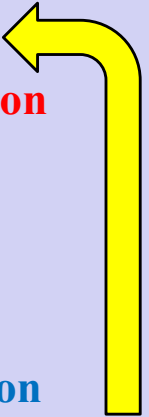


# Assignment 6, Part 1: Big Picture

## Make the doubly-nested big for loops into a kernel

```
__global__ void gpu_histogram_kernel(float *r, float *nhis) {  
    // Perform only doubly nested big pair (i, j) loops on device  
}  
  
void histogram() { // Isolate the kernel in a wrapper function  
    ① // Allocate device data & copy values (atom positions etc.) needed for computation  
        gpu_histogram_kernel<<<numBlocks, threads_per_block>>>(dev_r, dev_nhis);  
    ③ // Copy back computed histogram from device to host  
}  
  
int main() { // No change in the rest of the program  
    // Read atomic positions from pos.d  
    histogram();  
}
```



**Kernel definition**

**② Kernel invocation**

## Map doubly-nested loops to 2D grid & block

# On Assignment 6, Part 1

---

**Q.** Why am I getting “nvcc: command not found” error?

**A.** After adding the following lines in `.bashrc`, you need to either log out and log in again to your account for these commands to be automatically executed, or type “`source .bashrc`” to explicitly execute them:

```
module purge  
module load usc/8.3.0  
module load cuda
```

**Q.** What's `pos.d`?

**A.** Your `pdf.cu` will open `pos.d`, and compute pair distribution function for the atomic positions in it; after compilation

```
nvcc -o pdf pdf.cu
```

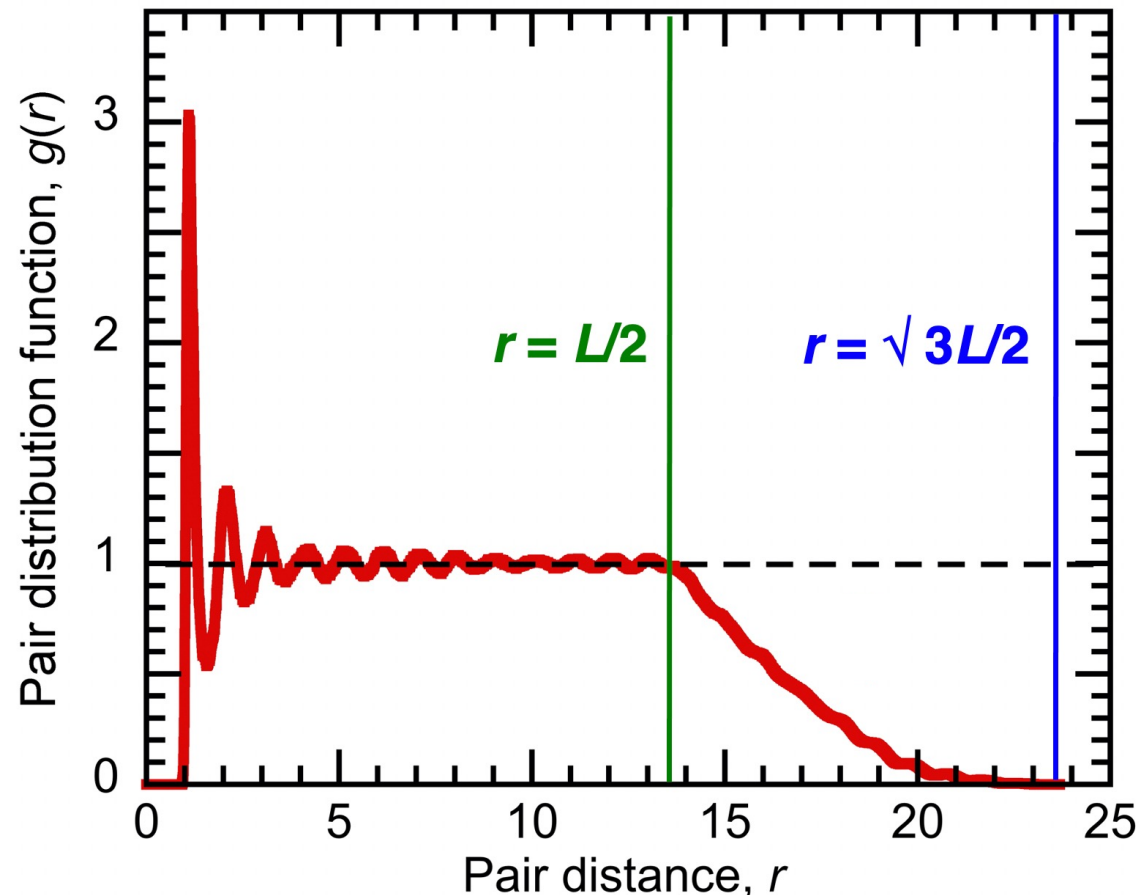
the executable `pdf` & input data `pos.d` must be placed in the same directory as the Slurm script `pdf.sl`

# On Assignment 6, Part 1

**Q.** What to plot?

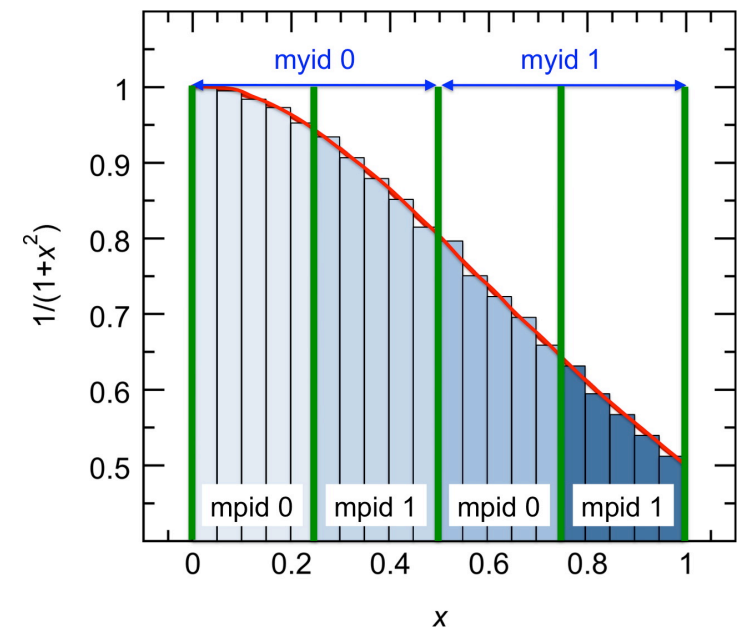
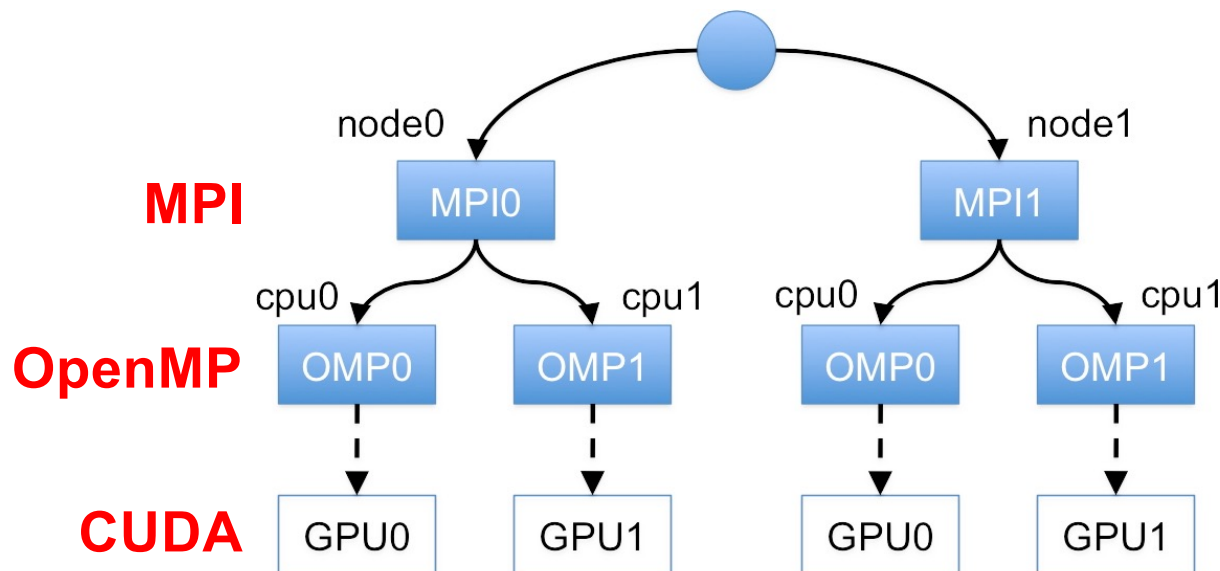
**A.** Pair distribution function  $g(r)$  (right column) vs. atomic-pair distance  $r$  (left column) that will be output into `pdf.d` by your `pdf.cu` program.

With minimum-image convention with simulation box size  $L$ , physically meaningful  $g(r)$  is for  $r \leq L/2$  while we compute it to  $R = R_{\max} = \sqrt{3}L/2$ ; see lecture note on “Pair distribution computation with CUDA”.



# Assignment 6, Part 2: Big Picture

- Hands-on practice of the current default programming language (**MPI + OpenMP + CUDA**) on a cluster of GPU-accelerated multicore computing nodes
- **Who does what:** Hierarchical spatial decomposition with MPI (across computing nodes) + OpenMP (across CPU cores), along with interleaved assignment of loop indices among CUDA threads per OpenMP thread



**Start using MPI+OpenMP+CUDA programming!**