School of Informatics

Informatics Project Proposal
Developing an Application to Introduce Parallel Task Scheduling

s1843212
April 2019

Abstract

This project aims to develop an game-like application to introduce the concept and details parallel task scheduling to students. The report starts with backgrounds in parallel programming, providing the general scope, then describes the complexity of task graph scheduling problems, including all the variants to be included in the application. Afterwards, the design of the application is introduced, including ways of interaction, appearance of interface and selection of libraries. Finally, it briefly introduces the evaluation method, expected outcomes and planning of the project.

Date: Thursday 11th April, 2019

Tutor: Valentina Andries
Supervisor: Murray Cole

Contents

1 Motivation

Computing performance is highly demanded in IT industry and academic researches. For companies, they can easily acquire terabytes of user data, which requires massive computing resources to process timely. For researches in areas like physics, researchers rely on simulation to evaluate the behavior of models, which also requires more computing resource to have higher resolution. However, the speed of improvements in computing power of single processor core is slowing down. In this case, the developers have to rely on parallel computing to utilize more computing resources and to reduce the time cost to finish one job.

In parallel programming, a common practice is 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[1]. To have the job executed in a cluster, a task scheduler will use scheduling algorithms to assign each task in the task graph to processors in the given cluster. Different scheduling algorithms will lead to different assignments of tasks, resulting in different performance characteristics of the system, like delay and throughput. Therefore, it is beneficial for students to learn such algorithms 1) to understand the design and behavior of such scheduling frameworks; 2) to implement tasks that are more friendly for parallel execution and 3) to tweak or implement scheduling platforms according to engineering requirements.

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 to understand the issues in task graph scheduling and to learn the algorithms in a more interesting and interactive way. Using this application, the students should be able 1) to try different allocation of tasks for various combination of models and task graphs, then see the execution results and 2) to simulate and verify the behavior of standard algorithms.

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 solid learning outcomes and 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 in system models and scheduling algorithms, 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

For the content to be introduced, although some of the scheduling algorithms are invented tens of years before, they are still meaningful to learn and widely used in industrial practice, because parallel computing is gaining popularity to solve computation intensive problems. Many companies now rely on distributed computing frameworks like Hadoop and Spark for data processing, where YARN, a resource management system, is used to allocate computing tasks to processors. Task scheduling algorithms are important theory backgrounds in design of such systems.

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, while using such applications is promising to produce better learning outcomes. There seems to be no such projects related to task scheduling algorithms, so this project can propose a possible application of this teaching method in such area.

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. As well as 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

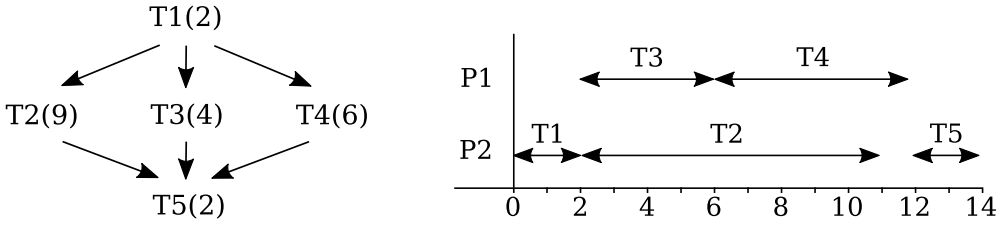
To implement all the variances of model and algorithm with step-by-step introduction, it is estimated to take about 5-10k lines of pure code according to previous experience. Given a 2 month period excluding document writing and testing, it results in over 100 lines of code added per day, which seems 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

To enable a big task for paralleled execution, a common way is to divide the original task into small tasks, then distribute the sub-tasks to several processors. If the task is infinitely separable, using this method reduces the total time by a factor of n, where n is the amount of processors. For real-time systems, it also means reduction in delay. 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.

Figure 1a shows an example of task graph, where each task is described in the format of "identifier (duration)" and dependencies are given as arrows pointing to dependant. An arrow pointing from task A to task B means task A have to be finished before the start of task B.

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 that the schedule reduces total execution time from 23 units sequentially 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 equally powerful. For heterogeneous clusters [2], a simple model is to have different multipliers for each processor, making some of them faster to finish one task compared to other processors. In addition, complex models can include specialized processors in the cluster, usually related to specialized instruction sets or usage of co-processors, 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 depend 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 well discussed algorithm, 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. This project will try to cover all the variances described above.

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[3]. The robot game Karel is used by many universities to teach programming, especially for ideas related to sequence control, making the learning process more enjoyable[4]. In addition, Harteveld et al. reviewed 36 projects for similar purposes [5].

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 Learning experience

This applications is expected to provide a complete description of different scenarios of task graph scheduling. For example, static systems are required to be optimized for only one task graph, but for dynamic systems, the have to adapt to changes in task graphs or to discover new tasks in real-time, which could lead to difference in preferred allocation. Therefore, it is designed to have different modes for different purposes:

1. **Static:** The user is required to discover a schedule with requirements like total time or utilization rate, base on one fixed task graph. There is no time limit.
2. **Dynamic:** The user is required to schedule and finish all the tasks from a dynamically expanding task graph. The processors will run in real time and the task graph will be gradually revealed. So the user have to determine the allocation quickly.
3. **Leaning:** The user is required to provide the schedules generated by existing algorithms. The application will provide instructions to learn the algorithms and check if the provided schedule matches the expected one.
4. **Multiplayer:** Similar to dynamic mode, but multiple players are competing to finish the tasks in shortest time.
5. **Testing:** The user can develop a scheduling algorithm (possibly in Java), then the application will run the algorithm and show the result.

Some modes like static mode and dynamic mode seems simple to implement, while modes like testing mode seems challenging, especially when letting the user to write programs inside the application. Therefore, testing mode will be implemented only if time allows, while other modes are expected to be finished.

3.2 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. One possible method to represent task

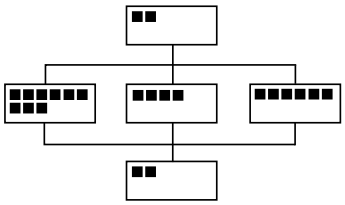


Figure 2: A representation of task graph using shapes

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

allocation is to have several rows, each for one processor, to show the timeline of processors, so that the user can drag and drop tasks into the timeline of any processor with results shown in real-time. There can be many choices in layout, but it should provide significant amount of information to the user.

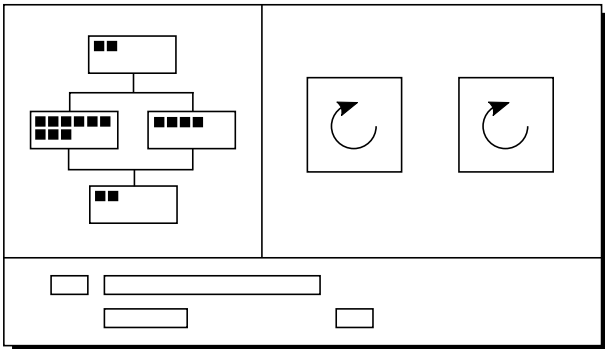


Figure 3: A example layout of interface

3.3 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 3.2, 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

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

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.

4 Evaluation

This project will mainly be evaluated in two aspects: quality of design, which is more important, and quality of implementation. Since the application is developed for education purpose, the design will be evaluated according to learning results of testers. Although there are some arguments, the application will be tested base on five user assumption [6], due to the limit of time. According to the assumption, 5 students will be invited to use the application to learn task graph scheduling algorithms. The testers will be suggested to continuously describe their thoughts about the experience, so that the entire process can be recorded. Afterwards, the testers will be asked some questions to evaluate their understanding of the algorithms. Ideally, the users are expected to know variances described in section 2.1 and tell the detailed execution of algorithms in different models.

The process is expected to take less than 4 hours, because it will become hard to arrange if it takes longer. The duration seems not sufficient for a student to understand all the details of the algorithms. In this case, if the tester is willing to, he/she can copy the application for further study and provide more feedback on user experience.

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

Since this project is mainly development based, the main outcome is the compiled executable and source code of the application. For testing mode described in section 3.1, there will be brief documentation if it requires extra steps to setup. In addition, test results described in section 4 will be given. As described in previous sections, this application is expected to improve efficiency of learning. It is also important to provide interesting and satisfying learning experience because it makes the students willing to study.

6 Research Plan, Milestones and Deliverables

[3] API: application programming interface

Milestone	Week	Description
M_1	2	Building blocks and utilities completed
M_2	5	Static, dynamic and learning mode completed
M_3	8	Multiplayer and testing mode completed
M_4	9	Testing completed
M_5	10	Submission of dissertation

Table 2: Milestones defined in this project.

References

[1] Yves Robert, Sameer Shende, Allen D. Malony, Alan Morris, Wyatt Spear, Scott Biersdorff, Burton Smith, Dali Wang, Daniel Ricciuto, Wilfred Post, Michael W. Berry, François Irigoin, Katherine Yelick, Susan L. Graham, Paul Hilfinger, Dan Bonachea, Jimmy Su, Amir Kamil, Kaushik Datta, Phillip Colella, Tong Wen, Jack Dongarra, Piotr Luszczek, Abhinav Bhatele, Stefan M. Freudenberger, Volker Diekert, Anca Muscholl, Maurice Herlihy, and J. Eliot B. Moss. Task Graph Scheduling. In *Encyclopedia of Parallel Computing*, pages 2013–2025. Springer US, Boston, MA, 2011.

[2] H. Topcuoglu, S. Hariri, and Min-You Wu. Task scheduling algorithms for heterogeneous processors. In *Proceedings. Eighth Heterogeneous Computing Workshop (HCW'99)*, pages 3–14. IEEE Comput. Soc.

[3] Sahar S. Shabanah, Jim X. Chen, Harry Wechsler, Daniel Carr, and Edward Wegman. Designing Computer Games to Teach Algorithms. In *2010 Seventh International Conference on Information Technology: New Generations*, pages 1119–1126. IEEE, 2010.

[4] Byron Weber Becker, Byron Weber, Becker, and Byron Weber. Teaching CS1 with karel the robot in Java. In *Proceedings of the thirty-second SIGCSE technical symposium on Computer Science Education - SIGCSE '01*, volume 33, pages 50–54, New York, New York, USA, 2001. ACM Press.

[5] C Harteveld, G Smith, G Carmichael, and E Gee. A design-focused analysis of games teaching computer science. 2014.

[6] Robert A. Virzi. Refining the Test Phase of Usability Evaluation: How Many Subjects Is Enough? *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 34(4):457–468, aug 1992.