King Saud University
College of Computer and Information Sciences
Department of Computer Science
CSC453 – Parallel Processing – Tutorial No – Spring 2022

Question

A Quadtree is a tree:

- that is empty, or
- that is composed of a root and 4 possible sub-Quadtrees.

Let's consider that the data of a Quadtree is stored in a N by N matrix called ***Data***.

Let's consider that we would like to process this Quadtree (data) in parallel. Let's consider the following kernel:

__global__ void **Quadtree_Kernel(**int * **Data,** int **L,** int **C,** int **W,** int **level);**

- This kernel will process the sub-Quatree that is represented by a sub-Matrix of size ***W* * *W*** starting from ***Data[L,C]***.
- ***level*** is the level of the sub-Quadtree.

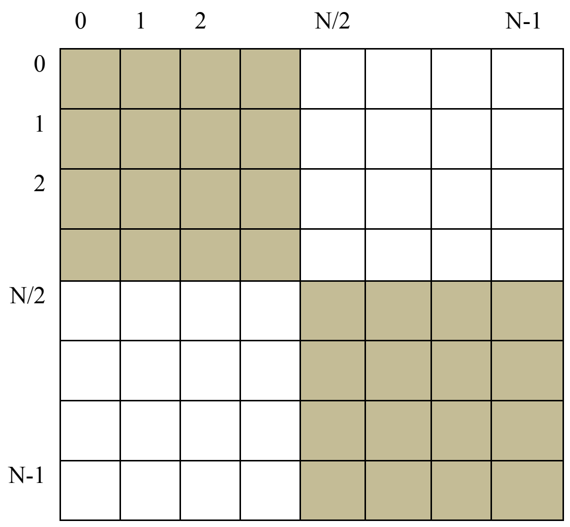
The parallel processing of a Quadtree is launched by the main program using the following call:

Quadtree_Kernel<<<1,4>>>(Data, 0, 0, N, 1);

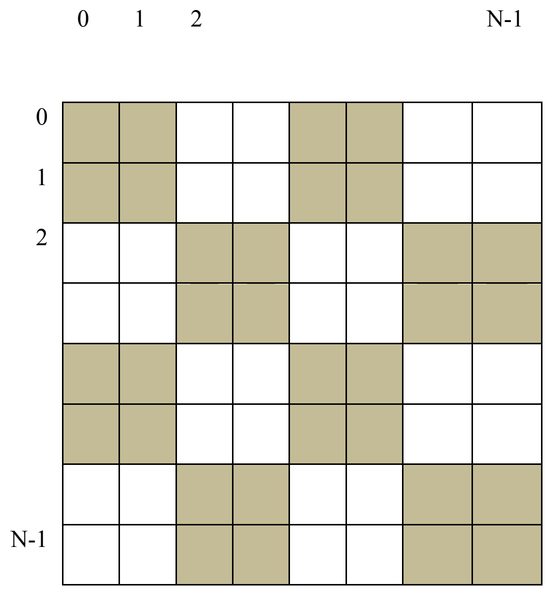
This will launch a grid composed of 1 block of 4 threads. Every thread will process a sub-Quadtree as follows:

- Thread T_0: will process the sub-Quadtree S_0, that corresponds to the data starting from ***Data*** [0, 0] with width = N/2
- Thread T_1: will process the sub-Quadtree S_1 that corresponds to the data starting from ***Data*** [0, N/2] with width = N/2
- Thread T_2: will process the sub-Quadtree S_2 that corresponds to the data starting from ***Data*** [N/2, 0] with width = N/2
- Thread T_3: will process the sub-Quadtree S_3 that corresponds to the data starting from ***Data*** [N/2, N/2] with width = N/2

King Saud University
College of Computer and Information Sciences
Department of Computer Science
CSC453 – Parallel Processing – Tutorial No – Spring 2022



Every sub-Quadtree will be decomposed recursively into 4 sub-Quatrees until no more decomposition are possible.



So, every thread T_i will process a sub-Qaudtree S_i. Every thread T_i will launch 4 threads to decompose its corresponding sub-Qautree as explained above.

1. Give the sub-Quadtree that will be processed by a thread T_i at level 1. **(1 Point)**
2. Give an implementation of the kernel. We assume that we stop at level 10.

—global_void Quadtke(*data, Row, col, width, level)

2)

```
X = threaIdx.x/2
Y = threadIdx.x %2

Ri = X * (width/2) + Row
ci = Y * (width/2) + col
  if( level < 10) {
     Qua....<<<1,4>>>(data, Ri, ci, width/2, level+1)
}
```

1)

```
x = [illegible]
y = [illegible]

Ri = x * (N/2)
Ci = y * (N/2)
```