GPU Computation for the Masses

Abdul Dakkak

Purpose

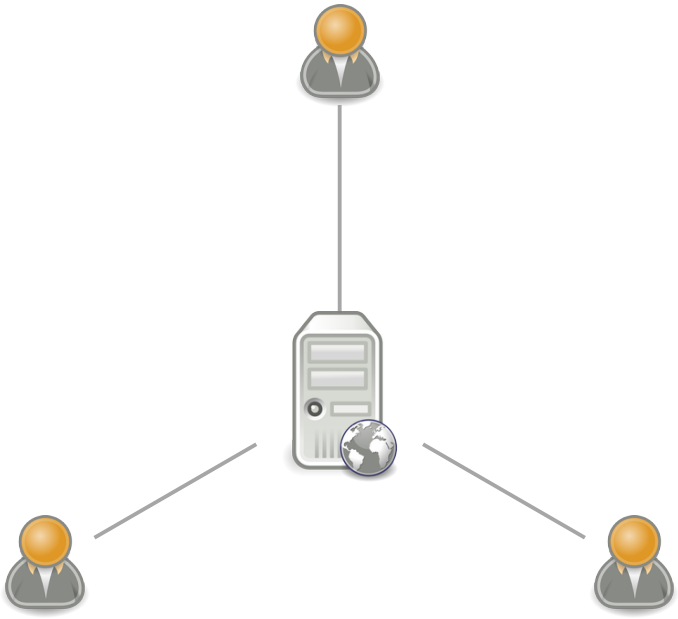
Cannot assume a user has a GPU/development environment

or a C compiler for that matter. These people should be able to participate

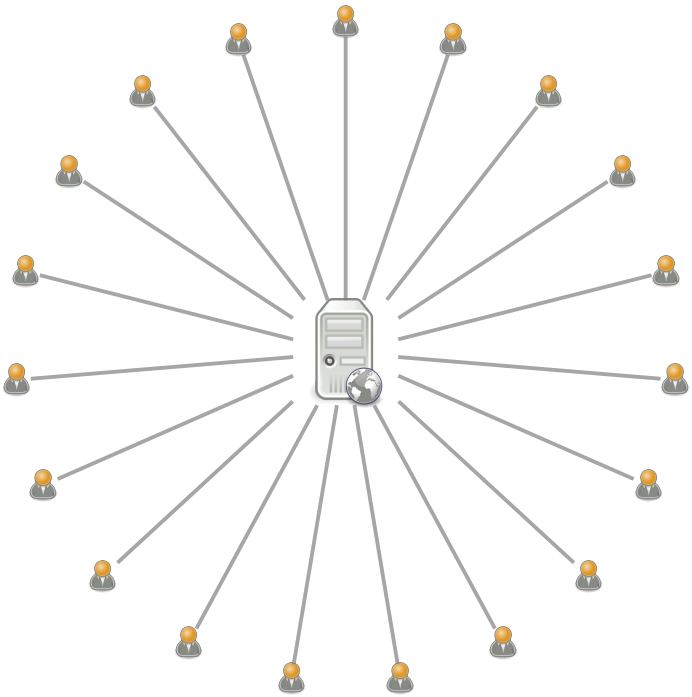
Most of use do not have a C compiler or CUDA on most of our machines, but we still might want to use them to program

Constraint 1: Support a bazillion users

Should be able to scale from a few users



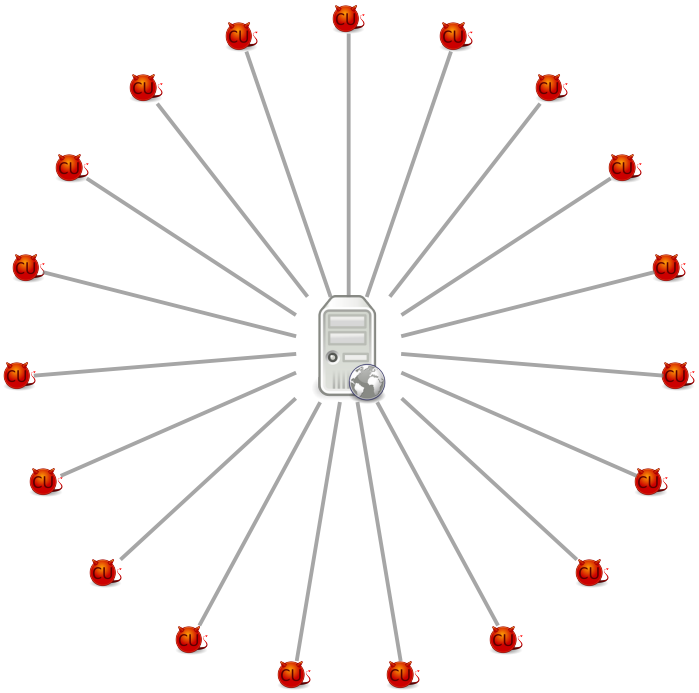
to thousands



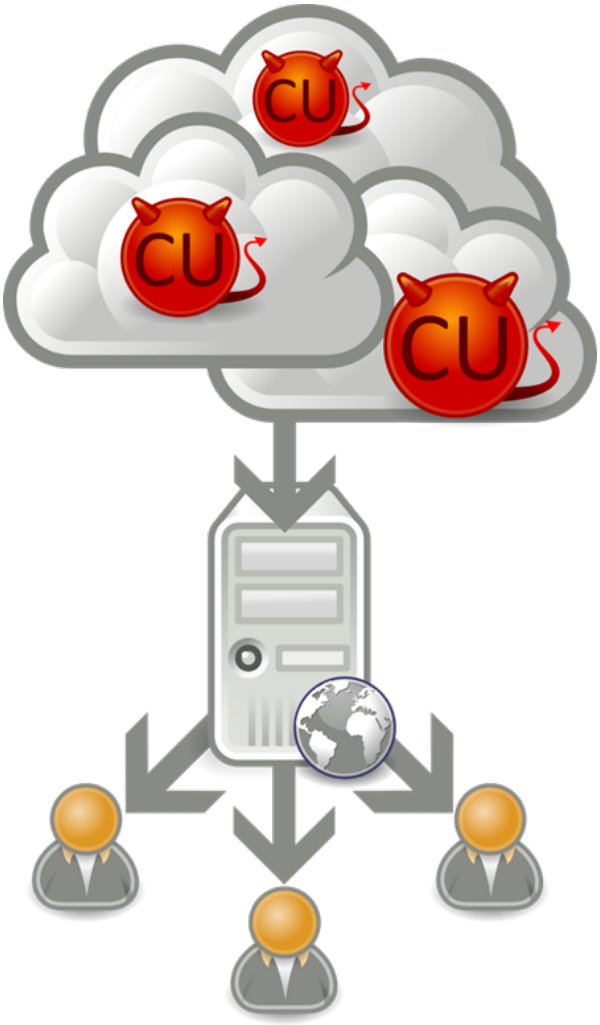
Constraint 2: Support to have a dynamic number of worker nodes

the server is supposed to support one daemon (slave worker)

or thousands

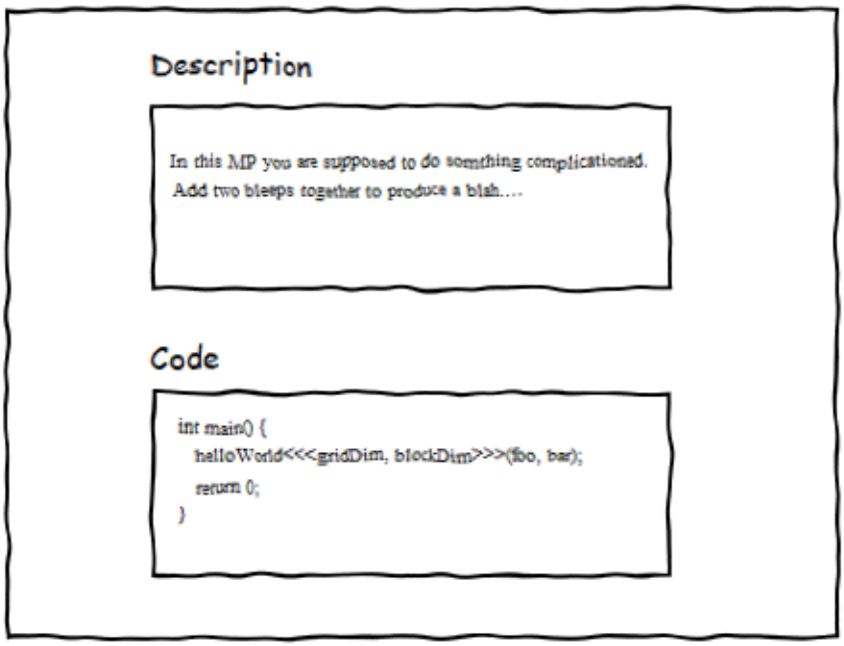


Architecture



Constraint 3: Supposed to be user friendly

The course is about learning how to program CUDA, not about how to login to a system and use ssh



Users worry about writing their code, nothing else.

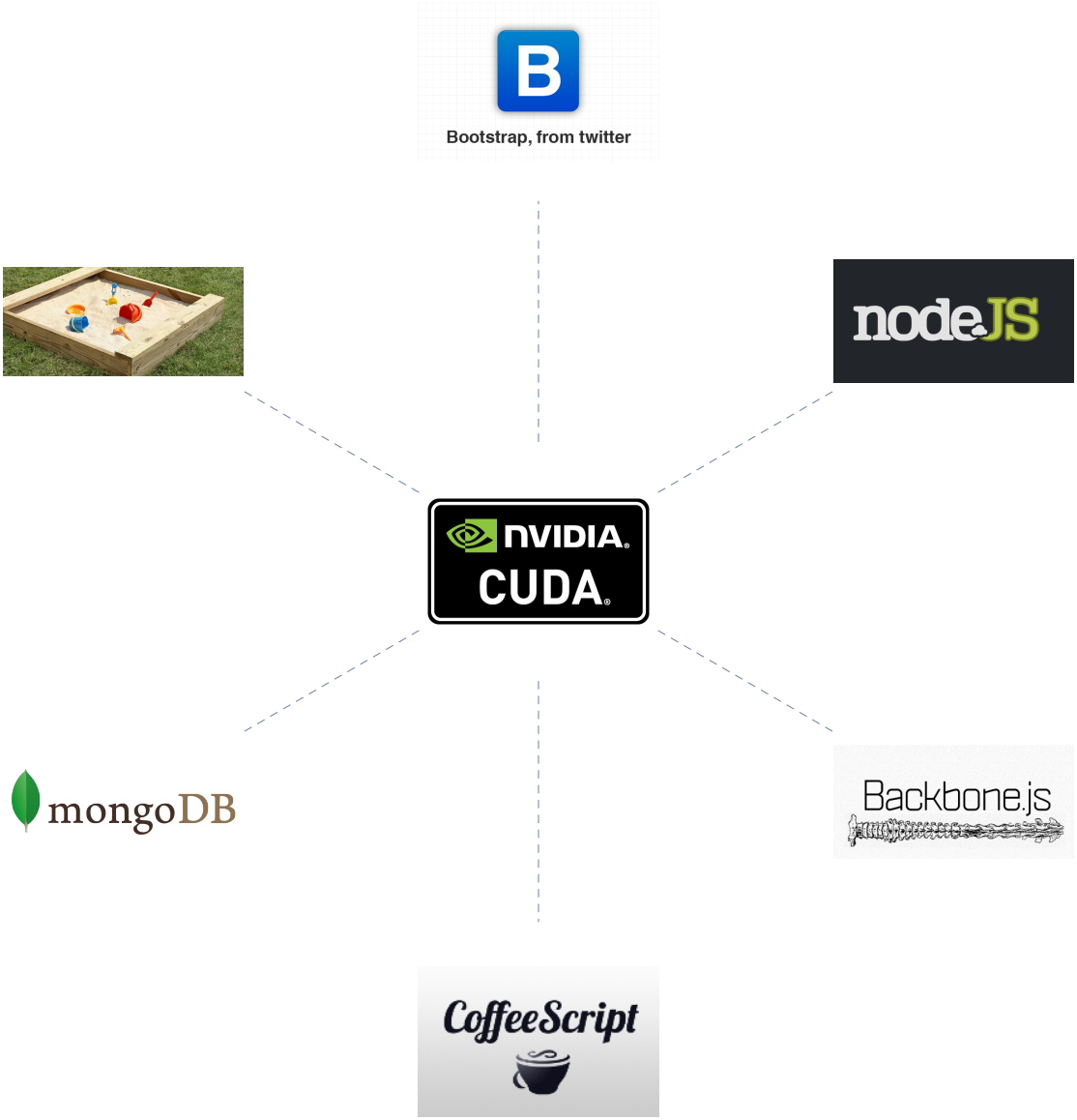
Constraint 4: Supposed to be low risk for deployers

A person should not comprimise the system or interfear with others' code.

Demo

Technical Details

A collection of buzz words



Bootstrap

Provides use with cross platform CSS and Javascript GUI elements for the website's frontend

Backbone.js

Provides use with a way to have MVC on the client side

CUDA

What we are teaching!!!!

NodeJS

Both the server and worker daemon are written in Node and CoffeeScript

CoffeeScript

Coffeescript is a language that compiles to Javascript

Mongodb (will be switched with mysql)

A NoSQL database (will be switched with a SQL database)

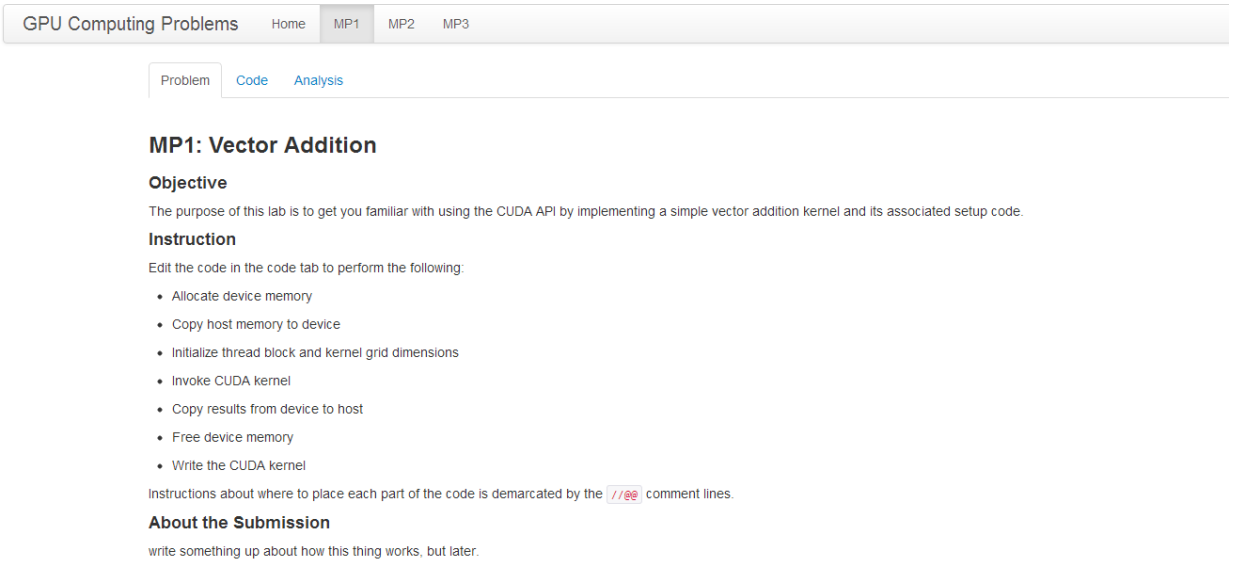
Sandbox

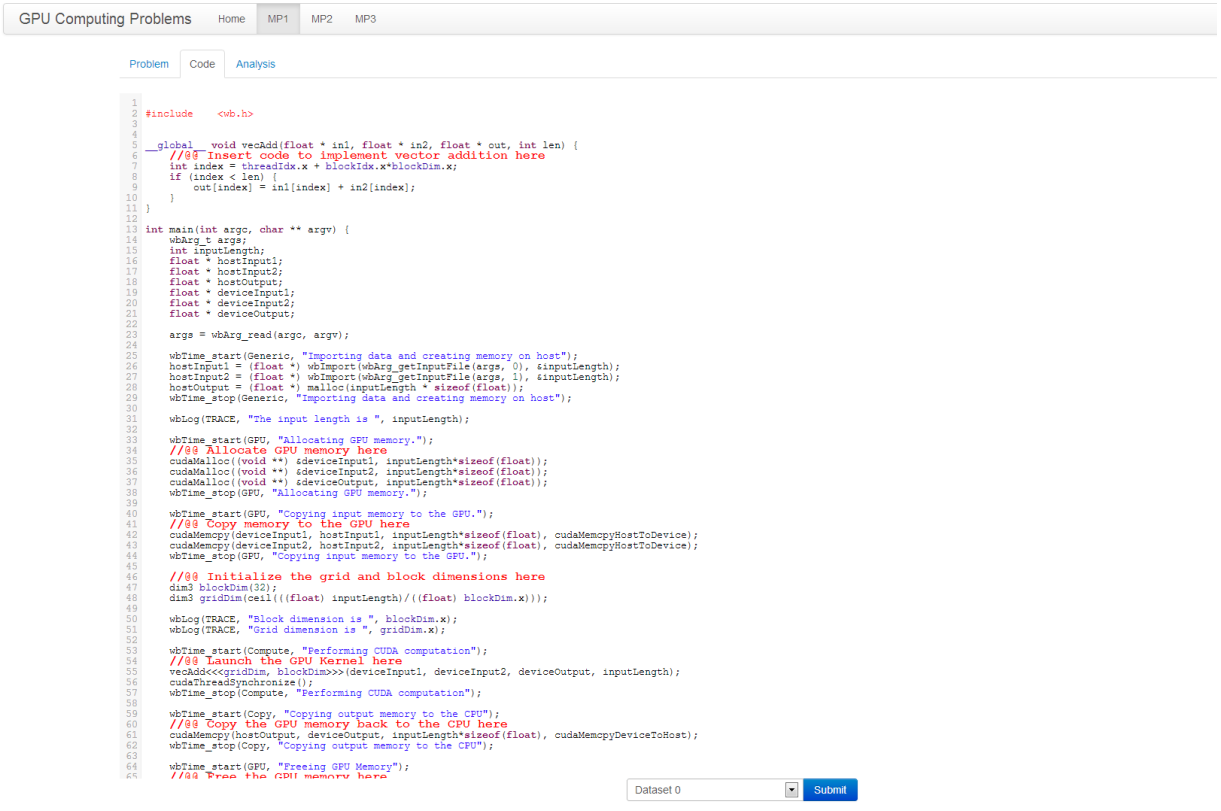
We use a combination of syscall captures and program analysis to make sure nothing malicious is being compiled/executed

Features

- Login system
- Grading system
- Decentralized
- Sandboxed
- Cross Platform (even on windows)
- Dynamic scheduler

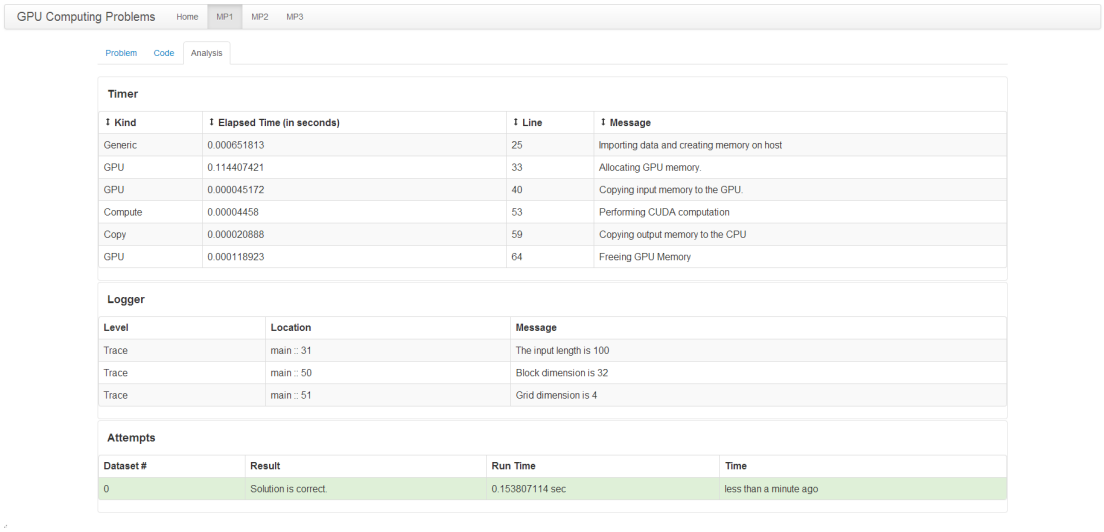
Demo


GPU Computing Problems
Home
MP1
MP2
MP3
Problem
Code
Analysis
MP1: Vector Addition
Objective
The purpose of this lab is to get you familiar with using the CUDA API by implementing a simple vector addition kernel and its associated setup code.
Instruction
Edit the code in the code tab to perform the following:
Allocate device memory
Copy host memory to device
Initialize thread block and kernel grid dimensions
Invoke CUDA kernel
Copy results from device to host
Free device memory
Write the CUDA kernel
Instructions about where to place each part of the code is demarcated by the //@@ comment lines.
About the Submission
write something up about how this thing works, but later.


GPU Computing Problems
Home
MP1
MP2
MP3
Problem
Code
Analysis
#include <wb.h>
__global__ void vecAdd(float * in1, float * in2, float * out, int len) {
//@@ Insert code to implement vector addition here
int index = threadIdx.x + blockIdx.x*blockDim.x;
if (index < len) {
out[index] = in1[index] + in2[index];
}
}
int main(int argc, char ** argv) {
wbArg_t args;
int inputLength;
float * hostInput1;
float * hostInput2;
float * hostOutput;
float * deviceInput1;
float * deviceInput2;
float * deviceOutput;
args = wbArg_read(argc, argv);
wbTime_start(Generic, "Importing data and creating memory on host");
hostInput1 = (float *) wbImport(wbArg_getInputFile(args, 0), &inputLength);
hostInput2 = (float *) wbImport(wbArg_getInputFile(args, 1), &inputLength);
hostOutput = (float *) malloc(inputLength * sizeof(float));
wbTime_stop(Generic, "Importing data and creating memory on host");
wbLog(TRACE, "The input length is ", inputLength);
wbTime_start(GPU, "Allocating GPU memory.");
//@@ Allocate GPU memory here
cudaMalloc((void **) &deviceInput1, inputLength*sizeof(float));
cudaMalloc((void **) &deviceInput2, inputLength*sizeof(float));
cudaMalloc((void **) &deviceOutput, inputLength*sizeof(float));
wbTime_stop(GPU, "Allocating GPU memory.");
wbTime_start(GPU, "Copying input memory to the GPU.");
//@@ Copy memory to the GPU here
cudaMemcpy(deviceInput1, hostInput1, inputLength*sizeof(float), cudaMemcpyHostToDevice);
cudaMemcpy(deviceInput2, hostInput2, inputLength*sizeof(float), cudaMemcpyHostToDevice);
wbTime_stop(GPU, "Copying input memory to the GPU.");
//@@ Initialize the grid and block dimensions here
dim3 blockDim(32);
dim3 gridDim(ceil(((float) inputLength)/((float) blockDim.x)));
wbLog(TRACE, "Block dimension is ", blockDim.x);
wbLog(TRACE, "Grid dimension is ", gridDim.x);
wbTime_start(Compute, "Performing CUDA computation");
//@@ Launch the GPU Kernel here
vecAdd<<<gridDim, blockDim>>>(deviceInput1, deviceInput2, deviceOutput, inputLength);
cudaThreadSynchronize();
wbTime_stop(Compute, "Performing CUDA computation");
wbTime_start(Copy, "Copying output memory to the CPU");
//@@ Copy the GPU memory back to the CPU here
cudaMemcpy(hostOutput, deviceOutput, inputLength*sizeof(float), cudaMemcpyDeviceToHost);
wbTime_stop(Copy, "Copying output memory to the CPU");
wbTime_start(GPU, "Freeing GPU Memory");
Dataset 0
Submit

Demo Senarios

Solution is correct



GPU Computing Problems Home MP1 MP2 MP3

Problem Code Analysis

Timer

Kind	Elapsed Time (in seconds)	Line	Message
Generic	0.000651813	25	Importing data and creating memory on host
GPU	0.114407421	33	Allocating GPU memory.
GPU	0.000045172	40	Copying input memory to the GPU.
Compute	0.00004458	53	Performing CUDA computation
Copy	0.000020888	59	Copying output memory to the CPU
GPU	0.000118923	64	Freeing GPU Memory

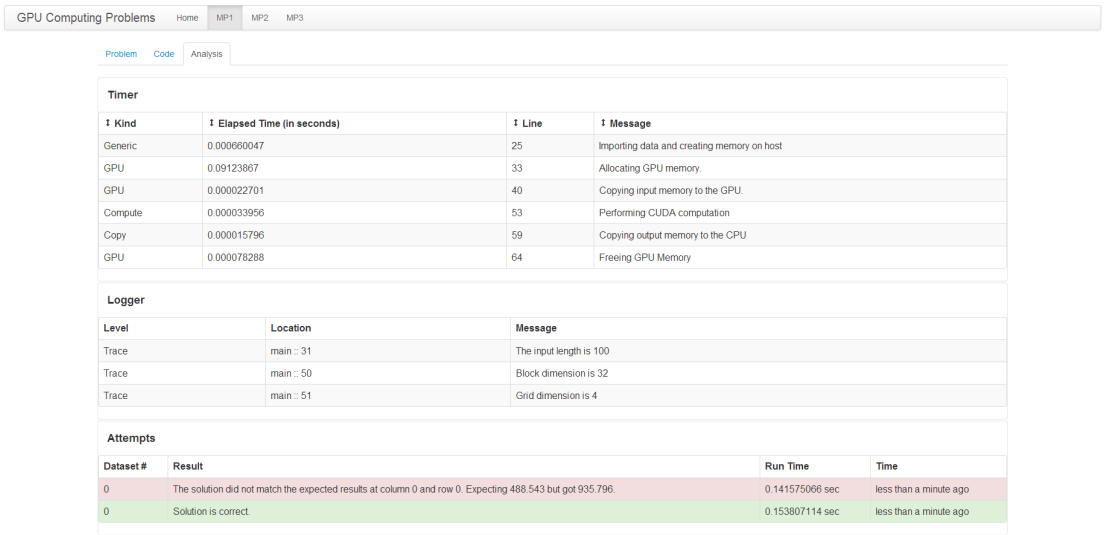
Logger

Level	Location	Message
Trace	main :: 31	The input length is 100
Trace	main :: 50	Block dimension is 32
Trace	main :: 51	Grid dimension is 4

Attempts

Dataset #	Result	Run Time	Time
0	Solution is correct.	0.153807114 sec	less than a minute ago

Solution is incorrect



GPU Computing Problems Home MP1 MP2 MP3

Problem Code Analysis

Timer

Kind	Elapsed Time (in seconds)	Line	Message
Generic	0.000660047	25	Importing data and creating memory on host
GPU	0.09123867	33	Allocating GPU memory.
GPU	0.000022701	40	Copying input memory to the GPU.
Compute	0.000033956	53	Performing CUDA computation
Copy	0.000015796	59	Copying output memory to the CPU
GPU	0.000078288	64	Freeing GPU Memory

Logger

Level	Location	Message
Trace	main :: 31	The input length is 100
Trace	main :: 50	Block dimension is 32
Trace	main :: 51	Grid dimension is 4

Attempts

Dataset #	Result	Run Time	Time
0	The solution did not match the expected results at column 0 and row 0. Expecting 488.543 but got 935.796.	0.141575066 sec	less than a minute ago
0	Solution is correct.	0.153807114 sec	less than a minute ago

Logging and debugging

Logger

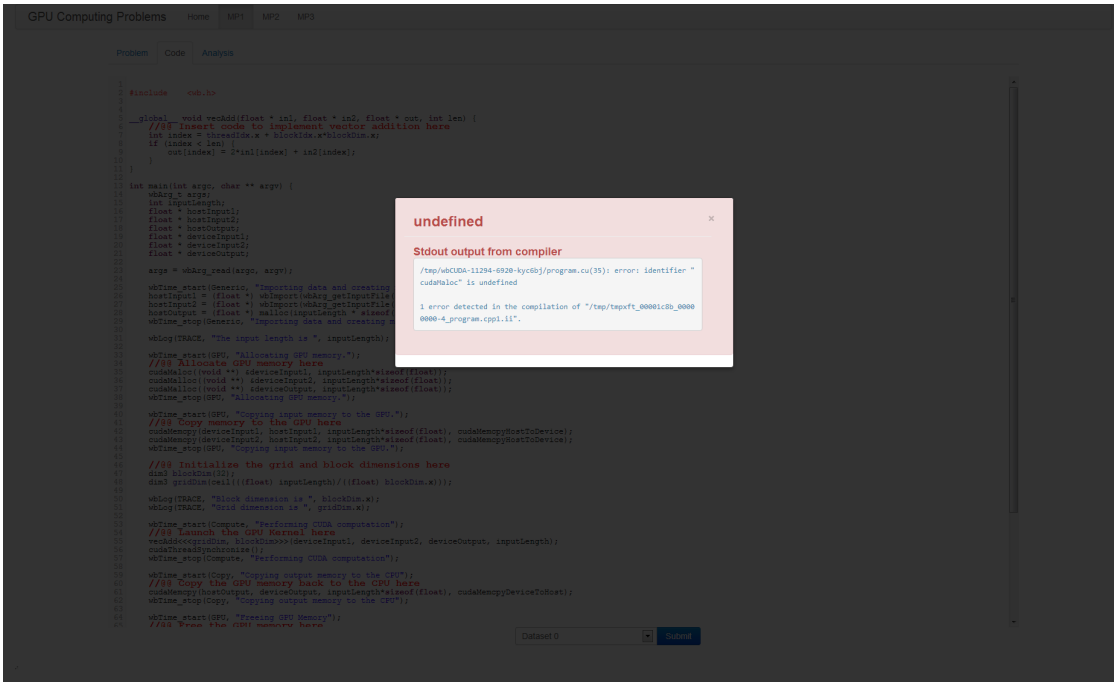
Level	Location	Message
Trace	main :: 26	Read arguments 9
Trace	main :: 35	The input length is 100
Trace	main :: 54	Block dimension is 32
Trace	main :: 55	Grid dimension is 4

Timing

Timer

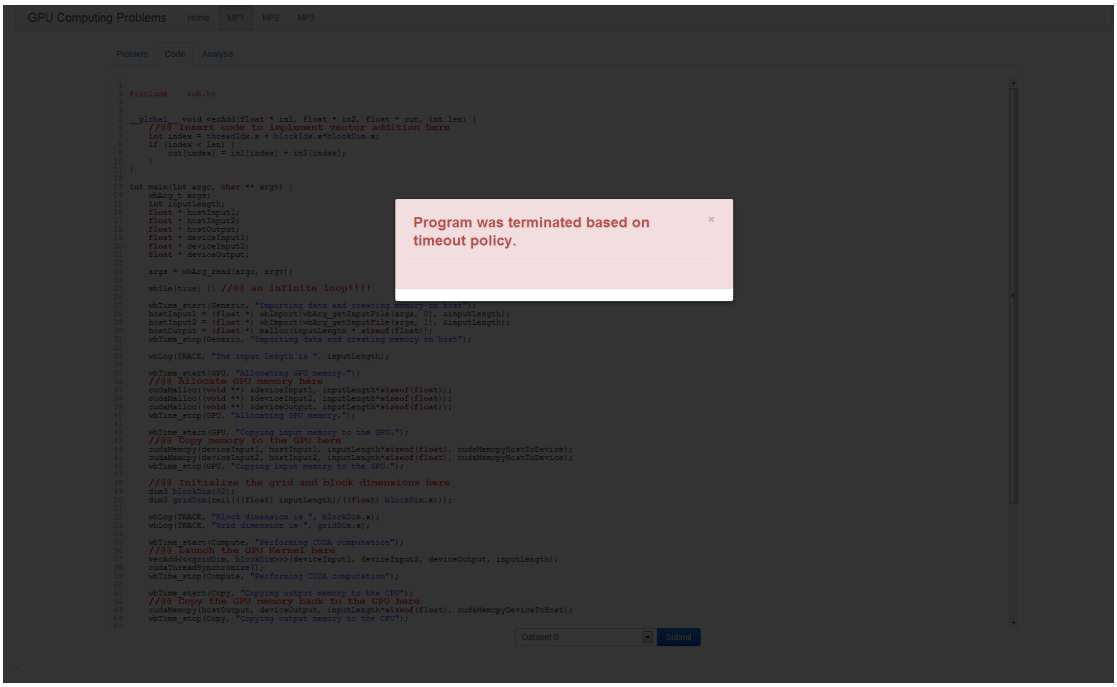
↕ Kind	↕ Elapsed Time (in seconds)	↕ Line	↕ Message
Generic	0.000634308	29	Importing data and creating memory on host
GPU	0.099005559	37	Allocating GPU memory.
GPU	0.00002367	44	Copying input memory to the GPU.
Compute	0.000033975	57	Performing CUDA computation
Copy	0.000016072	63	Copying output memory to the CPU
GPU	0.000076214	68	Freeing GPU Memory

Invalid Synax



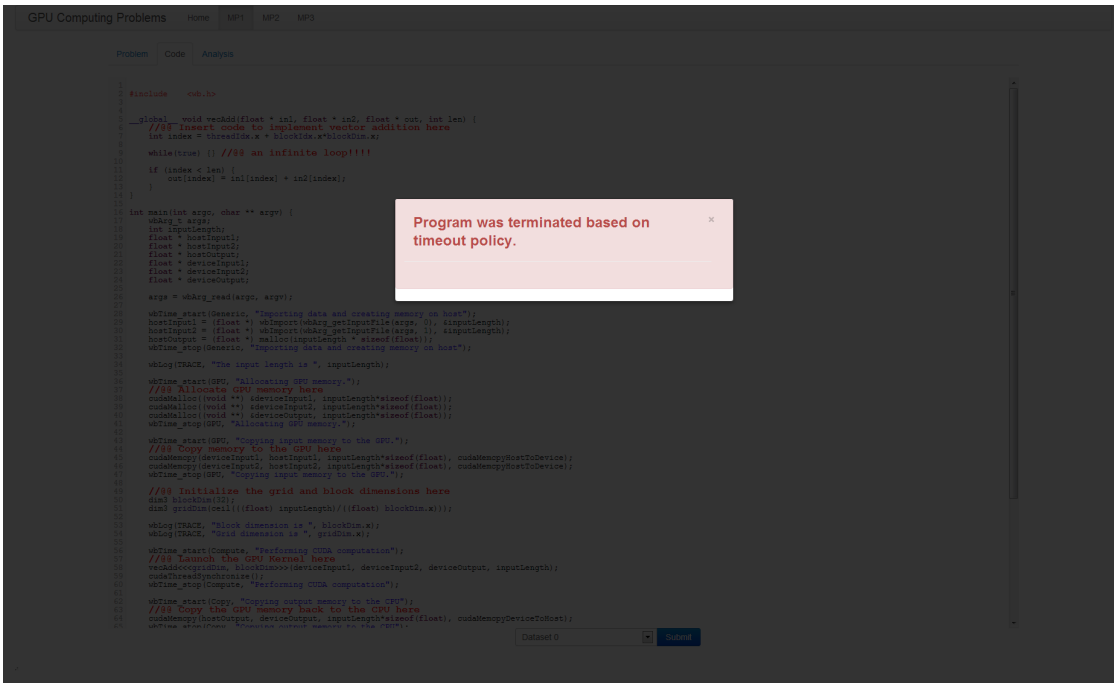
Infinite loop in host code

```
21      float * deviceOutput;
22
23      args = wbArg_read(argc, argv);
24
25      while(true) {} //@@ an infinite loop!!!!
26
27      wbTime_start(Generic, "Importing data and creati
28      hostInput1 = (float *) wbImport(wbArg_getInputFi
29      hostInput2 = (float *) wbImport(wbArg_getInputFi
```



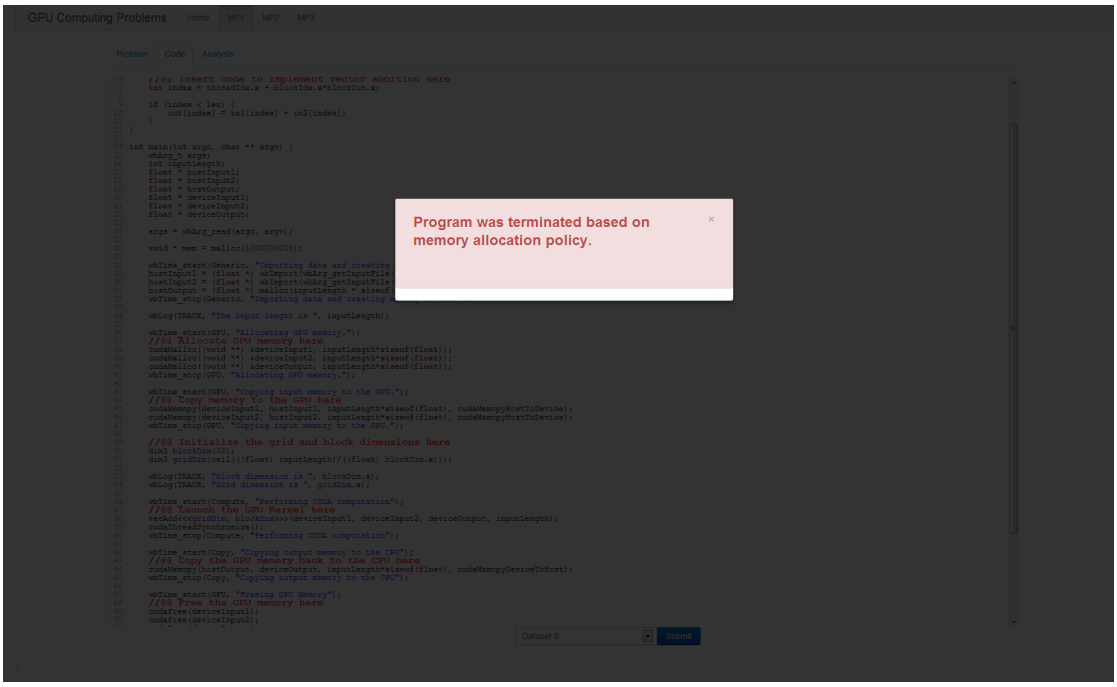
Infinite loop in GPU code

```
5  __global__ void vecAdd(float * in1, float * in2, float *
6      //@@ Insert code to implement vector addit
7      int index = threadIdx.x + blockIdx.x*blockDim.x;
8
9      while(true) {} //@@ an infinite loop!!!!
10
11     if (index < len) {
12         out[index] = in1[index] + in2[index];
13     }
14 }
```



Allocating too much memory in host code

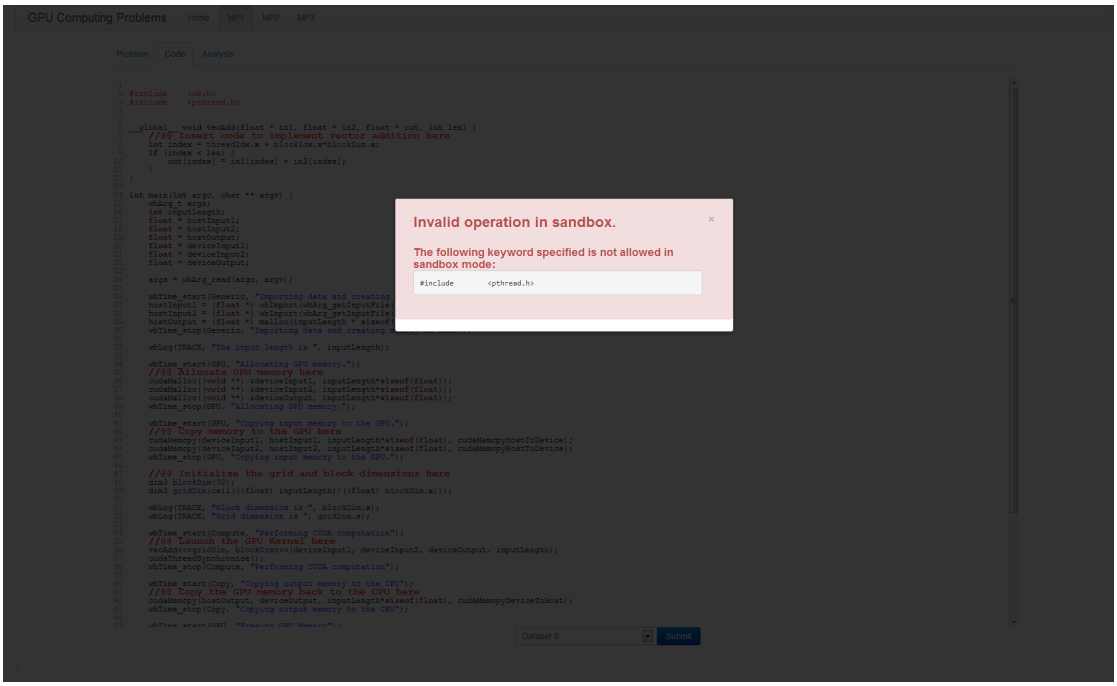
```
25
26      void * mem = malloc(1000000000);
27
28      wbTime_start(Generic, "Importing dat
29      hostInput1 = (float *) wbImport(wbAr
```



Restrict user from adding includes

Problem | Code | Analysis

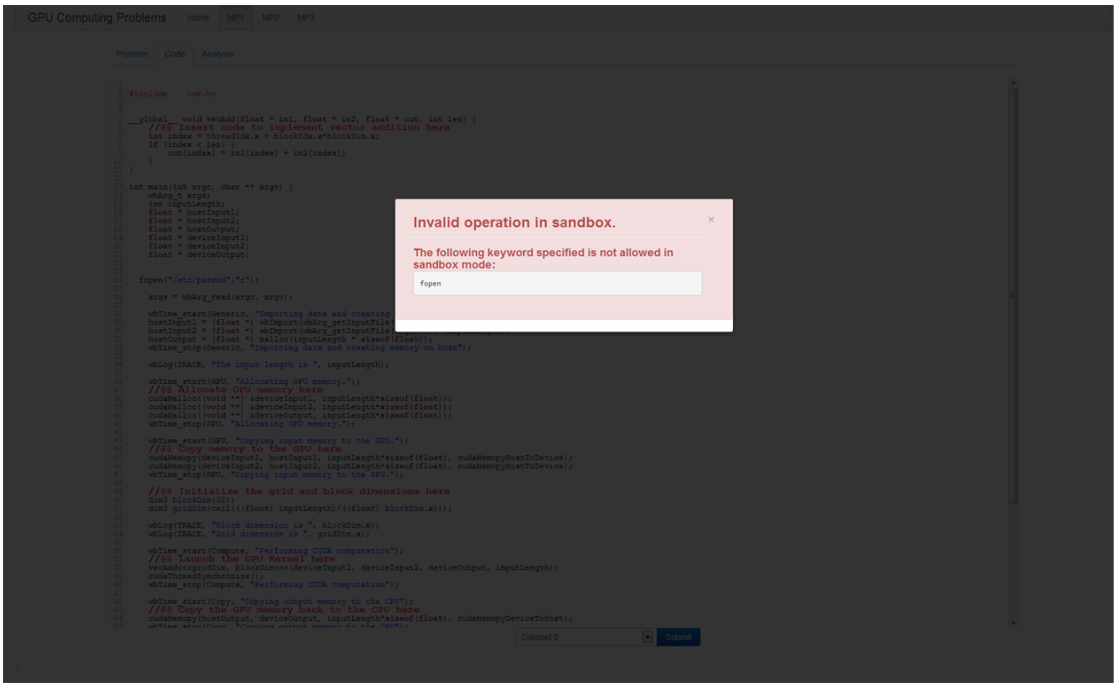
```
1
2  #include    <wb.h>
3  #include    <pthread.h>
4
5
6  __global__ void vecAdd(float
7      //@@ Insert code to
8      int index = threadIdx.x
9      if (index < len) {
```



Restricted syscall in host code

uses a combination of textual filtering and seccomp (linux kernel syscall filtering)

```
22
23
24   fopen("/etc/passwd","r");
25
26     args = wbArg_read(argc, argv);
```



See previous attempts

Attempts

Dataset #	Result	Run Time	Time
2	Solution is correct.	0.201906503 sec	less than a minute ago
1	Solution is correct.	0.142507239 sec	less than a minute ago
0	Solution is correct.	0.141190539 sec	less than a minute ago
	Failed to determine solution	0 sec	less than a minute ago
	Failed to determine solution	0 sec	about a minute ago
	Failed to determine solution	0 sec	2 minutes ago
	Compilation failed	0 sec	2 minutes ago
	Failed to determine solution	0 sec	3 minutes ago
	Failed to determine solution	0 sec	4 minutes ago
	Failed to determine solution	0 sec	6 minutes ago
	Failed to determine solution	0 sec	6 minutes ago
	Failed to determine solution	0 sec	6 minutes ago

Applications

- Teaching GPU Programming
- Remote GPU evaluation from non-accelerated devices
- Batch processing of GPU jobs

Code Breakdown

- ~5kLOC for C support code (timer/logger/solution verifier/…)
- ~1.5kLOC of coffeescript for the daemon
- ~1.5kLOC of coffeescript for the server
- ~1kLOC of javascript for the interface

Extending

Model for running native code in restricted environment

Questions/Discussion