

(Due: May 1)

**Project Description**

The purpose of this project is to write a distributed application using the Remote Procedure Call (RPC) protocol the pthread library, and the CUDA toolkit. In this project you are expected to use the ONC RPC Protocol to distribute computation from a Linux workstation to a Linux workstation with the lowest load.

This project can be divided into two parts. The client program running on a workstation provides the interface to the users. The servers running on the pre-selected four Linux workstations provide the machine load reporting service and another two computational services. You are required to follow the eight steps of RPC distributed application implementation discussed in the class and use the protocol compiler rpcgen to implement the client/server programs.

The client program firstly reads the file machinefile which contains a list of four available Linux workstation names in our lab. Then it invokes a remote procedure getload() to collect the load average (over the last 1 minute) from each workstation in the list and selects the workstation with the lowest load average. Next, based on the command line option (-max or -upd), the client invokes the computation function findmax\_gpu() or update\_lst() on the selected workstation, gets the result back, and prints out the result.

Two examples are shown below. The first example, with the command line arguments -max N M S, finds the largest element of an array which has  $2^N$  (e.g.  $2^{20}$ ) elements. The elements of the array have been initialized using the exponential distribution with the mean value M (e.g. 5) with the initial seed S (e.g. 17). The computation is executed on the smallest load machine chopin with its GPU. In the second example, the client calls the remote function update\_lst() on the machine bach which converts each value F in the linked list by using the formula  $\sqrt{F * 10.0}$  and returns the updated linked list back to the client.

N M S

```
haydn % ldshr -max 20 5 17
arthur: 2.30 bach: 3.3 chopin: 0.50 degas: 1.27
(executed on chopin)
75.161147
```

```
haydn % ldshr -upd 64 1 49 99 88
arthur: 1.30 bach: 0.50 chopin: 2.27 degas: 6.99
(executed on bach)
80.0 10.0 70.0 99.5 93.8
```

The server provides three services (i.e. functions). To get the load average (over the last 1 minute) on Linux, you can call the system function getloadavg(). To find the largest element of an array, the function findmax\_gpu() firstly initializes the very large array using the values N, M, and S passed



from the client. It then launches the kernel function on GPU to find the largest element in the array and returns the value back to the client. You are required to modify the code in reduction\_kernel.cu discussed in the class to implement this. Similarly, the function update\_lst() gets a linked list of doubles from the client and then utilizes a higher-order function (i.e. a map function) along with a formula function to update the values in the list.

Note that you should run the server program on each of the four Linux workstations before you start the client.

## Turnin

Each group (at most two students) needs to submit this project using the following turnin command:

```
turnin -c cis620s -p proj4 makefile ldshr.x ldshr.c ldshr_svc_proc.c \  
      findmax.cu machinefile
```

You also need to hand in a hard-copy document which includes the description of your code, data structures, and experiences in debugging. The document should be typed. The cover page should contain your names, pictures, and your login-id you turned in.