutexsum;
/* In the Thread Function*/
pthread_mutex_lock (&mutexsum);
dotstr.sum += mysum;
pthread_mutex_unlock (&mutexsum);
pthread_exit((void*) 0);
/* In the Main Function*/
pthread_mutex_init(&mutexsum, NULL);
...
pthread_mutex_destroy(&mutexsum);
pthread_exit(NULL); }
```

HTCondor™

HTCondor High Throughput Computing

HTCondor is a specialized workload management system for compute-intensive jobs. Like other full-featured **batch systems**, HTCondor provides a job queueing mechanism, scheduling policy, priority scheme, resource monitoring, and resource management. Users submit their serial or parallel jobs to HTCondor, HTCondor places them into a queue, chooses when and where to run the jobs based upon a policy, carefully monitors their progress, and ultimately informs the user upon completion.

http://research.cs.wisc.edu/htcondor/

http://research.cs.wisc.edu/htcondor/manual/v8.4/condor-V8_4_3-Manual.pdf

http://research.cs.wisc.edu/htcondor/tutorials/alliance98/submit/submit.html

cs6472:> **condor_version**

$CondorVersion: 7.8.7 Dec 12 2012 BuildID: 86173 $

$CondorPlatform: x86_64_deb_6.0 $

cs6472:> **uname -a**

Linux cs6472 3.2.0-41-generic #66-Ubuntu SMP Thu Apr 25 03:27:11 UTC 2013 x86_64 x86_64 x86_64 GNU/Linux

cs6472.in.cs.ucy.ac.cy

```
cs6472:> lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                12
On-line CPU(s) list:   0-11
Thread(s) per core:    1
Core(s) per socket:    6
Socket(s):             2
NUMA node(s):          2
Vendor ID:             AuthenticAMD
CPU family:            16
Model:                 8
Stepping:              0
CPU MHz:               800.000
BogoMIPS:              4389.06
Virtualization:        AMD-V
L1d cache:             64K
L1i cache:             64K
L2 cache:              512K
L3 cache:              6144K
NUMA node0 CPU(s):     0-5
NUMA node1 CPU(s):     6-11
```

cs6472.in.cs.ucy.ac.cy

```
>cat /proc/meminfo
MemTotal:       32950004 kB
MemFree:        26260460 kB
Buffers:          391508 kB
Cached:          5643936 kB
SwapCached:            0 kB
Active:          4542152 kB
Inactive:        1568552 kB
Active(anon):      76160 kB
Inactive(anon):     2204 kB
Active(file):    4465992 kB
Inactive(file):  1566348 kB
Unevictable:           0 kB
Mlocked:               0 kB
...
```

Condor Submission - Job description file

```
## Condor submit description (script) file for matrix_threads.c
user_path = /home/faculty/petrosp/EPL372/SPRING2014/Labs/Lab3-pThreadsMMCondor

## 1. Specify the [path and] name for the executable file...
executable = $(user_path)/matrix_threads.out

## 2. Specify Condor execution environment.
notification = error
universe = vanilla
getenv = True

## 3. Specify remote execution machines running Windows XP (required)...
REQUIREMENTS = (machine == "cs6472.in.cs.ucy.ac.cy") && ((ARCH == "INTEL") || (ARCH == "X86_64"))

## 4. Define input files and arguments
#Input = stdin.txt
#Arguments = in1 in2 out1

## 5. Define output/error/log files
#------------------------------
Initialdir = $(user_path)
arguments = 1
output = $(user_path)/log/matrix_threads.output
error = $(user_path)/log/matrix_threads.err
log = $(user_path)/log/matrix_threads.log

## 7. Add the job to the queue
Queue
```

Condor commands

Submit a job description file:

`condor_submit <submission_file>`

Monitor the jobs:

`condor_q`

`condor_q <username>`

To monitor condor status:

`condor_status`

To remove your running jobs:

All jobs: **`condor_rm -all`**

A single cluster: **`condor_rm <clusterID>`**

A single job: **`condor_rm <clusterID.jobID>`**

Condor Execution Example

Login to **`cs6472.in.cs.ucy.ac.cy`** using ssh

(i.e **`ssh cs6472.in.cs.ucy.ac.cy`**)

Compile: (add **`–pthread for multithreading applications)`**

```
gcc -Werror -Wall matrix_serial.c -o matrix_serial
```

Edit **`matrixScript.txt`** job description file and change to the **user_path** to the correct path.

Submit to condor: **`condor_submit matrixScript.txt`**

Monitor your jobs: **`condor_q`**

View your results: **`cat log/matrix_threads.output`**

You can also use Scripts to generate execution Scenarios

```
#!/bin/tcsh
#
# Variables
#
set SIM = "~/mysims/sim-alpha/sim-alpha"
set CACHESIZES = (16 32 64 128 256)
set CACHELATENCY = (2 2 3 4 6)
set BENCHMARKS = (ammp gcc equake twolf)

foreach bench ($BENCHMARKS)
  foreach cache ($CACHESIZES)
    @ i++;
    $SIM -bench $bench -size $cache -latency $CACHELATENCY[$i]
  end
  set i = 0;
end
```