# Lab Submission System

We will be using a web based lab development and submission system through this course. The purpose of this web system is two fold:

1. it allows you to develop CUDA applications without the need to buy and/or setup a CUDA system, and
2. it allows us to grade your submissions more efficiently, uniformly, and promptly.

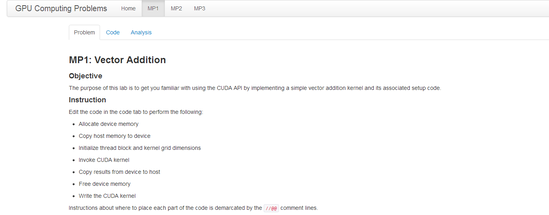
This document describes the online system, walking through the pages that you will see, and describing a sample submission.

## Web System

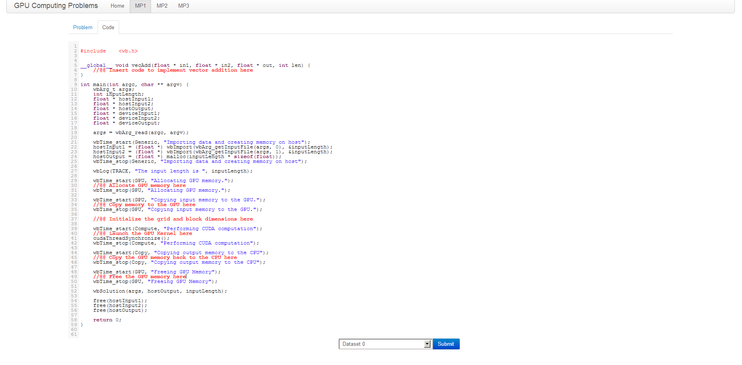
This section describes the interface and system you will be using for this course.

### System Interface

When you login to your account, you are directed to the first lab as shown in the following figure

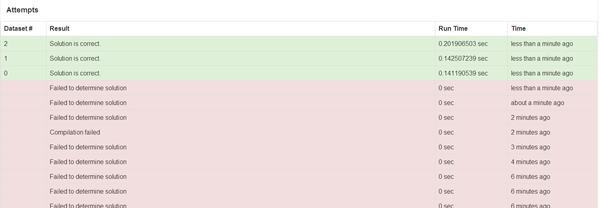
[](/description_images/org/mp1.png)

The lab's objective, instructions, and possibly the grading policy are shown in this tab. The second tab is the code view. This is where you'll be developing your lab.

[](/description_images/org/mp1_code.png)

Once you have a solution, you click on the submit tab. This will first save your work on the server and then submit it to be processed. In case the server is overloaded, your computation would be placed in a queue (this is a great motivation to not start the lab at the last minute).

The third tab (if shown) shows the previous attempts made at this problem

[](/description_images/org/attempts.png)

### Insertion Points

Most of instructions for the lab are found in the code. We use //@@ to demarcated where code needs to be inserted. In lab 1, for example, you see code such as:

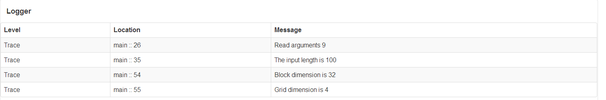
wbTime\_start(GPU, "Copying input memory to the GPU.");  
//@@ Copy memory to the GPU here  
wbTime\_stop(GPU, "Copying input memory to the GPU.");

This means that you need to insert a CUDA memory copy operation where the comment is located.

### Logging and Debugging

Logging is facilitated through the use of a logging API. The logging function wbLog takes a level which is either OFF, FATAL, ERROR, WARN, INFO, DEBUG, or TRACE and a message to be printed. To print the value of variable x, for example you type wbLog(TRACE, "The value of x = ", x). Note that variables are serialized to a string when placed in the wbLog function, so you can place reals, integers, and strings in the logging function.

The result of the logging messages is shown once you submit your code.

[](/description_images/org/logging.png)

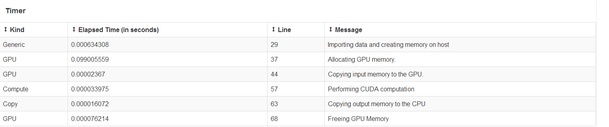
Debugging is done in the form of logging, so you cannot place break points, etc...

### Timing Code

Logging is facilitated through the use of a timing API. The timing functions wbTime\_start and wbTime\_stop. To time a section of code, you write wbTime\_start(tag, msg) and then wbTime\_stop(tag, msg), where the tag and msg must match. In the template code given for Lab 1, for example, we do

wbTime\_start(GPU, "Allocating GPU memory.");  
//@@ Allocate GPU memory here...  
wbTime\_stop(GPU, "Allocating GPU memory.");

the above would time the amount it takes to allocate GPU memory. The timing result is in shown in the attempts tab after you submit the program.

[](/description_images/org/timing.png)

The timing information includes the tag and message you specified when you timed your code, and also which line you are timing. Since it is important to understand the performance of your code, it is valuable to spend time to understand the output of the timer.

### Previous Attempts

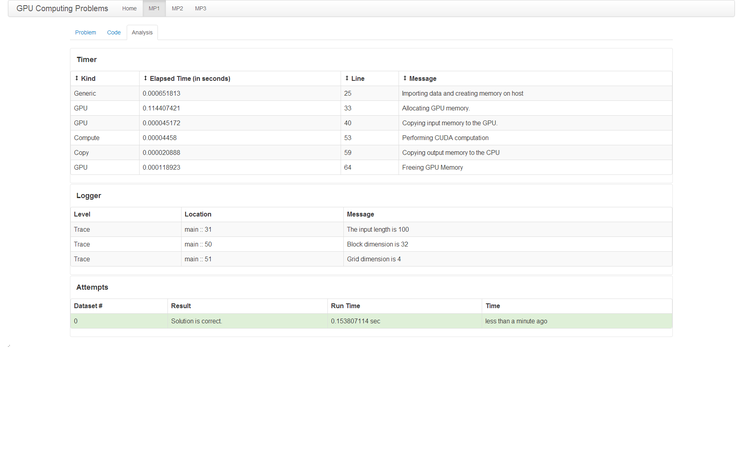
All previous attempts are saved and recorded on the server side, but you are only graded on the latest attempt.

### Walk Through

# Behaviors

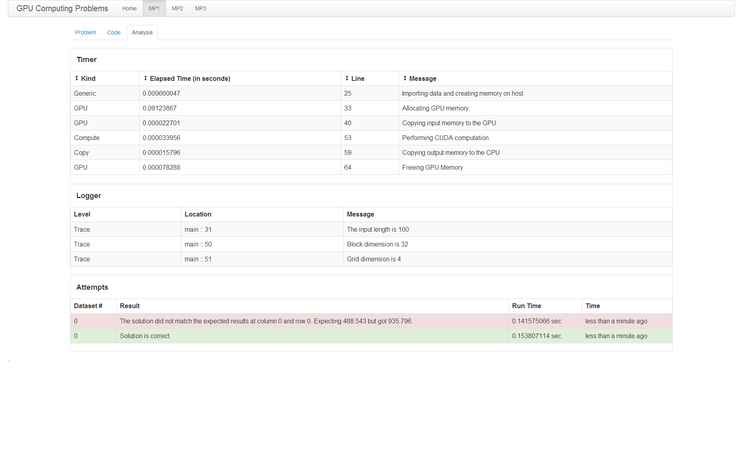
While developing the labs, you may encounter one or more of the following behaviors.

### No Error

[](/description_images/org/correct.png)

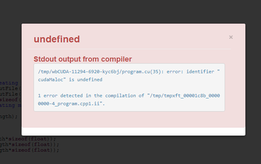
This means that no errors were found while running the program. Your program has been checked against the expected solution of the dataset you selected. Note that it might be the case that the program may run correctly on one dataset and not the other.

### Solution is Incorrect

[](/description_images/org/incorrect.png)

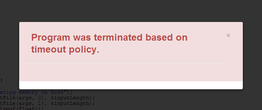
This means that while the program did compile and run without errors, the solution did not match the expected results. This problem could stem from either having an implementation error in your algorithm, or from having incorrect logic in the host code. Logging the state of your program at different points would help you debug the problem.

### Compilation Failed

[](/description_images/org/syntax.png)

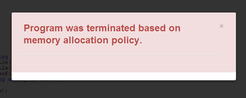
This error message occurs when the program submitted failed to compile. The output from the compiler is shown in the error window.

### Program Terminated due to Timeout

[](description_images/org/timeout.png)

The most likely cause of this error is that you have inadvertently placed an infinite loop either in your CPU or GPU code. Part of the reason for this behavior is that the system ensures fairness (i.e. you should not hog down the machine). To ensure fairness, the system is configured to terminate long running processes. When you see this error, you have hit that timeout limit.

### Memory Allocation Error

[](/description_images/org/memory.png)

To maintain system stability, users are not allowed to allocate too much memory.

### Sandboxing Error

[](/description_images/org/sandbox.png)

To maintain security, we run your program in a sandboxed mode. This means that you are restricted to using our API functions for certain tasks, rather than using system calls or C library functions. The sandbox error would sometimes tell you what call is being caught, although that might not always be the case.

## System Flow

## Suggestions

The following suggestions are recommended when getting started using the system

* Develop your application incrementally
* Do not modify the template code written -- only insert code where the //@@ demarcation is placed
* Make sure to test your results against all the datasets provided
* Do not wait until the last minute to attempt the lab.

## Grading

Grading is performed based on criteria that are specific for each lab. Aside from the program submitted compiling and running, we also look at how fast the program is in relation to the other submissions.

## System Requirements

A recent web browser is the only requirement for using and submitting labs in this course.

## Issues/Questions/Suggestions

In case of issues or questions, please post to the forums. Suggestions are welcome and can also be posted on the forums.