Mergesort with CUDA

25.02.2020

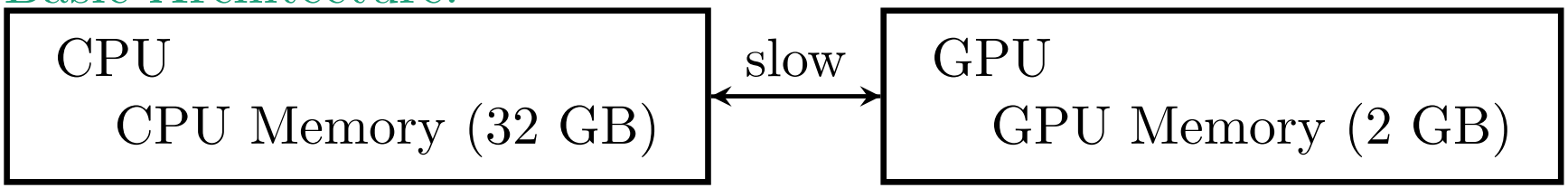
Basic Architecture

Big Ideas

SMID, No fancy stuff, Latency Hiding. Can I show that?

GPU and CPU uses different memories

Basic Architecture:



Memory Creation:

```
1 T* gpu_pointer;
2 unsigned int nBytes = 10*sizeof(int);
3 cudaMalloc((void**)&_gpu_pointer, nBytes);
```

Memory Transfer:

```
1 cudaMemcpy(_gpu_pointer, _cpu_pointer, nBytes, cudal
```

Ideas for Memory Management

```
1  template <typename T>
2  class Storage {
3      public:
4       explicit Storage(const std::vector<T>&);
5
6      private:
7       std::vector<T> _data;
8       T* _cpu_pointer;
9       T* _gpu_pointer;
10      void initialize_gpu_memory();
11 };
```

Memory pool, takes ownership

Initializes the gpu memory as copy

Pointers for cpu/gpu locations

Lazy Memory Sync

```
1  class Storage {
2      public:
3       T* cpu_pointer();
4       T* gpu_pointer();
5       const T* cpu_pointer_const();
6       const T* gpu_pointer_const();
7
8      private:
9       std::string head;
10      void sync_to_cpu();
11      void sync_to_gpu();
12 };
```

accesses const or non-const pointers

head $\in \{\text{CPU}, \text{GPU}, \text{SYNC}\}$

if non-const function: change head to location

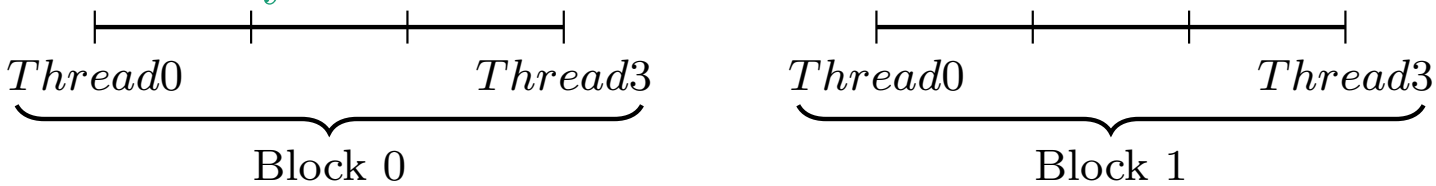
Lazily sync if required pointer $!=$ head

Launching CUDA threads

Cuda Program

```
1  dim3 Grid(2)
2  dim3 Block(4)
3  add_kernel<<<Grid, Block>>>(...)
```

Thread Layout:



Addition:

```
1  __global__
2  add_kernel(float* A, float* B, float* C, int n) {
3      int i = blockDim.x * blockIdx.x + threadIdx.x;
4      if (i < n) {
5          C[i] = A[i] + B[i];
6      }
7  }
```

Merge

Basic Merge Operation

$$A = 578912141516$$

$$B = 12346101113$$

$$C = ????????????????$$

```
1  void merge(T* a, T* b, T* c, int sz_a, int sz_b) {
2       int i = 0, j = 0, k = 0;
3       while (k < sz_a + sz_b) {
4            if (i == sz_a)
5                 c[k++] = b[j++];
6            else if (j == sz_b)
7                 c[k++] = a[i++];
8            else if (a[i] <= b[j])
9                 c[k++] = a[i++];
10           else
11                c[k++] = b[j++];
12      }
13 }
```

How to spwan to many threads?

Naive: 2 Threads, half A and B

Example:

$$A = 0000$$

$$B = 1111$$

$$C = ????????$$

$$A = \underbrace{00}_{\text{Thread 1}} \mid \underbrace{00}_{\text{Thread 2}}$$

$$B = \underbrace{11}_{\text{Thread 1}} \mid \underbrace{11}_{\text{Thread 2}}$$

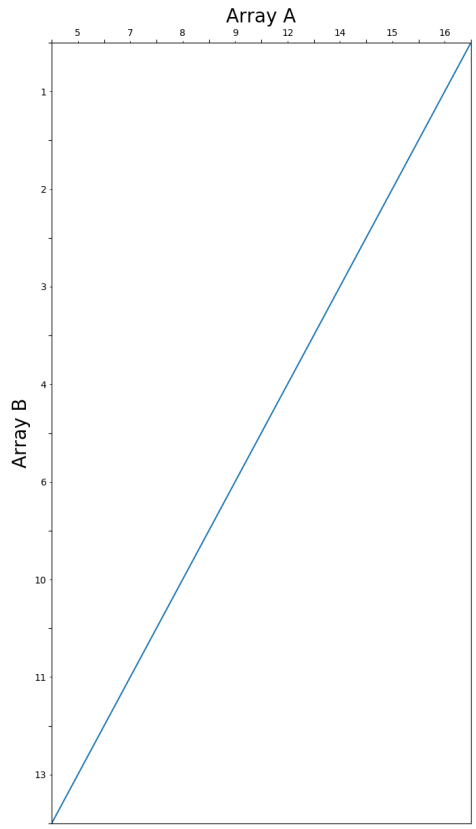
$$C = \underbrace{????}_{\text{Thread 1}} \mid \underbrace{????}_{\text{Thread 2}}$$

Result:

$$C = \underbrace{0011}_{\text{Thread 1}} \mid \underbrace{0011}_{\text{Thread 2}}$$

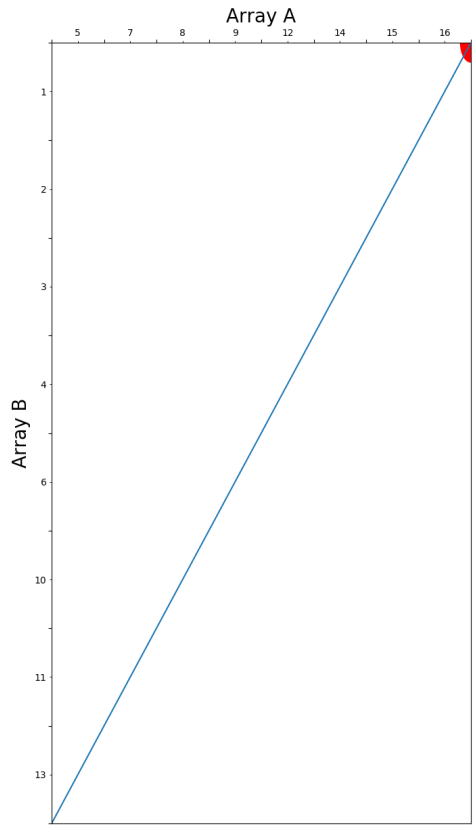
Merge

How to allocated work?



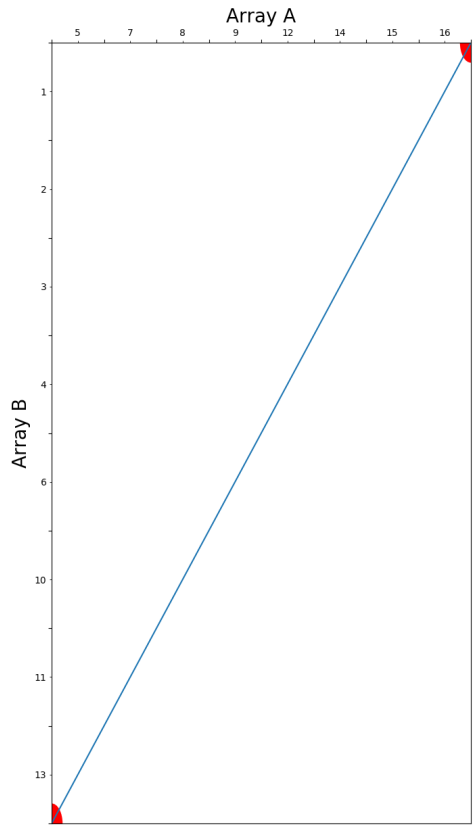
Merge

How to allocated work?



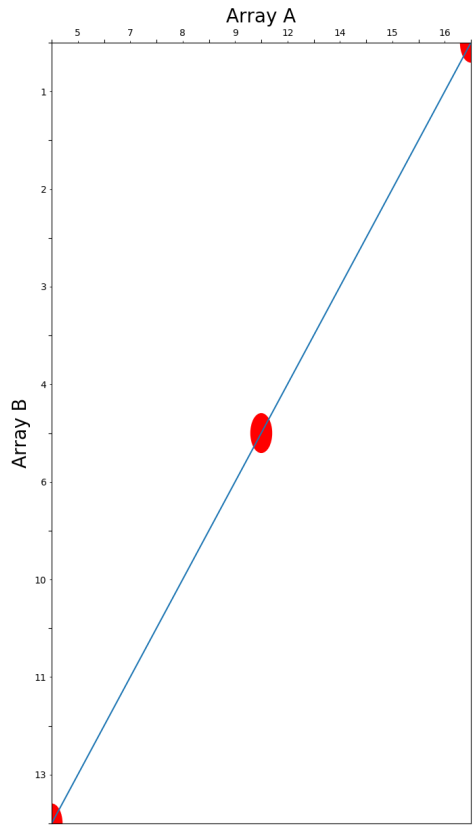
Merge

How to allocated work?



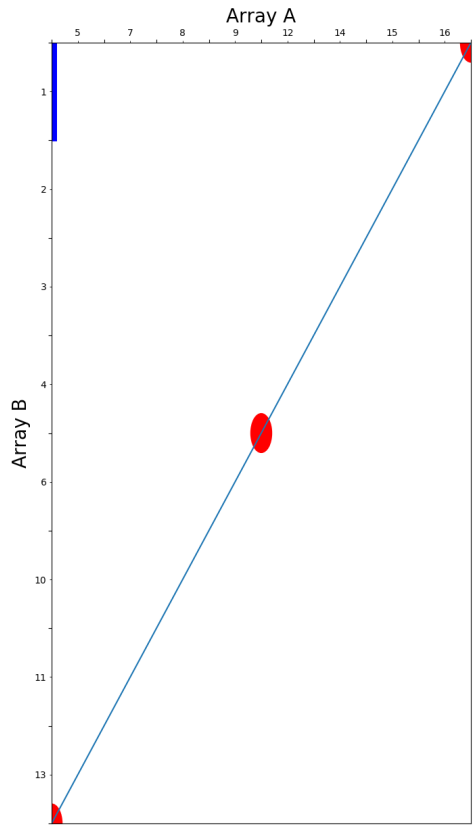
Merge

How to allocated work?



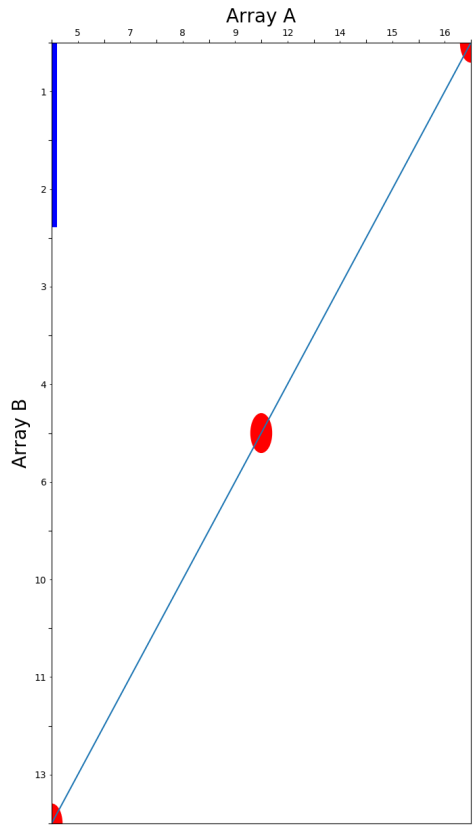
Merge

How to allocated work?



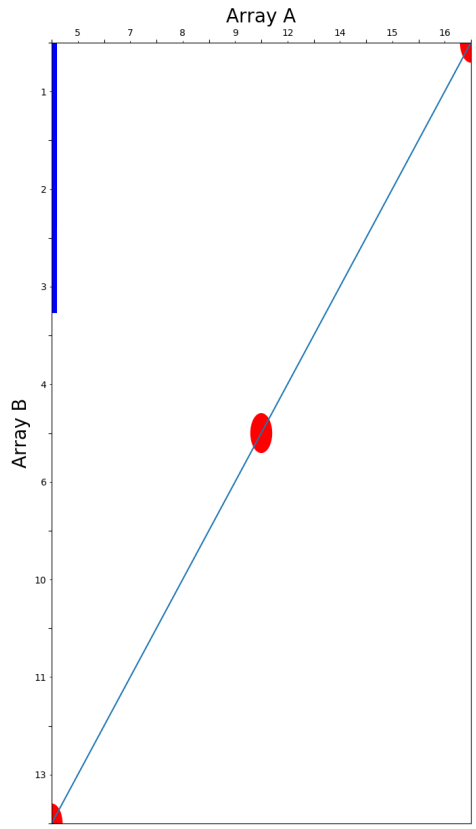
Merge

How to allocated work?



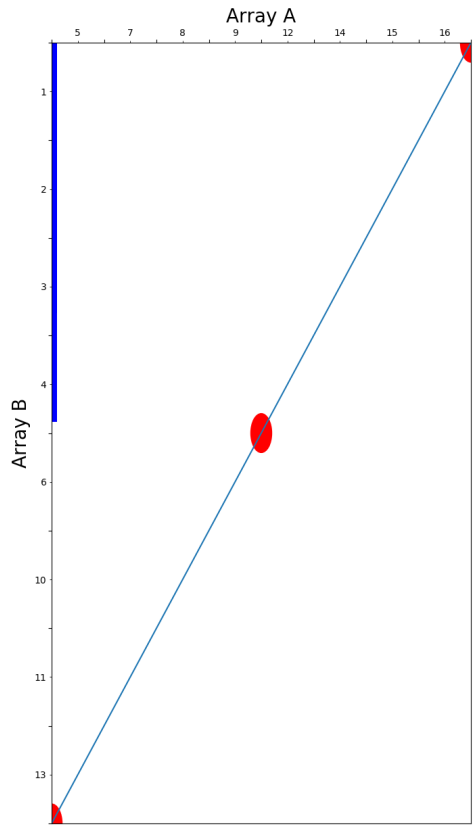
Merge

How to allocated work?



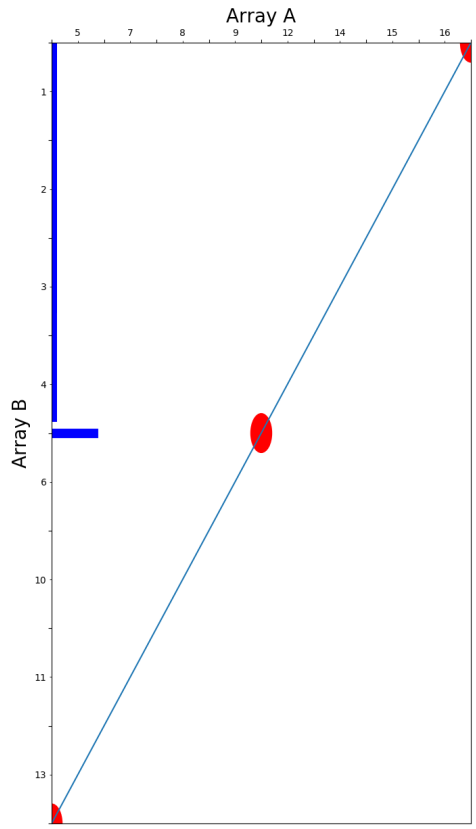
Merge

How to allocated work?



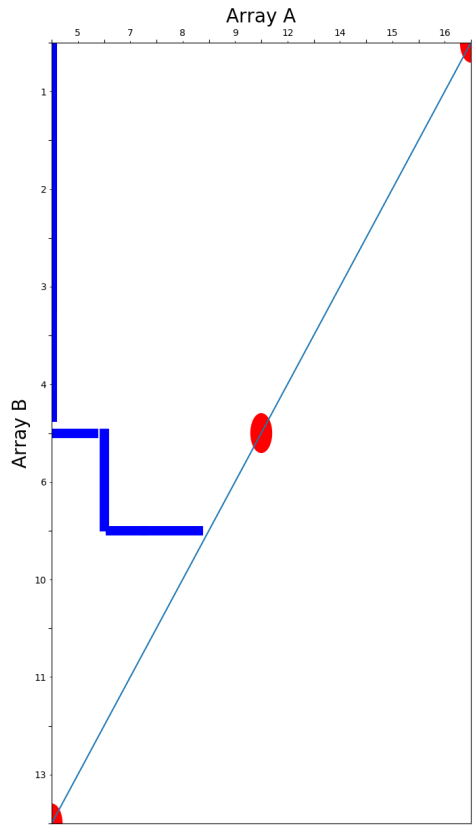
Merge

How to allocated work?



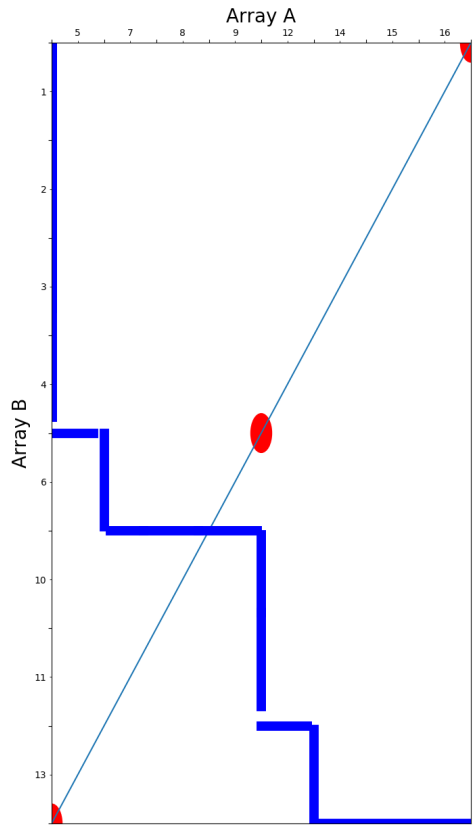
Merge

How to allocated work?



Merge

How to allocated work?



Merge

Comutation Procedure

```
1  ___global___
2  void paralleMerge(int* a, int sz_a, int* b, int sz_
3       int diag = threadIdx.x * length;
4       int a_start = mergepath(a, sz_a, b, sz_b, diag)
5       int b_start = diag - a_start;
6       merge2(a, a_start, sz_a, b, b_start, sz_b, c, d
7  }
```

Each tread works on one part

Thread calculates the value A_lower for itself

Calcultes the also the B_lower (Why does that work again?)

merges the two arrays

Problem: Slow as a Snail

show the growth rates vs std::mergesort

Reason: So much global meory access

50 Percent of the global memory traffic is caused by 3 Percent of the values
Corrobation: Cuda performance tool

Memory Hirachy of CUDA

The different memories and their sizes

show the plot of the different memories and their relative size on my card

Merging with local memory

Describe the shared memory

show the plot of the different memories and their relative size on my card

Show the results

show the plot of the different memories and their relative size on my card