School of Informatics

Informatics Project Proposal
A Tutor for Parallel Task Scheduling

s1843212
April 2019

Abstract

Date: Friday 5th April, 2019

Tutor: Valentina Andries
Supervisor: Murray Cole

Contents

1 Motivation

With speed of hardware improvements reducing, the computing power of single processor core is getting limited. In this case, the industry have to rely on parallel computing for massive computation, of which the core part is to divide the task for better parallelism. It is common to divide a big job into several tasks with dependency. Typically, the tasks and dependencies will form a directed acyclic graph (DAG), which is called task graph. Afterwards, algorithms are applied to assign each task in the task graph to processors in the given cluster. In this case, students are required to have descent understanding of the algorithms to better utilize computing resources in the cluster.

1.1 Problem Statement

As is described in previous section, algorithms are used for resource allocation of task graph. This process is called task graph scheduling. However, it is not easy to learn, mainly due to the variation of system configurations and algorithms, which will be introduced in detail in section 2.1. For different combination of cluster model and scheduling algorithms, it leads to different steps in calculation, and for students, makes the algorithms more confusing to understand.

Therefore, this project aims to develop an application to help the students learn the algorithms in a more interesting and interactive way.

1.2 Research Hypothesis and Objectives

As stated before, this project aims to help students learn scheduling algorithms better. It is hypothesized that using this application can provide 1) solid learning outcome and 2) pleasant learning experience. To achieve better results, the design of application should provide strong feedback and satisfaction. Also, it should provide intuitive control logic to make it easier to use.

For variances described in section 1.1, the application will include all of them if time allows. Otherwise, it should at least include some representative variances to provide a complete view of the problem.

In this project, the learning outcome will not be compared to that of traditional learning methods due to the limit of time and participants. In this case, it should typically provide a qualitative feedback on user experience and learning outcomes.

1.3 Timeliness and Novelty

The project can be considered timely and novel for two reasons. Firstly, the content introduced by the application is timely. Although some of the algorithms are invented tens of years before, they are still meaningful to learn and widely used in industrial practice.

Secondly, although it is not common, it is a trend to use such applications or games to assist teaching in university education, since the volume of knowledge sometimes makes it hard for students to learn in limited time. This project can propose a possible application of this teaching method.

1.4 Significance

Since this project is proposed for practical usage, the main significance is to help the students learn better about the algorithms and to possibly save time for tutors. It should also perform as a handy development and testing environment for graph task scheduling algorithms. Despiting the details of algorithms, this application actually introduces the concept of scheduling, which is not only used in parallel programing tasks. Therefore, it can be useful to introduce similar concepts like database query planning or project management, with slight tweaks to the model. In addition, it can be considered an example implementation of game-like applications for teaching purpose on selected platforms and libraries. The design logic and implementation details can be referred by similar projects in the future.

1.5 Feasibility

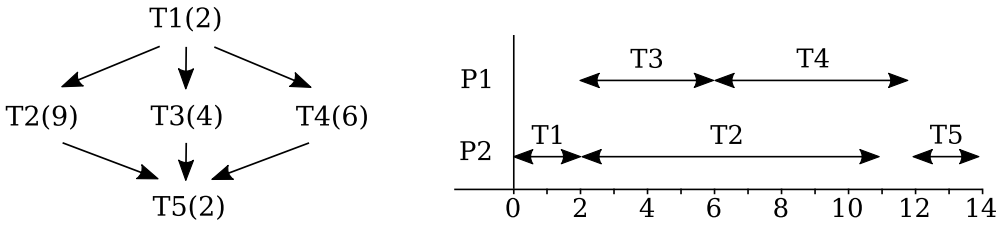
To implement all the variances of model and algorithm with step-by-step introduction, it is estimated to take about 5-10k lines of pure code. Given a 2 month period excluding document writing and testing, it results in over 100 lines of code added per day, which seems feasible, but slightly challenging, considering there might be some reconstitution for projects in this size. For a basic working application, it is estimated to take less than 2k lines, reducing the production rate to less than 50 lines per day, which seems relatively easy.

1.6 Beneficiaries

As described in section 1.4, the direct beneficiaries are students learning and tutors teaching the task graph scheduling algorithms, but it could also be useful to teach similar concepts. In addition, the implementation details can be helpful as reference when anyone is building similar applications.

2 Background

2.1 Task Graph Scheduling



(a) task graph

(b) schedule

Figure 1: A example of task graph scheduling

When processing a big task, a common way is to divide the original task into small tasks. However, in many of cases, some sub-tasks have to be executed in certain order, or called dependency. Such relationship can be represented by a task graph, in which figure 1a shows an example, where each task is described in the format of "identifier (duration)" and dependencies are given as arrows pointing to dependant. Figure 1b shows an example schedule using two processors P1 and P2, where lines with arrows shows that one processor is allocated to one task in given period. It can be observed schedule reduces total execution time from 23 units to 14 units. In most of schedules, processors can not be made running during the entire period due to the dependencies, so that scheduling algorithms are used to better utilize the processors.

The complexity of this topic is the variation of models. The model given in figure 1 is based on the simplest assumption: each task takes same time on different processors and data can be distributed to other processors immediately. However, to better describe practical tasks, models are given more details. Here are several well-developed variances of models:

- Homogeneous vs. heterogeneous in performance of each processor: For homogeneous clusters, the processors are considered to be the same, making tasks take generally the same time to finish in each processor. For heterogeneous clusters, a simple model is to have different multipliers for each processor, making some of them faster to finish every task. In addition, complex models can include specialized processors in the cluster, making them faster only for certain types of tasks.
- Ideal (immediate) vs. practical in communication performance: For ideal situation, it is considered to happen immediately to transfer data from one processor to another, while it can take some time in practical models. With more details, the time consumption of communication can also dependent on the size of messages. In this case the algorithms have to consider size of messages between each tasks due to the difference in time delay.

- Multiple vs. single communication channel in each processor: For models assuming multiple communication channels, each processor can communicate with many processors simultaneously, while it is considered to be able to send or receive data from only one processor in models assuming single communication channel.

There are also variances in scheduling algorithms. For the most popular one, list scheduling algorithm, different functions can be used to determine the priority of tasks, such as longest path, longest time and critical path, which can have significant effect to allocation results. In this project, the applications of different algorithms on different models will be introduced.

2.2 Education Software

There are many projects trying to develop applications for teaching assistance, for topics including algorithms, data structures or more complex concepts like programming. Shabanah et al. developed several applications to introduce basic algorithms like binary search and data structures like linked list and binary search tree. The robot game Karel is famous for its usage in MIT to teach programming, especially for ideas related to sequence control, making the learning process more enjoyable. In addition, Harteveld et al. reviewed 36 projects for similar purposes.

There are more games designed based on engineering tasks. The game company Zachtronics is famous for such games. For example, "TIS-300" is designed based on the model with a mixture of assembly programming and manycore parallelism and "Infinityfactory" is designed based on production line management and geometric relationships. In addition, the game "Screeps" uses Javascript, a widely used programming language, as the main method to play the game, by programming AI of entities in the in-game world, for which the game is also suggested as a tool to learn programing.

3 Programme and Methodology

3.1 Appearance

There are many choices to be made in design of this project. An important part is the design of the interface. The main idea is to provide intuitive interaction and representation of the task. For example, the user should be able to use drag and drop to allocate tasks to processors. For tasks, there can be many representations, which is given in figure 2 and table 1, according to the task graph given in figure 1a. In design of games, text are usually avoided because 1) it takes more effort for human to read compared to shapes and 2) text is more complex in layout of interface, especially when considering localization. Figure 2 is made according to this design logic, using rectangles for tasks with solid squares for duration. Table 1 shows the same information using a more formatted structure. It is easy to be read by machine and stored in database, but for human, it is more difficult to recognize the dependencies. There can be more formats to represent the same structure, but in a game-like application, formats similar to figure 2 are more likely to be used.

In addition to details, the quality of layout also contributes to the overall usability. Figure 3 shows an typical layout of interface, showing task graph on the left side, status of processors on the right side and allocation of tasks in the bottom. There can be many choices in layout, but it should provide significant amount of information to the user.

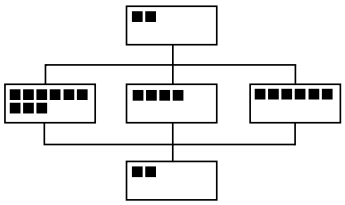


Figure 2: A representation of task graph using shapes

Identifier	Duration	Dependencies
task 1	2 units	
task 2	9 units	task 1
task 3	4 units	task 1
task 4	6 units	task 1
task 5	2 units	task 2, 3, 4

Table 1: A representation of task graph using one table

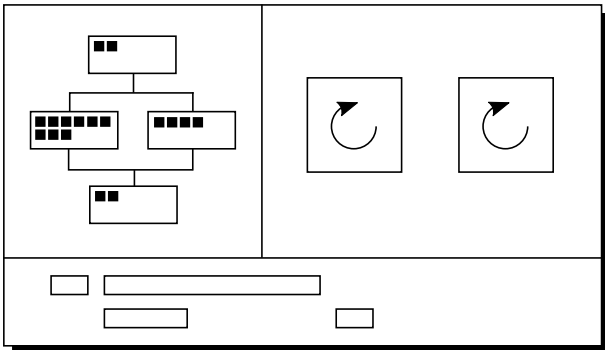


Figure 3: A example layout of interface

3.2 Implementation

To provide simplicity in usage and accessability in different environments, the application is decided to be a cross-platform native application. There are many options to build a cross-platform application like 1) write the program in C++ and cross compile to different platforms, 2) write the program in Java (or other JVM languages) and use dedicated virtual machine on different platforms and 3) write the program in Python or other scripting language and use different interpreter on different platforms. The final choice is to use Java because it provides a balance in performance and simplicity of development. Although installation of Java virtual machine is sometimes annoying, considering the application is intended to be used on DICE [1] machines and by computer science students, it should be relatively easy to setup.

As is described in section 3.1, this program might require usage of textures and animations, which is not emphasized in traditional GUI [2] frameworks. Although there are several well known GUI frameworks in Java like AWT, Swing and JavaFx, the final choice is to build a dedicate GUI framework to provide more flexibility and consistency on different platforms, similar to many games. There are several low-level libraries or game engines to build an application with customized rendering, including libGDX, jMonkeyEngine and LWJGL. The final choice is to use LWJGL because 1) it has great community support; 2) it provides full exposure of native interfaces of OpenGL – a cross-platform graphical API [3] and 3) it contains handy support for other parts in application development like audio playback.

[1] DICE: the computing platform of university base on Linux with Java 8 installed by default
[2] GUI: graphical user interface
[3] API: application programming interface

4 Evaluation

This project will mainly be evaluated in two aspects. Quality of design and quality of implementation. Since the application is developed for education purpose, the design will be evaluated according to learning results of testers. Several students will be invited to use the application to learn task graph scheduling algorithms, then tested by some questions. Ideally, the users are expected to know variances described in section 2.1 and tell the detailed execution of algorithms in different models.

For quality of implementation, it will be evaluated according to compatibility in different environments like variant operating systems, hardware configurations and screen resolution. It should at least run successfully out-of-box on DICE machines and is highly expected to run correctly on Linux and Windows. In addition, criterions like quality of documentation and program structure will be discussed.

5 Expected Outcomes

6 Research Plan, Milestones and Deliverables

References