Msc Project Progress Report: Developing an Application to Introduce Parallel Task Scheduling

s1843212

1. Goal of Project

The goal of this project is to develop an game like application to introduce parallel task scheduling to students and to help the students to learn. The content should cover a variety of concepts in this area. For example:

- What is task graph in parallel computing and how it's related to real world problems.
- What are clusters and varieties of clusters when analyzing task scheduling problems.
- General problems in this area like communication cost and heterogeneous clusters.
- General techniques in this area for performance improvements like keeping tasks on same processor to minimize communication cost.
- Common algorithms for task graph scheduling and their applicable type of cluster.

2. Methods Used

This project focuses on development of the application. The current technical choice is LWJGL on Java, which includes Java binding of OpenGL, OpenAL and several other native libraries, same as described in project proposal. It is chosen because it provides cross platform support for hardware rendering, which enables better performance and produces more flexibility in development like manual matrix control and immediate rendering. The drawback of the choice is that work is required to implement an abstract layer over low level APIs to build a complex application.

This application uses many levels (or missions) to introduce different aspects of the concept. Some may ask the student to simulate the behavior of standard algorithms, while some may ask the student to provide the schedule that finishes within a period of time, with the help of specific techniques. It encourages students to do experiments and shows the result, to provide an intuitive learning experience and to reinforces the understanding.

3. Current State

Figure 1 shows a screenshot of the application. Inside the interface, the user can drag tasks to assign them into queues of each processor and the engine will execute them automatically and display the history of each processor. Although the interface is still missing some details, it seems many parts are working correctly. Up to now, this project contains over 5k lines of code and the components listed below has been finished:

- A flexible GUI framework using immediate rendering based on OpenGL.
- Many widgets with advanced features like drag and drop, animation and cursor highlight.
- A simple sound system based on OpenAL.
- An engine to simulate execution of task graph on clusters with different configurations.
- Data driven models for games, graphs and clusters to program levels.
- A algorithm systems to let the user develop and run customized algorithms.

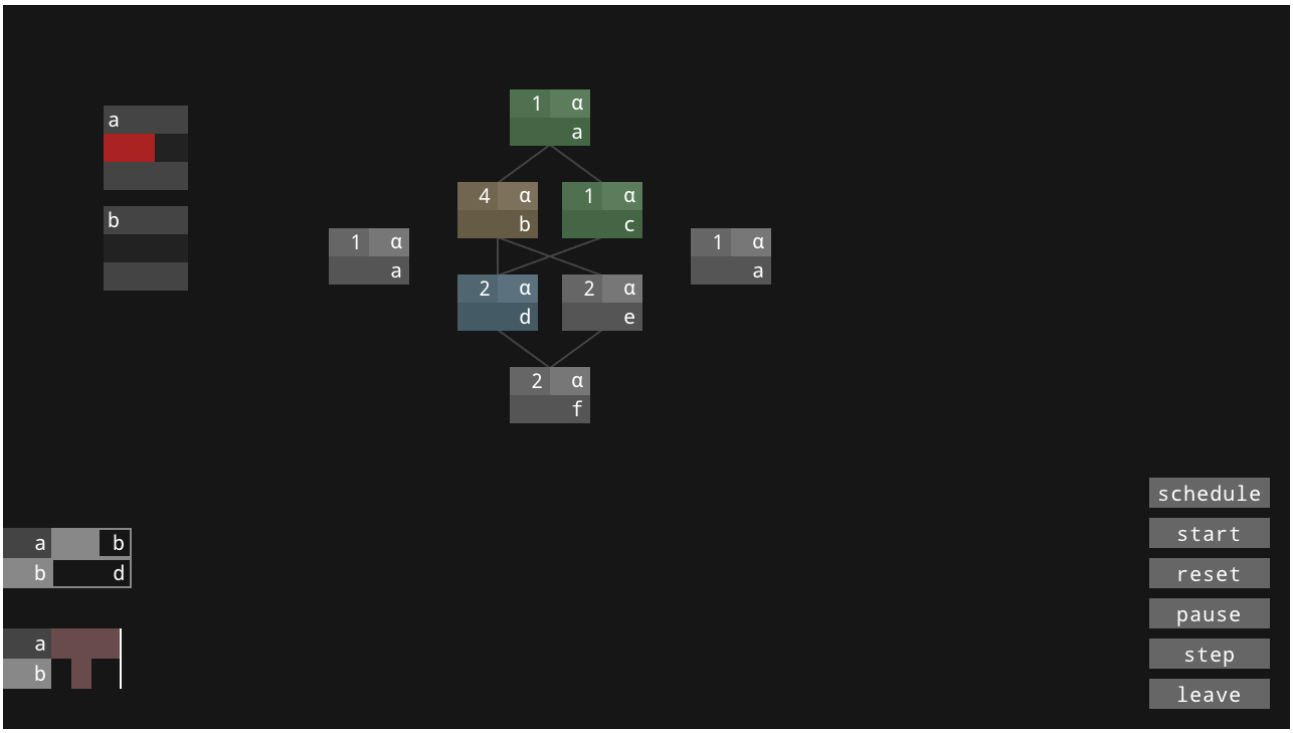

Figure 1. Screenshot of the application. The upper left shows the state of processors. The upper right shows three task graphs. The bottom shows task queues and execution timeline. Every task has three elements: duration, type and name. In a task graph, a task is grey when unassigned, blue when assigned, yellow when processing and green when finished.

4. Future Plan

In the remaining time, two things will be done. The first is to implement the tutorial system to introduce the mechanisms of the system and standard algorithms. It requires some tweaks to the GUI framework and much text writing. So it could take some time to finish. Another thing is user testing. One week will be reserved for it.

If time allows, extra features will be added to the application, like multiplayer mode and testing mode. Decisions will be made according to progress. Also, extra details will be added to the interface to make it more appealing.