I. Research Overview and Outcome

Problem Statement:

Nowadays, many GPU Computing systems are starting to have multiple GPU devices to solve problems [1]. In order to distribute the workload across multiple GPUs, developers must manage data exchange between main memory and these devices, guaranteeing consistency between the multiple copies of data and making the development for these architectures is more difficult for the developer. In order to use GPU resources effectively and avoid of resource wasting it is better to share resources. There are different mechanisms for sharing resources. It is possible to share one GPU among multiple users by installing a hypervisor on the GPU or just share multiple GPUs among multiple users. Since most GPU programs are computation intensive [2], we are going to implement a method which can share multiple resources among multiple users.

Proposed Solution:

We have designed an architecture for Collaboration among GPU clusters (RDGPUC). This software is responsible for receiving requests from a user interface and distributing requests among one or more GPU clusters. When GPU clusters finish processing one request, RDGPUC will notify the user of the result. Request in RDGPUC can be in variant forms, but, in general, one request consists of one or more kernel codes which need to run on GPUs.

Users can submit kernel codes individually or add a GPU/GPU code together. In our design, we tried to make a generic design for managing GPU clusters. Since there are many companies, universities and other institutes who are using GPU clusters and since each of them has their own regulations and their own software to manage their GPU cluster, in our design we make a standard interface for communication with RDGPUC. This application consists of the modules which are shown in Figure 1. Each module is responsible for one general task. But it is possible for individual users to add customized modules to the system. Each component is responsible for one or more tasks.

This software allows a user to add a request, observe the request list, observe results, and edit profile. The administrator of this system is able to manage resources, manage users, manage user groups, manage resources groups, manage schedules, analyze requests, fetch requests, allocate requests, and collect results.

The User Interface is responsible for accepting requests from users and put them into the Database. The Database stores users, resources, and requests as temporary information. The Analyzer analyzes each request to determine sub-requests, estimate execution time of each request, and put this information into the Database. The Resource Allocator enables communication between GPU clusters and RDGPUC. The GPU clusters are responsible for executing jobs.

Remarks and Future Work:

We have proposed an architecture which can be used to manage many GPU clusters. Each GPU cluster can have its own software for managing GPU resources and communicate with RDGPUC by using common XML-RPC interface. So, it is easier for institutes to collaborate in this project without sharing information of their resources. Administrators need to add resources, users, resource groups and user groups to the system. User groups and resource groups bring control over resources. Each resource group is associated with one schedule, which shows availability of resources during the time. Each user group can be assigned to one or many resource groups, which means that requests from this user group will go directly to particular resource group. This software consists of many modules. For individuals, it is easy to add a new module to extend the software for special requirements. Each module is responsible for one special task, but it is possible to extend modules or add new modules, in order to satisfy special requirements. Different kinds of user interfaces can be developed for special devices. Artificial intelligent algorithms can be used by the analyzer to estimate execute time more accurately. It is also possible to add more features on the management node of each GPU cluster.

II. International Experience

I learned a lot of things, made a lot of friends, and started interesting research in Brazil. During my trip to Brazil, I have edited one paper, finished another one, and also started working on other interesting topics of software engineering. I visited many places in Rio de Janeiro and enjoyed the food and coffee of Brazil. I also really enjoyed working at Fluminense Federal University; everything and everyone was so friendly. My advisor, Professor Cluá and his students are really nice and they invited me to many places. They helped me a lot during my research in Brazil.

III. Acknowledgement

The material presented in this poster is based upon the work supported by the National Science Foundation under Grant No. OISE-0730065. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or the Department of Education.

References: