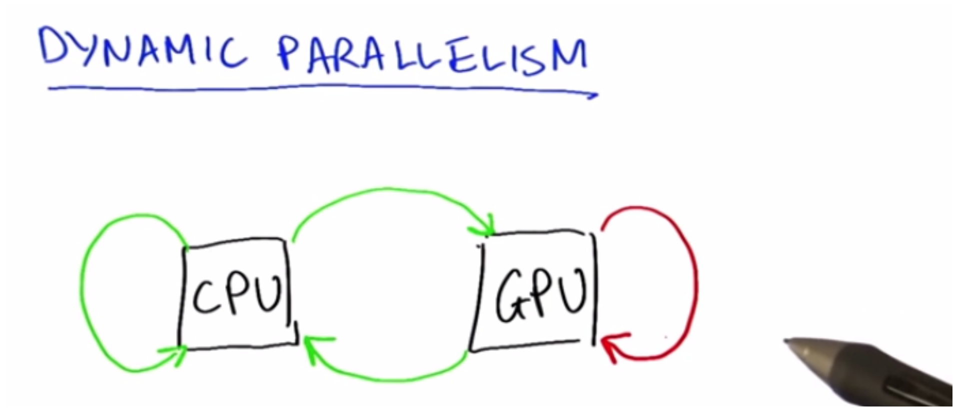
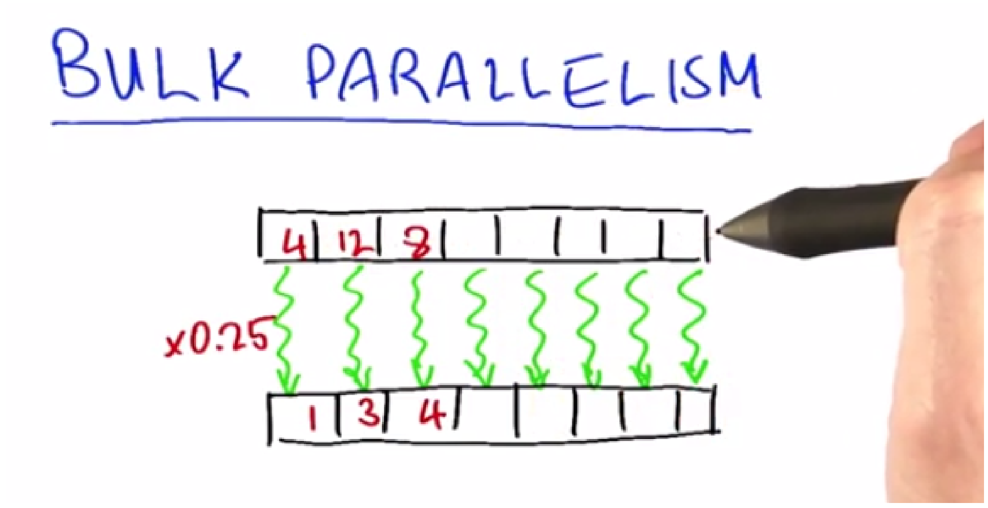
CS344 Introduction to Parallel Programming

Lesson 7.2: Dynamic Parallelism

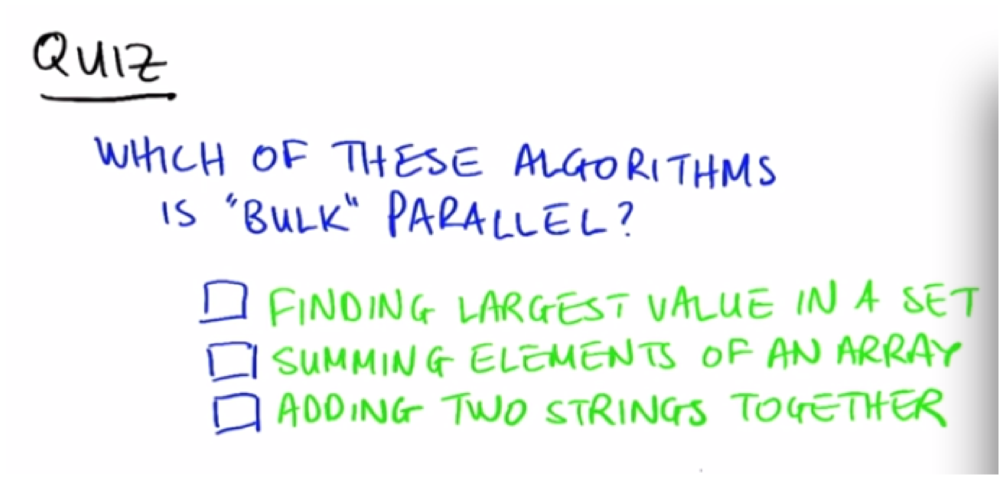
L7.2-7.1-Introduction to Dynamic Parallelism



L7.2-7.2-Bulk Parallelism



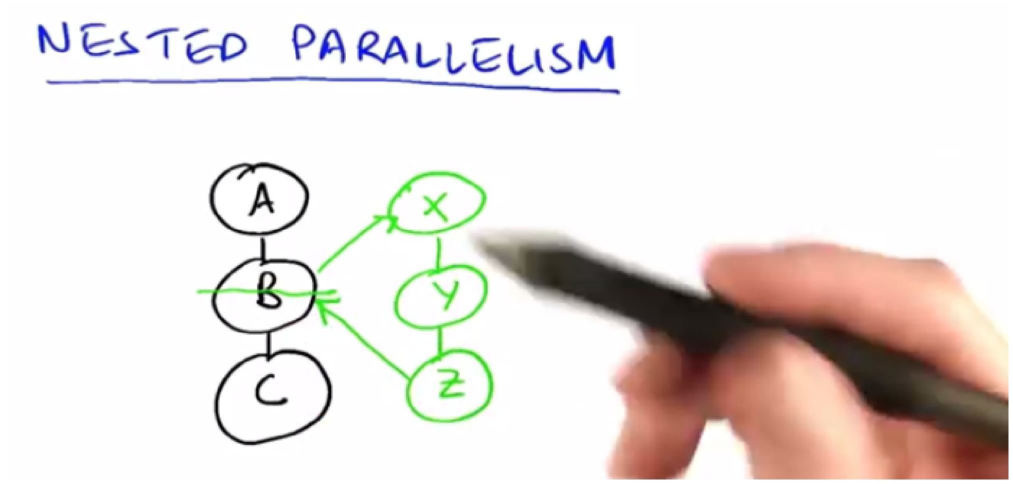
L7.2-7.3-Bulk Parallelism Quiz

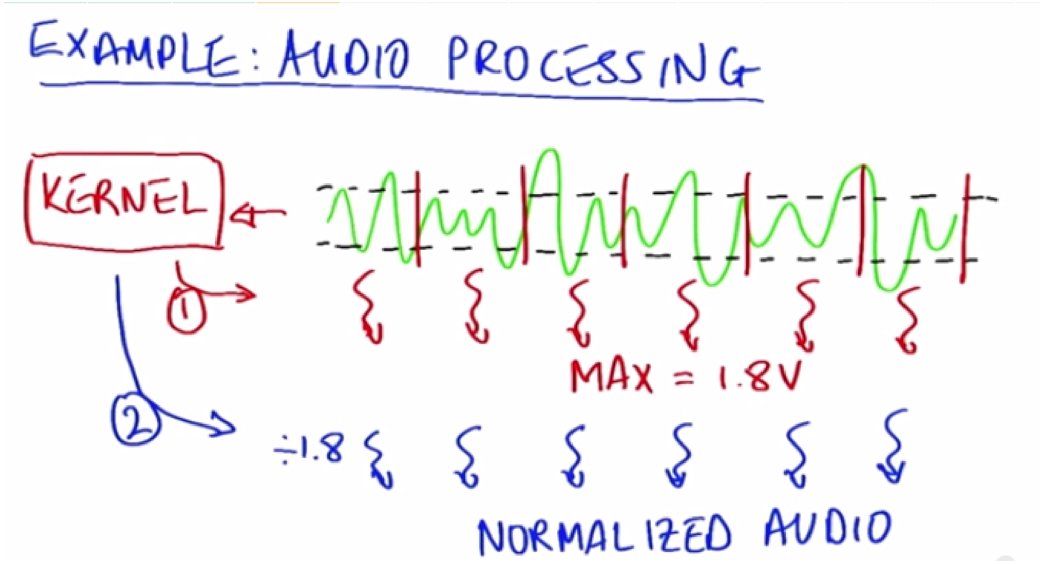

QUIZ

WHICH OF THESE ALGORITHMS IS "BULK" PARALLEL?

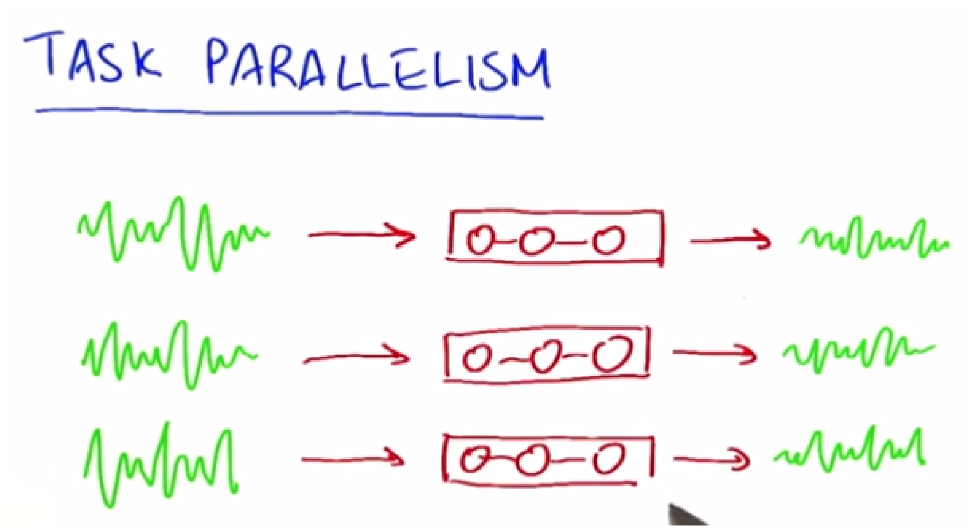
- ☐ FINDING LARGEST VALUE IN A SET
- ☐ SUMMING ELEMENTS OF AN ARRAY
- ☐ ADDING TWO STRINGS TOGETHER

L7.2-7.4-Nested Parallelism



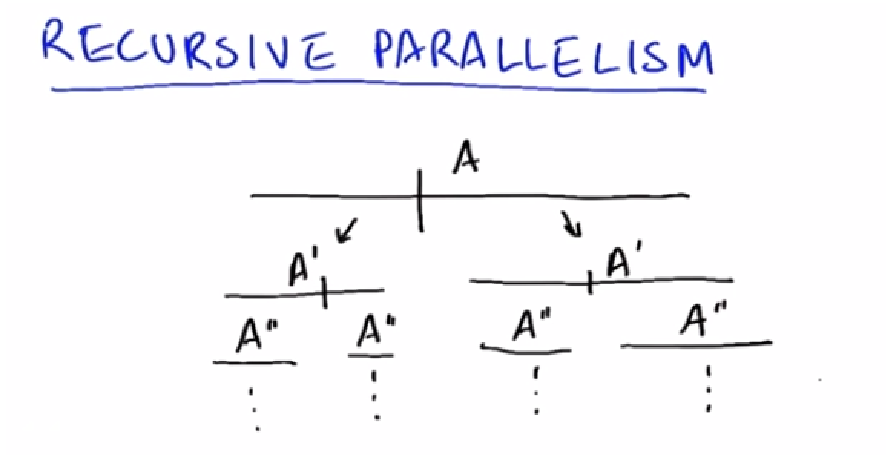


L7.2-7.5-Task Parallelism

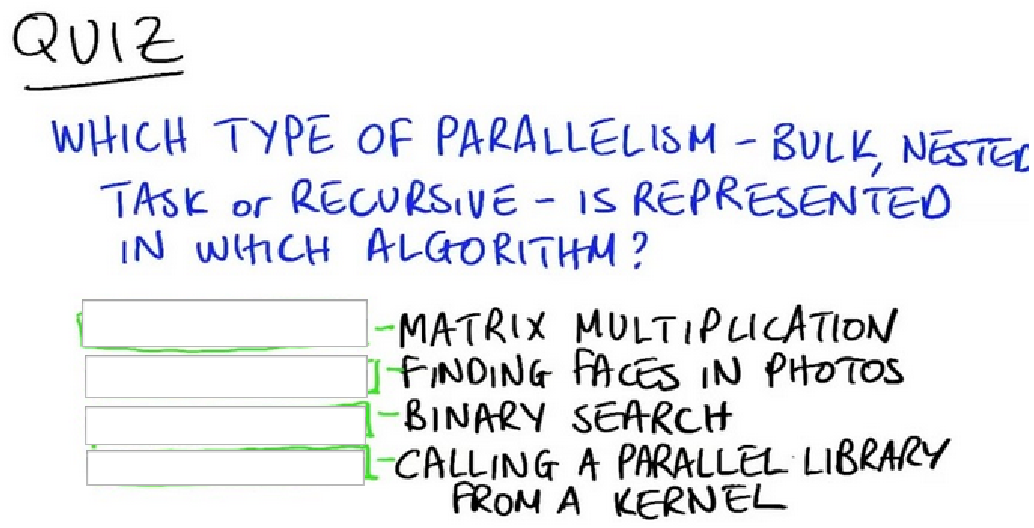


[CS344 Intro to Parallel Programming, Lesson 7.2]

L7.2-7.6-Recursive Parallelism



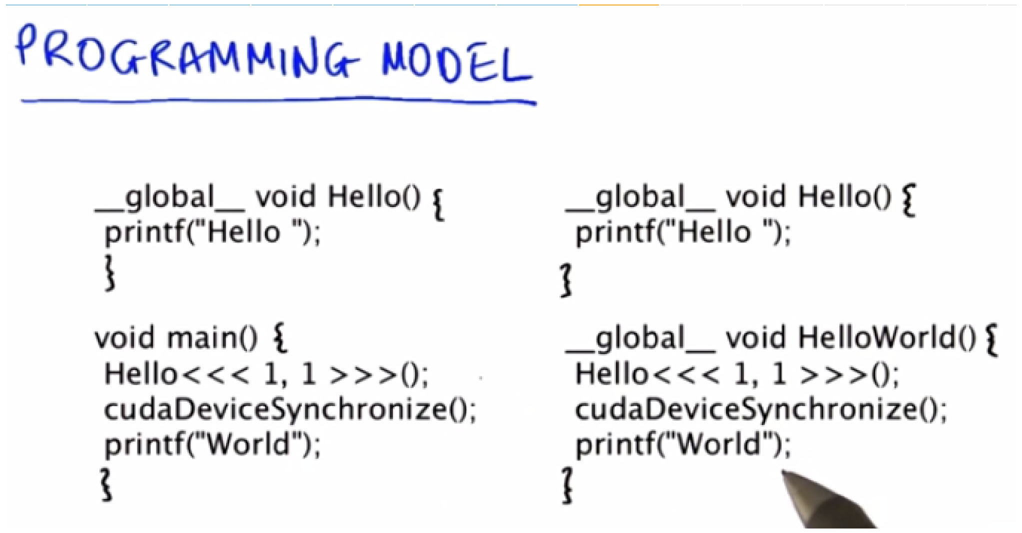
L7.2-7.7-Which Type of Parallelism


QUIZ

WHICH TYPE OF PARALLELISM - BULK, NESTED TASK or RECURSIVE - IS REPRESENTED IN WHICH ALGORITHM?

- MATRIX MULTIPLICATION
- FINDING FACES IN PHOTOS
- BINARY SEARCH
- CALLING A PARALLEL LIBRARY FROM A KERNEL

L7.2-7.8-Programming Model


PROGRAMMING MODEL

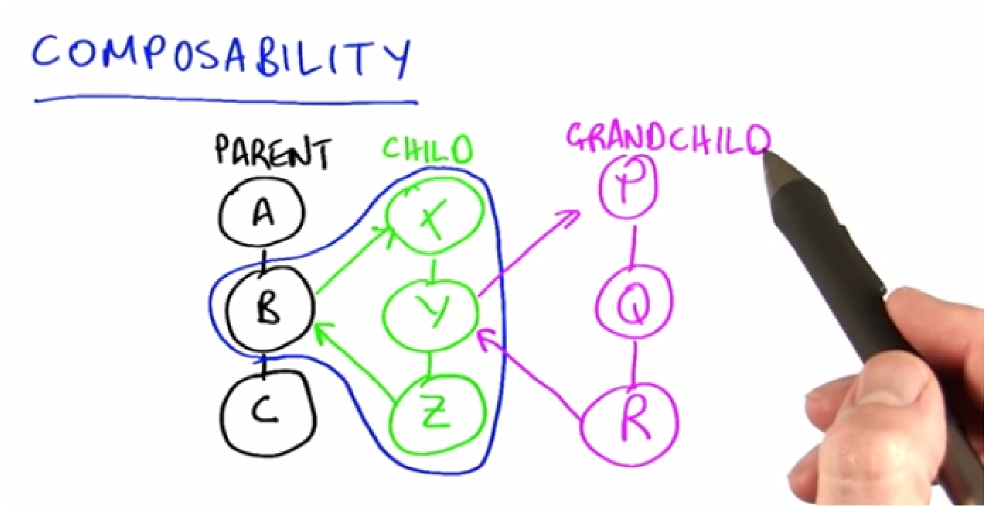
```
__global__ void Hello() {
 printf("Hello ");
}

void main() {
 Hello<<< 1, 1 >>>();
 cudaDeviceSynchronize();
 printf("World");
}
```

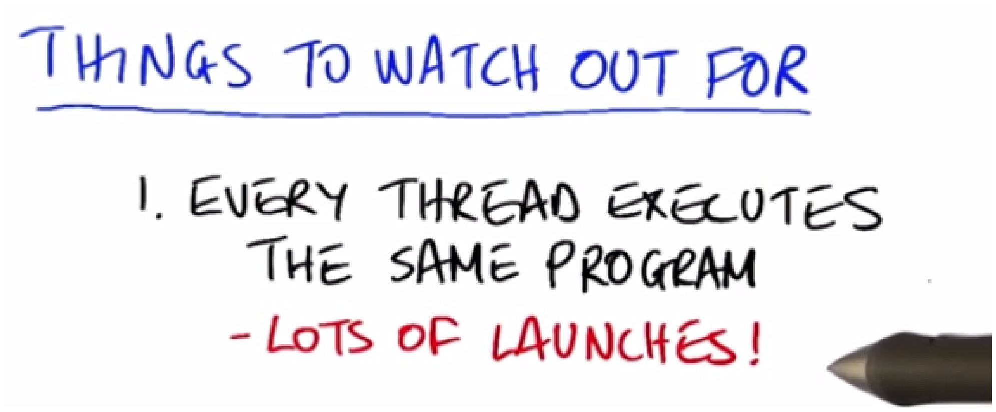
```
__global__ void Hello() {
 printf("Hello ");
}

__global__ void HelloWorld() {
 Hello<<< 1, 1 >>>();
 cudaDeviceSynchronize();
 printf("World");
}
```

L7.2-7.9-Composability



L7.2-7.10-Things To Watch Out For

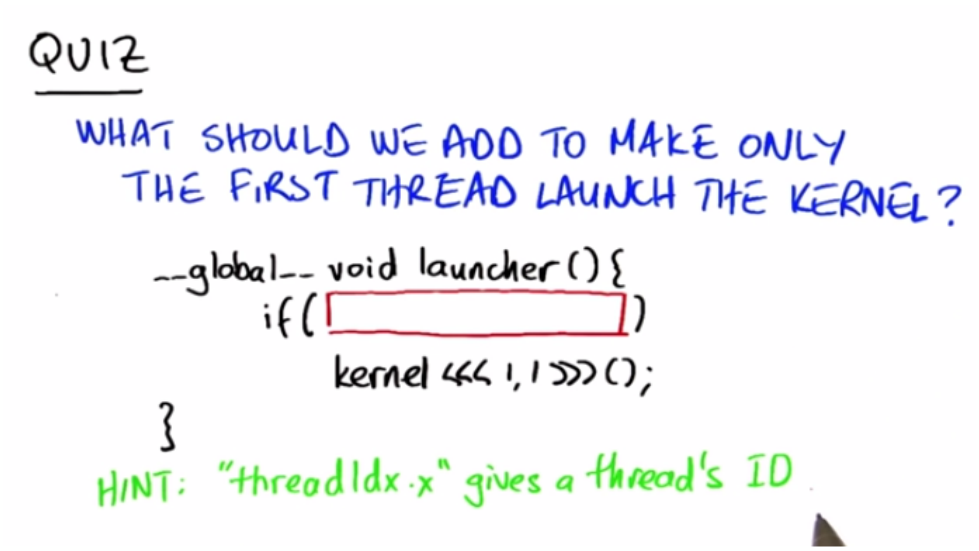


THINGS TO WATCH OUT FOR

1. EVERY THREAD EXECUTES THE SAME PROGRAM

- LOTS OF LAUNCHES!

L7.2-7.11-How Make Only the First Thread Launch a Kernal



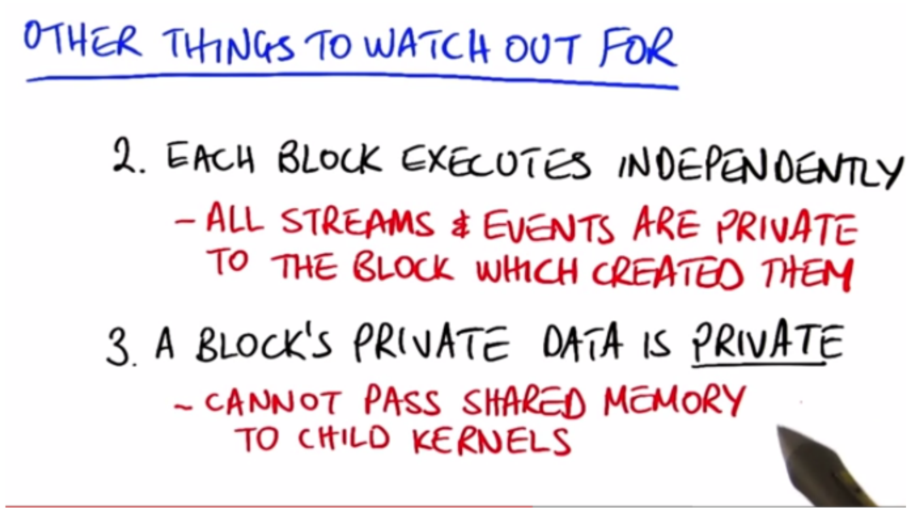
QUIZ

WHAT SHOULD WE ADD TO MAKE ONLY THE FIRST THREAD LAUNCH THE KERNEL?

```
__global__ void launcher() {
    if(              )
        kernel<<<1,1>>>();
}
```

HINT: "threadIdx.x" gives a thread's ID

L7.2-7.12-Other Things to Watch Out For

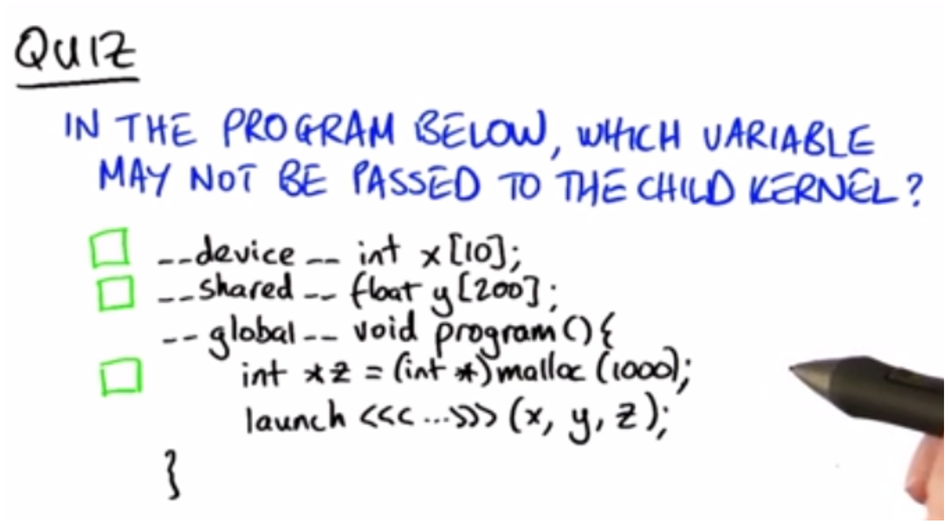


OTHER THINGS TO WATCH OUT FOR

2. EACH BLOCK EXECUTES INDEPENDENTLY
   - ALL STREAMS & EVENTS ARE PRIVATE TO THE BLOCK WHICH CREATED THEM

3. A BLOCK'S PRIVATE DATA IS PRIVATE
   - CANNOT PASS SHARED MEMORY TO CHILD KERNELS

L7.2-7.13-Whith Variable Cannot Be Passed to the Child Thread

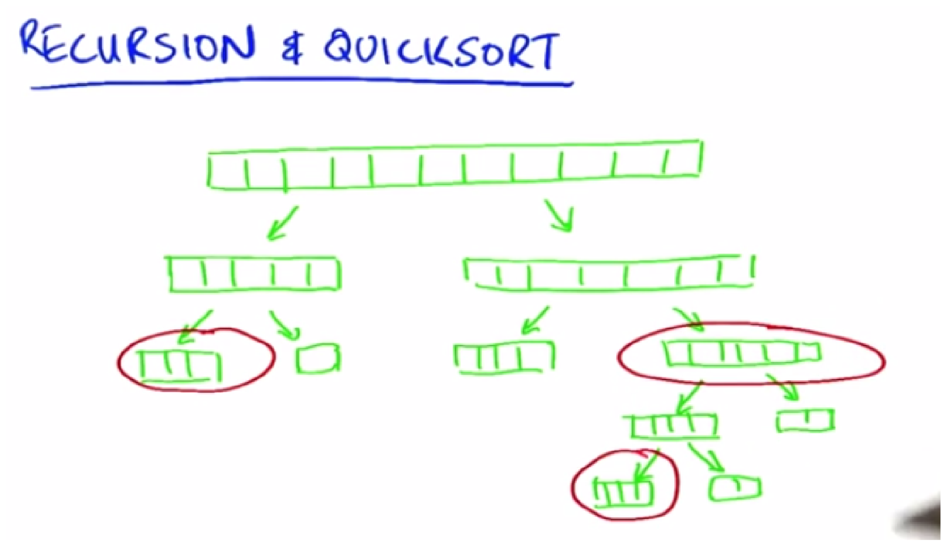


QUIZ

IN THE PROGRAM BELOW, WHICH VARIABLE MAY NOT BE PASSED TO THE CHILD KERNEL?

```
☐ __device__ int x[10];
☐ __shared__ float y[200];
  __global__ void program(){
☐     int *z = (int *)malloc(1000);
      launch <<<...>>> (x, y, z);
  }
```

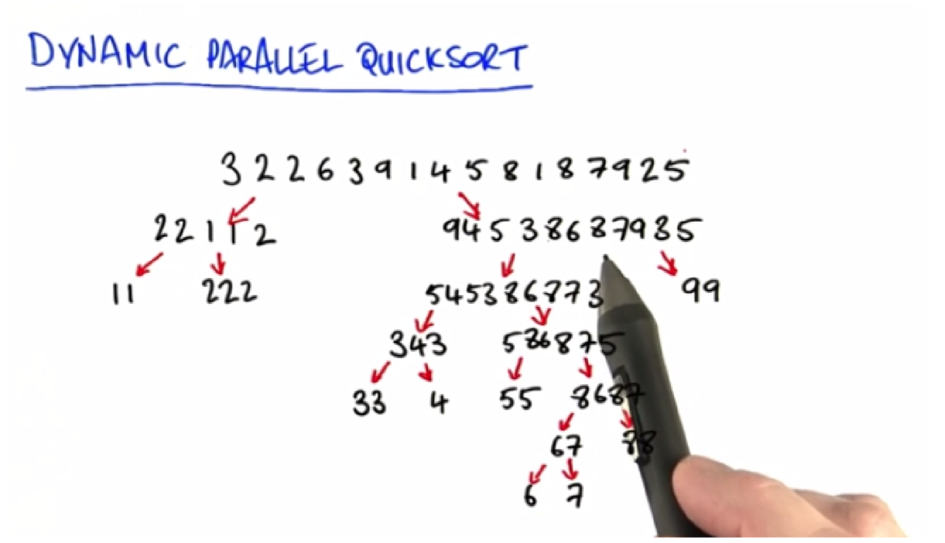
L7.2-7.14-Recursion and Quicksort

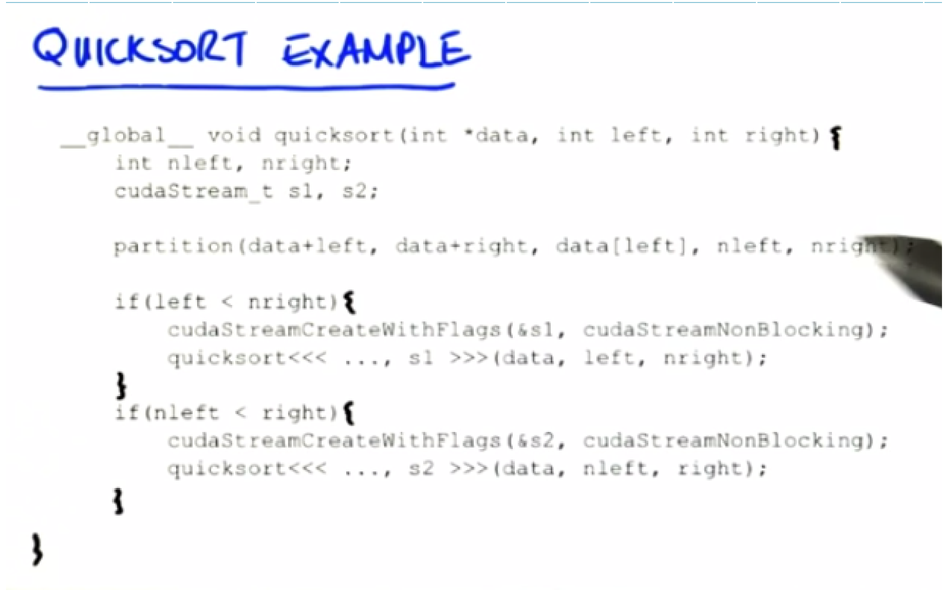


RECURSION & QUICKSORT

[CS344 Intro to Parallel Programming, Lesson 7.2]

L7.2-7.15-Dynamic Parallel Quicksort



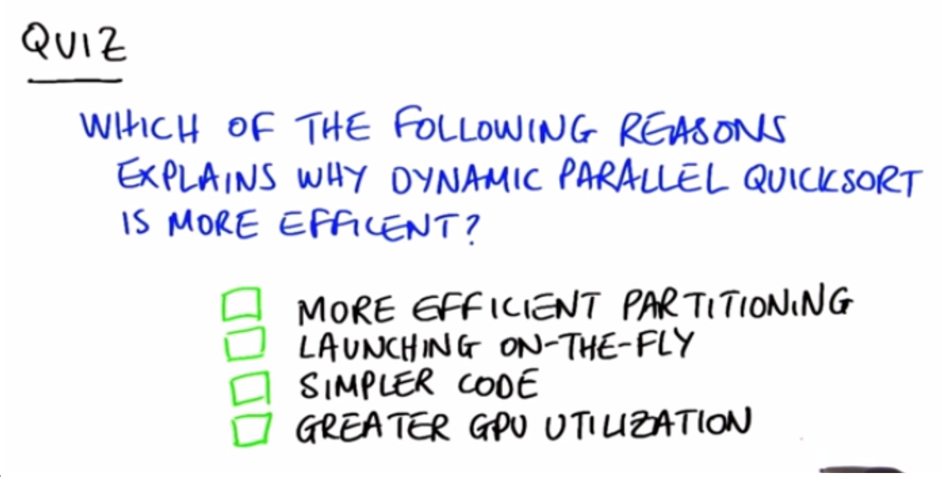

QUICKSORT EXAMPLE

```
__global__ void quicksort(int *data, int left, int right){
    int nleft, nright;
    cudaStream_t s1, s2;

    partition(data+left, data+right, data[left], nleft, nright);

    if(left < nright){
        cudaStreamCreateWithFlags(&s1, cudaStreamNonBlocking);
        quicksort<<< ..., s1 >>>(data, left, nright);
    }
    if(nleft < right){
        cudaStreamCreateWithFlags(&s2, cudaStreamNonBlocking);
        quicksort<<< ..., s2 >>>(data, nleft, right);
    }
}
```

L7.2-7.16-Why Is Dynamic Parallel Quicksort


QUIZ

WHICH OF THE FOLLOWING REASONS EXPLAINS WHY DYNAMIC PARALLEL QUICKSORT IS MORE EFFICENT?

- ☐ MORE EFFICIENT PARTITIONING
- ☐ LAUNCHING ON-THE-FLY
- ☐ SIMPLER CODE
- ☐ GREATER GPU UTILIZATION