MERGING LISTS

Merging Lists

- Merging lists using CUDA involves leveraging the parallel processing capabilities of NVIDIA GPUs to efficiently combine two or more lists into a single sorted list.
- CUDA is a parallel computing platform and programming model that allows you to use the power of GPUs for general-purpose computing tasks.

Merging Lists

Divide Data: Divide the input lists into smaller chunks that can be processed independently by different threads on the GPU. Each thread will handle a portion of the data.

Allocate GPU Memory: Allocate memory on the GPU to hold the input and output lists. Copy the input lists from the CPU to the GPU memory.

Sorting: Depending on the merging algorithm you choose, you might need to sort the input chunks before merging. This can be done using sorting algorithms like merge sort or parallel sorting networks.

Merging Lists

Merging: Implement a parallel merging algorithm to combine the sorted chunks. One common approach is to use a parallel merge algorithm similar to merge sort's merge step.

Copy Results: Copy the merged and sorted list back from the GPU memory to the CPU memory.

Clean Up: Release the allocated GPU memory.

Merging Lists

```
_global__ void merge_kernel(int* input1, int* input2, int* output, int size1,
int size2) {
    int tid = blockIdx.x * blockDim.x + threadIdx.x;

    if (tid < size1 + size2)
 {
        if (tid < size1) {
            output[tid] = input1[tid];
        } else {
            output[tid] = input2[tid - size1];
        }
    }
}
```

Merging Lists

```
int main() {
    // Initialize input lists on CPU
    int size1 = // size of the first list
    int size2 = // size of the second list
    int* host_input1 = // allocate and populate
    int* host_input2 = // allocate and populate

    // Allocate GPU memory
    int* device_input1;
    int* device_input2;
    int* device_output;
    cudaMalloc(&device_input1, size1 * sizeof(int));
    cudaMalloc(&device_input2, size2 * sizeof(int));
    cudaMalloc(&device_output, (size1 + size2) * sizeof(int));
```

Merging Lists

```
// Copy input data to GPU
    cudaMemcpy(device_input1, host_input1, size1 * sizeof(int),
cudaMemcpyHostToDevice);
    cudaMemcpy(device_input2, host_input2, size2 * sizeof(int),
cudaMemcpyHostToDevice);

    // Configure kernel launch parameters
    int block_size = 256;
    int grid_size = (size1 + size2 + block_size - 1) / block_size;

    // Launch the merge kernel
    merge_kernel<<<grid_size, block_size>>>(device_input1,
device_input2, device_output, size1, size2);
```

Merging Lists

```
// Copy results back to CPU
    int* host_output = new int[size1 + size2];
    cudaMemcpy(host_output, device_output, (size1 + size2) * sizeof(int),
cudaMemcpyDeviceToHost);

    // Clean up
    cudaFree(device_input1);
    cudaFree(device_input2);
    cudaFree(device_output);
    delete[] host_output;

    return 0;
}
```