```
!pwd
!ls -la
```

```
    /content
    total 16
    drwxr-xr-x 1 root root 4096 Oct 14 16:31 .
    drwxr-xr-x 1 root root 4096 Oct 22 17:56 ..
    drwxr-xr-x 1 root root 4096 Oct 14 16:32 .config
    drwxr-xr-x 1 root root 4096 Oct 14 16:31 sample_data
```

```
%%writefile labsolReduction0.cu

// includes, kernels
#include <stdio.h>
#include <assert.h>

#define NUM_ELEMENTS 512


// **===-----------------------------------------------------------------===**
//! @param g_idata  input data in global memory
//                  result is expected in index 0 of g_idata
//! @param n        input number of elements to scan from input data
// **===-----------------------------------------------------------------===**

__global__ void reduction(float *g_data, int n)
{
  int stride;
  // Define shared memory
  __shared__ float scratch[NUM_ELEMENTS];

  // Load the shared memory
  scratch[threadIdx.x ] = g_data[threadIdx.x];
  if(threadIdx.x + blockDim.x < n)
    scratch[threadIdx.x + blockDim.x] = g_data[threadIdx.x + blockDim.x];
  __syncthreads();

  // Do sum reduction from shared memory

  for(stride = 1 ; stride < blockDim.x; stride *= 2)
  {
        __syncthreads();
        if(threadIdx.x % (2*stride) == 0)
              scratch[threadIdx.x] += scratch[threadIdx.x + stride];

  }

  // Store results back to global memory
  if(threadIdx.x == 0)
    g_data[0] = scratch[0];

  return;
}
```

```
////////////////////////////////////////////////////////////////////////////
// Program main
////////////////////////////////////////////////////////////////////////////
void runTest( int argc, char** argv);
float computeOnDevice(float* h_data, int array_mem_size);
extern "C" void computeGold( float* reference, float* idata, const unsigned int len

int main( int argc, char** argv)
{
    runTest( argc, argv);
    return EXIT_SUCCESS;
}

////////////////////////////////////////////////////////////////////////////
//! Run naive scan test
////////////////////////////////////////////////////////////////////////////
void runTest( int argc, char** argv)
{
    int num_elements = NUM_ELEMENTS;

    const unsigned int array_mem_size = sizeof( float) * num_elements;

    // allocate host memory to store the input data
    float* h_data = (float*) malloc( array_mem_size);

    // * No arguments: Randomly generate input data and compare against the host's

            // initialize the input data on the host to be integer values
            // between 0 and 1000
            for( unsigned int i = 0; i < num_elements; ++i)
            {
                //h_data[i] = floorf(1000*(rand()/(float)RAND_MAX));
                h_data[i] = i*1.0;

            }

        // compute reference solution
    float reference = 0.0f;
    computeGold(&reference , h_data, num_elements);


    float result = computeOnDevice(h_data, num_elements);

    // We can use an epsilon of 0 since values are integral and in a range
    // that can be exactly represented
    float epsilon = 0.0f;
    unsigned int result_regtest = (abs(result - reference) <= epsilon);
    printf( "Test %s\n", (1 == result_regtest) ? "PASSED" : "FAILED");
    printf( "device: %f  host: %f\n", result, reference);
    // cleanup memory
    free( h_data);
}

////////////////////////////////////////////////////////////////////////////
```

```
// Take h_data from host, copies it to device, setup grid and thread
// dimentions, excutes kernel function, and copy result of scan back
// to h_data.
// Note: float* h_data is both the input and the output of this function.
////////////////////////////////////////////////////////////////////////


float computeOnDevice(float* h_data, int num_elements)
{
  float* d_data = NULL;
  float result;

  // Memory allocation on device side
  cudaMalloc((void**)&d_data, num_elements*sizeof(float));

  // Copy from host memory to device memory
  cudaMemcpy(d_data, h_data, num_elements*sizeof(float), cudaMemcpyHostToDevice);

  //int threads = (num_elements/2) + num_elements%2;
  int threads = num_elements;

  // Invoke the kernel
  reduction<<<1,threads>>>(d_data,num_elements);

  // Copy from device memory back to host memory
  cudaMemcpy(&result, d_data, sizeof(float), cudaMemcpyDeviceToHost);
  cudaFree(d_data);
  return result;
}

////////////////////////////////////////////////////////////////////////
void computeGold( float* reference, float* idata, const unsigned int len)
{
  reference[0] = 0;
  double total_sum = 0;
  unsigned int i;
  for( i = 0; i < len; ++i)
  {
      total_sum += idata[i];
  }
  reference[0] = total_sum;
}
////////////////////////////////////////////////////////////////////////
```

```
 ⇨  Overwriting labsolReduction0.cu
```

```
!/usr/local/cuda/bin/nvcc -I /usr/local/cuda/samples/common/inc -arch=sm_35 -rdc=tr
```

```
!./labsolReduction0
```

```
    Test PASSED
```

```
!nvprof ./labsolReduction0
```

```
    ==2810== NVPROF is profiling process 2810, command: ./labsolReduction0
    Test PASSED
    device: 130816.000000  host: 130816.000000
    ==2810== Profiling application: ./labsolReduction0
    ==2810== Profiling result:
                Type  Time(%)      Time     Calls       Avg       Min       Max  N
     GPU activities:   68.19%  8.0960us         1  8.0960us  8.0960us  8.0960us  r
                       16.17%  1.9200us         1  1.9200us  1.9200us  1.9200us  [
                       15.63%  1.8560us         1  1.8560us  1.8560us  1.8560us  [
          API calls:   99.47%  188.56ms         1  188.56ms  188.56ms  188.56ms  c
                        0.30%  572.94us         1  572.94us  572.94us  572.94us  c
                        0.13%  246.88us        97  2.5450us     177ns  107.95us  c
                        0.04%  77.374us         1  77.374us  77.374us  77.374us  c
                        0.02%  37.803us         2  18.901us  17.338us  20.465us  c
                        0.02%  29.895us         1  29.895us  29.895us  29.895us  c
                        0.01%  27.932us         1  27.932us  27.932us  27.932us  c
                        0.00%  4.3930us         1  4.3930us  4.3930us  4.3930us  c
                        0.00%  1.8920us         3     630ns     178ns  1.2840us  c
                        0.00%  1.2160us         2     608ns     326ns     890ns  c
                        0.00%     336ns         1     336ns     336ns     336ns  c
```

CASO1

```
%%writefile labsolReduction1.cu

// includes, kernels
#include <stdio.h>
#include <assert.h>

#define NUM_ELEMENTS 512


// **===-----------------------------------------------------------------===**
//! @param g_idata  input data in global memory
//                  result is expected in index 0 of g_idata
//! @param n        input number of elements to scan from input data
// **===-----------------------------------------------------------------===**

__global__ void reduction(float *g_data, int n)
{
  int stride;
  // Define shared memory
  __shared__ float scratch[NUM_ELEMENTS];

  // Load the shared memory
  scratch[threadIdx.x ] = g_data[threadIdx.x];
  if(threadIdx.x + blockDim.x < n)
    scratch[threadIdx.x + blockDim.x] = g_data[threadIdx.x + blockDim.x];
  __syncthreads();

  // Do sum reduction from shared memory
```

```
  for(stride = NUM_ELEMENTS/2 ; stride >= 1; stride >>= 1)
  {
      if(threadIdx.x < stride)
         scratch[threadIdx.x] += scratch[threadIdx.x + stride];
      __syncthreads();
  }

  // Store results back to global memory
  if(threadIdx.x == 0)
    g_data[0] = scratch[0];

  return;
}


////////////////////////////////////////////////////////////////////////////////
// Program main
////////////////////////////////////////////////////////////////////////////////
void runTest( int argc, char** argv);
float computeOnDevice(float* h_data, int array_mem_size);
extern "C" void computeGold( float* reference, float* idata, const unsigned int len

int main( int argc, char** argv)
{
    runTest( argc, argv);
    return EXIT_SUCCESS;
}

////////////////////////////////////////////////////////////////////////////////
//! Run naive scan test
////////////////////////////////////////////////////////////////////////////////
void runTest( int argc, char** argv)
{
    int num_elements = NUM_ELEMENTS;

    const unsigned int array_mem_size = sizeof( float) * num_elements;

    // allocate host memory to store the input data
    float* h_data = (float*) malloc( array_mem_size);

    // * No arguments: Randomly generate input data and compare against the host's

            // initialize the input data on the host to be integer values
            // between 0 and 1000
            for( unsigned int i = 0; i < num_elements; ++i)
            {
                //h_data[i] = floorf(1000*(rand()/(float)RAND_MAX));
                h_data[i] = i*1.0;

            }

        // compute reference solution
    float reference = 0.0f;
    computeGold(&reference , h_data, num_elements);
```

```
    float result = computeOnDevice(h_data, num_elements);

    // We can use an epsilon of 0 since values are integral and in a range
    // that can be exactly represented
    float epsilon = 0.0f;
    unsigned int result_regtest = (abs(result - reference) <= epsilon);
    printf( "Test %s\n", (1 == result_regtest) ? "PASSED" : "FAILED");
    printf( "device: %f  host: %f\n", result, reference);
    // cleanup memory
    free( h_data);
}

////////////////////////////////////////////////////////////////////////////
// Take h_data from host, copies it to device, setup grid and thread
// dimentions, excutes kernel function, and copy result of scan back
// to h_data.
// Note: float* h_data is both the input and the output of this function.
////////////////////////////////////////////////////////////////////////////


float computeOnDevice(float* h_data, int num_elements)
{
  float* d_data = NULL;
  float result;

  // Memory allocation on device side
  cudaMalloc((void**)&d_data, num_elements*sizeof(float));

  // Copy from host memory to device memory
  cudaMemcpy(d_data, h_data, num_elements*sizeof(float), cudaMemcpyHostToDevice);

  int threads = (num_elements/2) + num_elements%2;

  // Invoke the kernel
  reduction<<<1,threads>>>(d_data,num_elements);

  // Copy from device memory back to host memory
  cudaMemcpy(&result, d_data, sizeof(float), cudaMemcpyDeviceToHost);
  cudaFree(d_data);
  return result;
}

////////////////////////////////////////////////////////////////////////////
void computeGold( float* reference, float* idata, const unsigned int len)
{
  reference[0] = 0;
  double total_sum = 0;
  unsigned int i;
  for( i = 0; i < len; ++i)
  {
      total_sum += idata[i];
  }
  reference[0] = total_sum;
}
////////////////////////////////////////////////////////////////////////////
```

```
//////////////////////////////////////////////////////////////////////
```

```
    Writing labsolReduction1.cu
```

```
!/usr/local/cuda/bin/nvcc -I /usr/local/cuda/samples/common/inc -arch=sm_35 -rdc=tr
```

```
!ls
```

```
    lab1Reduction     lab1solReduction     labsolReduction1     sample_data
    lab1Reduction.cu  lab1solReduction.cu  labsolReduction1.cu
```

```
!./labsolReduction1
```

```
    Test PASSED
    device: 523776.000000  host: 523776.000000
```

```
!nvprof ./labsolReduction1
```

```
    ==1123== NVPROF is profiling process 1123, command: ./labsolReduction1
    Test PASSED
    device: 523776.000000  host: 523776.000000
    ==1123== Profiling application: ./labsolReduction1
    ==1123== Profiling result:
                Type  Time(%)      Time     Calls       Avg       Min       Max  N
     GPU activities:   56.94%  5.1200us         1  5.1200us  5.1200us  5.1200us  r
                       22.78%  2.0480us         1  2.0480us  2.0480us  2.0480us  [
                       20.28%  1.8240us         1  1.8240us  1.8240us  1.8240us  [
          API calls:   99.63%  196.11ms         1  196.11ms  196.11ms  196.11ms  c
                        0.20%  398.73us         1  398.73us  398.73us  398.73us  c
                        0.08%  153.97us        97  1.5870us     143ns  59.096us  c
                        0.04%  77.037us         1  77.037us  77.037us  77.037us  c
                        0.02%  35.271us         2  17.635us  14.457us  20.814us  c
                        0.01%  27.427us         1  27.427us  27.427us  27.427us  c
                        0.01%  24.317us         1  24.317us  24.317us  24.317us  c
                        0.00%  3.3170us         1  3.3170us  3.3170us  3.3170us  c
                        0.00%  2.0920us         3     697ns     162ns  1.3160us  c
                        0.00%  1.4470us         2     723ns     304ns  1.1430us  c
                        0.00%     269ns         1     269ns     269ns     269ns  c
```

CASO2

```
%%writefile labsolReduction2.cu

// includes, kernels
#include <stdio.h>
#include <assert.h>

#define NUM_ELEMENTS 512
```

```
// **===-----------------------------------------------------------------===**
//! @param g_idata  input data in global memory
//                  result is expected in index 0 of g_idata
//! @param n        input number of elements to scan from input data
// **===-----------------------------------------------------------------===**

__global__ void reduction(float *g_data, int n)
{
  int stride;
  // Define shared memory
  __shared__ float scratch[NUM_ELEMENTS];

  // Load the shared memory
  scratch[threadIdx.x ] = g_data[threadIdx.x];
  if(threadIdx.x + blockDim.x < n)
    scratch[threadIdx.x + blockDim.x] = g_data[threadIdx.x + blockDim.x];
  __syncthreads();

  // Do sum reduction from shared memory
  //Reduction scheme 2
  for (stride = NUM_ELEMENTS; stride > 1; stride >>= 1) {
         __syncthreads();
        if (threadIdx.x < (stride>>1)) {
        scratch[threadIdx.x] += scratch[stride - threadIdx.x - 1];
      }
  }
  __syncthreads();

  // Store results back to global memory
  if(threadIdx.x == 0)
    g_data[0] = scratch[0];

  return;
}

////////////////////////////////////////////////////////////////////////////////
// Program main
////////////////////////////////////////////////////////////////////////////////
void runTest( int argc, char** argv);
float computeOnDevice(float* h_data, int array_mem_size);
extern "C" void computeGold( float* reference, float* idata, const unsigned int len

int main( int argc, char** argv)
{
    runTest( argc, argv);
    return EXIT_SUCCESS;
}

////////////////////////////////////////////////////////////////////////////////
//! Run naive scan test
////////////////////////////////////////////////////////////////////////////////
void runTest( int argc, char** argv)
{
    int num_elements = NUM_ELEMENTS;
```

```
    const unsigned int array_mem_size = sizeof( float) * num_elements;

    // allocate host memory to store the input data
    float* h_data = (float*) malloc( array_mem_size);

    // * No arguments: Randomly generate input data and compare against the host's

            // initialize the input data on the host to be integer values
            // between 0 and 1000
            for( unsigned int i = 0; i < num_elements; ++i)
            {
                //h_data[i] = floorf(1000*(rand()/(float)RAND_MAX));
                h_data[i] = i*1.0;

            }

        // compute reference solution
    float reference = 0.0f;
    computeGold(&reference , h_data, num_elements);

    float result = computeOnDevice(h_data, num_elements);

    // We can use an epsilon of 0 since values are integral and in a range
    // that can be exactly represented
    float epsilon = 0.0f;
    unsigned int result_regtest = (abs(result - reference) <= epsilon);
    printf( "Test %s\n", (1 == result_regtest) ? "PASSED" : "FAILED");
    printf( "device: %f  host: %f\n", result, reference);
    // cleanup memory
    free( h_data);
}

////////////////////////////////////////////////////////////////////////////
// Take h_data from host, copies it to device, setup grid and thread
// dimentions, excutes kernel function, and copy result of scan back
// to h_data.
// Note: float* h_data is both the input and the output of this function.
////////////////////////////////////////////////////////////////////////////

float computeOnDevice(float* h_data, int num_elements)
{
  float* d_data = NULL;
  float result;

  // Memory allocation on device side
  cudaMalloc((void**)&d_data, num_elements*sizeof(float));

  // Copy from host memory to device memory
  cudaMemcpy(d_data, h_data, num_elements*sizeof(float), cudaMemcpyHostToDevice);

  int threads = (num_elements/2) + num_elements%2;

  // Invoke the kernel
  reduction<<<1,threads>>>(d_data,num_elements);
```

```
  // Copy from device memory back to host memory
  cudaMemcpy(&result, d_data, sizeof(float), cudaMemcpyDeviceToHost);
  cudaFree(d_data);
  return result;
}

////////////////////////////////////////////////////////////////////////
void computeGold( float* reference, float* idata, const unsigned int len)
{
  reference[0] = 0;
  double total_sum = 0;
  unsigned int i;
  for( i = 0; i < len; ++i)
  {
      total_sum += idata[i];
  }
  reference[0] = total_sum;
}
////////////////////////////////////////////////////////////////////////
```

```
Overwriting labsolReduction2.cu
```

```
!/usr/local/cuda/bin/nvcc -I /usr/local/cuda/samples/common/inc -arch=sm_35 -rdc=tr
```

```
!ls -la
```

```
total 3280
drwxr-xr-x 1 root root   4096 Oct 22 19:54 .
drwxr-xr-x 1 root root   4096 Oct 22 18:46 ..
drwxr-xr-x 1 root root   4096 Oct 14 16:32 .config
-rwxr-xr-x 1 root root 659784 Oct 22 18:57 lab1Reduction
-rw-r--r-- 1 root root   6837 Oct 22 18:54 lab1Reduction.cu
-rwxr-xr-x 1 root root 655504 Oct 22 19:43 lab1solReduction
-rw-r--r-- 1 root root   5121 Oct 22 19:30 lab1solReduction.cu
-rwxr-xr-x 1 root root 655504 Oct 22 19:37 labsolReduction1
-rw-r--r-- 1 root root   4338 Oct 22 19:35 labsolReduction1.cu
-rwxr-xr-x 1 root root 655504 Oct 22 19:54 labsolReduction2
-rw-r--r-- 1 root root   4371 Oct 22 19:45 labsolReduction2.cu
-rwxr-xr-x 1 root root 655504 Oct 22 19:50 labsolReduction3
-rw-r--r-- 1 root root   4408 Oct 22 19:49 labsolReduction3.cu
drwxr-xr-x 1 root root   4096 Oct 14 16:31 sample_data
```

```
!./labsolReduction2
```

```
Test FAILED
device: 81760.000000  host: 130816.000000
```

```
!nvprof ./labsolReduction2
```

```
==1931== NVPROF is profiling process 1931, command: ./labsolReduction2
Test PASSED
device: 130816.000000  host: 130816.000000
==1931== Profiling application: ./labsolReduction2
==1931== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  N
 GPU activities:   53.54%  4.3520us         1  4.3520us  4.3520us  4.3520us  r
                   23.62%  1.9200us         1  1.9200us  1.9200us  1.9200us  [
                   22.83%  1.8560us         1  1.8560us  1.8560us  1.8560us  [
      API calls:   99.67%  228.07ms         1  228.07ms  228.07ms  228.07ms  c
                    0.17%  380.32us         1  380.32us  380.32us  380.32us  c
                    0.06%  145.35us        97  1.4980us     151ns  60.700us  c
                    0.05%  114.10us         1  114.10us  114.10us  114.10us  c
                    0.02%  52.420us         2  26.210us  19.272us  33.148us  c
                    0.01%  27.815us         1  27.815us  27.815us  27.815us  c
                    0.01%  26.198us         1  26.198us  26.198us  26.198us  c
                    0.00%  3.5080us         1  3.5080us  3.5080us  3.5080us  c
```

CASO3

```
%%writefile labsolReduction3.cu

// includes, kernels
#include <stdio.h>
#include <assert.h>

#define NUM_ELEMENTS 512


// **===-----------------------------------------------------------------===**
//! @param g_idata  input data in global memory
//                  result is expected in index 0 of g_idata
//! @param n        input number of elements to scan from input data
// **===-----------------------------------------------------------------===**

__global__ void reduction(float *g_data, int n)
{
  int stride;
  // Define shared memory
  __shared__ float scratch[NUM_ELEMENTS];

  // Load the shared memory
  scratch[threadIdx.x ] = g_data[threadIdx.x];
  if(threadIdx.x + blockDim.x < n)
    scratch[threadIdx.x + blockDim.x] = g_data[threadIdx.x + blockDim.x];
  __syncthreads();

  // Do sum reduction from shared memory

  ///////  Reduction scheme 3 //////
  int stride2;
  for (stride = 1, stride2=1; stride <= 9; stride++, stride2<<=1)
  {
      int t = threadIdx.x << stride;
      if (t  + stride2 < n)
            scratch[ t ] = scratch[ t ]  + scratch[ t + stride2];
```

```
    __syncthreads();
  }

  // Store results back to global memory
  if(threadIdx.x == 0)
    g_data[0] = scratch[0];

  return;
}
////////////////////////////////////////////////////////////////////////////////
// Program main
////////////////////////////////////////////////////////////////////////////////
void runTest( int argc, char** argv);
float computeOnDevice(float* h_data, int array_mem_size);
extern "C" void computeGold( float* reference, float* idata, const unsigned int len

int main( int argc, char** argv)
{
    runTest( argc, argv);
    return EXIT_SUCCESS;
}

////////////////////////////////////////////////////////////////////////////////
//! Run naive scan test
////////////////////////////////////////////////////////////////////////////////
void runTest( int argc, char** argv)
{
    int num_elements = NUM_ELEMENTS;

    const unsigned int array_mem_size = sizeof( float) * num_elements;

    // allocate host memory to store the input data
    float* h_data = (float*) malloc( array_mem_size);

    // * No arguments: Randomly generate input data and compare against the host's

            // initialize the input data on the host to be integer values
            // between 0 and 1000
            for( unsigned int i = 0; i < num_elements; ++i)
            {
                //h_data[i] = floorf(1000*(rand()/(float)RAND_MAX));
                h_data[i] = i*1.0;

            }

        // compute reference solution
    float reference = 0.0f;
    computeGold(&reference , h_data, num_elements);

    float result = computeOnDevice(h_data, num_elements);

    // We can use an epsilon of 0 since values are integral and in a range
    // that can be exactly represented
    float epsilon = 0.0f;
    unsigned int result_regtest = (abs(result - reference) <= epsilon);
```

```
    printf( "Test %s\n", (1 == result_regtest) ? "PASSED" : "FAILED");
    printf( "device: %f  host: %f\n", result, reference);
    // cleanup memory
    free( h_data);
}

////////////////////////////////////////////////////////////////////////////
// Take h_data from host, copies it to device, setup grid and thread
// dimentions, excutes kernel function, and copy result of scan back
// to h_data.
// Note: float* h_data is both the input and the output of this function.
////////////////////////////////////////////////////////////////////////////

float computeOnDevice(float* h_data, int num_elements)
{
  float* d_data = NULL;
  float result;

  // Memory allocation on device side
  cudaMalloc((void**)&d_data, num_elements*sizeof(float));

  // Copy from host memory to device memory
  cudaMemcpy(d_data, h_data, num_elements*sizeof(float), cudaMemcpyHostToDevice);

  int threads = (num_elements/2) + num_elements%2;

  // Invoke the kernel
  reduction<<<1,threads>>>(d_data,num_elements);

  // Copy from device memory back to host memory
  cudaMemcpy(&result, d_data, sizeof(float), cudaMemcpyDeviceToHost);
  cudaFree(d_data);
  return result;
}

////////////////////////////////////////////////////////////////////////////
void computeGold( float* reference, float* idata, const unsigned int len)
{
  reference[0] = 0;
  double total_sum = 0;
  unsigned int i;
  for( i = 0; i < len; ++i)
  {
      total_sum += idata[i];
  }
  reference[0] = total_sum;
}
////////////////////////////////////////////////////////////////////////////
```

```
Overwriting labsolReduction3.cu
```

```
!/usr/local/cuda/bin/nvcc -I /usr/local/cuda/samples/common/inc -arch=sm_35 -rdc=tr
```

```
!ls -la
```

```
!./labsolReduction3
```

```
    Test PASSED
    device: 130816.000000  host: 130816.000000
```

```
!nvprof ./labsolReduction3
```

```
    ==2116== NVPROF is profiling process 2116, command: ./labsolReduction3
    Test PASSED
    device: 130816.000000  host: 130816.000000
    ==2116== Profiling application: ./labsolReduction3
    ==2116== Profiling result:
                Type  Time(%)      Time     Calls       Avg       Min       Max  N
     GPU activities:   54.41%  4.5440us         1  4.5440us  4.5440us  4.5440us  r
                       23.37%  1.9520us         1  1.9520us  1.9520us  1.9520us  [
                       22.22%  1.8560us         1  1.8560us  1.8560us  1.8560us  [
          API calls:   99.62%  191.23ms         1  191.23ms  191.23ms  191.23ms  c
                        0.20%  378.14us         1  378.14us  378.14us  378.14us  c
                        0.09%  166.79us        97  1.7190us     157ns  67.113us  c
                        0.04%  78.328us         1  78.328us  78.328us  78.328us  c
                        0.02%  35.975us         2  17.987us  15.838us  20.137us  c
                        0.02%  29.526us         1  29.526us  29.526us  29.526us  c
                        0.01%  27.831us         1  27.831us  27.831us  27.831us  c
                        0.00%  3.6840us         1  3.6840us  3.6840us  3.6840us  c
                        0.00%  1.7880us         3     596ns     155ns  1.2790us  c
                        0.00%  1.0180us         2     509ns     326ns     692ns  c
                        0.00%     327ns         1     327ns     327ns     327ns  c
```

```
!nvprof ./labsolReduction2
```

```
    ==2092== NVPROF is profiling process 2092, command: ./labsolReduction2
    Test PASSED
    device: 130816.000000  host: 130816.000000
    ==2092== Profiling application: ./labsolReduction2
    ==2092== Profiling result:
                Type  Time(%)      Time     Calls       Avg       Min       Max  N
     GPU activities:   53.75%  4.3520us         1  4.3520us  4.3520us  4.3520us  r
                       23.32%  1.8880us         1  1.8880us  1.8880us  1.8880us  [
                       22.92%  1.8560us         1  1.8560us  1.8560us  1.8560us  [
          API calls:   99.64%  194.61ms         1  194.61ms  194.61ms  194.61ms  c
                        0.19%  364.12us         1  364.12us  364.12us  364.12us  c
                        0.08%  161.47us        97  1.6640us     150ns  71.381us  c
                        0.04%  77.930us         1  77.930us  77.930us  77.930us  c
                        0.02%  38.405us         2  19.202us  18.295us  20.110us  c
                        0.02%  29.673us         1  29.673us  29.673us  29.673us  c
                        0.01%  26.666us         1  26.666us  26.666us  26.666us  c
                        0.00%  3.3290us         1  3.3290us  3.3290us  3.3290us  c
                        0.00%  2.3010us         3     767ns     165ns  1.6380us  c
                        0.00%  1.2510us         2     625ns     342ns     909ns  c
                        0.00%     330ns         1     330ns     330ns     330ns  c
```

```
!nvprof ./labsolReduction1
```

```
==2103== NVPROF is profiling process 2103, command: ./labsolReduction1
Test PASSED
device: 523776.000000  host: 523776.000000
==2103== Profiling application: ./labsolReduction1
==2103== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  N
 GPU activities:   57.04%  5.1840us         1  5.1840us  5.1840us  5.1840us  r
                   22.89%  2.0800us         1  2.0800us  2.0800us  2.0800us  [
                   20.07%  1.8240us         1  1.8240us  1.8240us  1.8240us  [
      API calls:   99.63%  204.88ms         1  204.88ms  204.88ms  204.88ms  c
                    0.19%  398.08us         1  398.08us  398.08us  398.08us  c
                    0.07%  149.94us        97  1.5450us     155ns  63.211us  c
                    0.05%  95.307us         1  95.307us  95.307us  95.307us  c
                    0.02%  48.822us         2  24.411us  16.850us  31.972us  c
                    0.01%  27.708us         1  27.708us  27.708us  27.708us  c
                    0.01%  26.020us         1  26.020us  26.020us  26.020us  c
                    0.00%  3.4330us         1  3.4330us  3.4330us  3.4330us  c
                    0.00%  1.8310us         3     610ns     176ns  1.2030us  c
                    0.00%  1.3370us         2     668ns     259ns  1.0780us  c
                    0.00%     318ns         1     318ns     318ns     318ns  c
```