Process Scheduling using Machine Learning

Daniel Houwen
Sai Kishore
COEN283 – Operating Systems
SPRING 2014

1 TABLE OF CONTENTS

1.1 TABLE OF FIGURES

2 INTRODUCTION

2.1 OBJECTIVE

The aim of this project is to develop a task scheduling method using supervised machine learning techniques that is more efficient than common existing methods. Since there many different techniques, with unique advantages and disadvantages, finding a suitable technique will be an important part to develop efficient scheduling.

2.2 PROBLEM BACKGROUND

In many areas of computing, developing efficient task scheduling algorithms can significantly increase the performance of the system. This is particularly true with parallel and distributed computing, which is becoming increasingly widespread. There has been much work in developing heuristic methods for task scheduling and using machine learning techniques for optimization problems, but there is still a lack of research combining the two areas.

2.3 APPLICATION OF THE PROBLEM TO OPERATING SYSTEMS

Process scheduling is one of the core concepts in operating system theory, because it is one of the most important tasks for the efficient operation of modern operating systems. This project is investigating an innovative method for process scheduling, which can lead to the improved performance of a system.

2.4 DISADVANTAGES OF OTHER APPROACHES

There have been many developments in process scheduling leading to better performance, however as systems are becoming more sophisticated, the approaches needed to optimize these systems must also become more sophisticated. There are several new approaches to task scheduling and mapping being applied to same systems as this approach. These include nondeterministic approaches, such as genetic algorithms or unsupervised learning algorithms, and deterministic approaches, such as directed acyclic graph scheduling algorithms. Nondeterministic approaches may provide near-optimal scheduling, but they increase the execution time of the system which reduces their efficiency. Deterministic approaches have the advantage of efficient computation, but their actual performance is typically worse than non-deterministic approaches.

2.5 ADVANTAGES OF SUPERVISED LEARNING

Developing a supervised learning algorithm is a deterministic approach, which means it has the advantage of fast execution. To overcome the performance issues associated with deterministic approaches, this approach is concerned with developing an accurate classification of task processing time, so that the best scheduling decisions may be made for each task. Where other deterministic approaches have little to no information about how tasks will execute, a supervised learning approach can attempt to learn how tasks will execute before the system even starts.

2.6 PROBLEM STATEMENT

Given the increasing complexity of processor architectures, sophisticated approaches are needed for efficient scheduling and mapping on these systems. Incorporating machine learning techniques has provided improved results in many related fields, so it is worthwhile to apply these techniques to scheduling problems.

2.7 PROJECT SCOPE

The first portion of this project will be determining which type of machine learning algorithm to use, which is tied to the specific scheduling aspect to be optimized, so it will be investigated in parallel. The implementation will cover feature extraction, developing training data, applying the training data, and implementing the algorithm in a scheduling program, using different metrics of testing against a common existing scheduling method.

3 THEORETICAL BASIS AND RELATED LITERATURE

3.1 PROBLEM DEFINITION

The reason efficient scheduling algorithms are needed is because of the requirement of most operating systems to perform multitasking. There are many requirements for a scheduling algorithm, including but not limited to: throughput, turnaround time, fairness, waiting, and optimal CPU utilization. As computing systems increase in complexity and become more diverse, more robust algorithms are needed to handle the new requirements of these systems.

3.2 THEORETICAL BACKGROUND

Process scheduling is not a new problem in operating systems. Many efficient algorithms have existed for quite some time, and new algorithms are continually being developed to address the scheduling issues involved with emerging computing systems.

The basis of machine learning is creating representations of data, or features of a system, and functions that evaluate the features to determine the output of the system. Supervised learning is an implementation of machine learning where the function, or classification, is created using known outputs for given test inputs. Support vector machines are models of supervised learning that are defined by a hyperplane separating categories of data, with the maximum margin between categories. A simple example of classification using an SVM in 2D space can be seen in Figure 1.

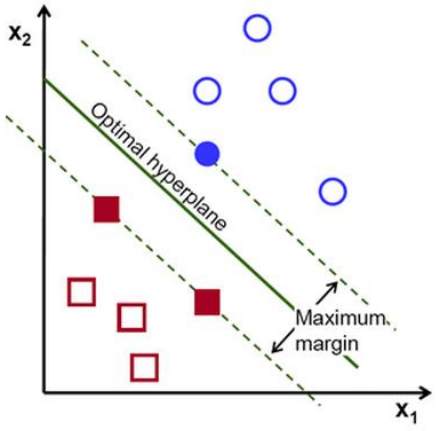

Figure 1 – Support Vector Machine Classification in 2D. [5]

3.3 RELATED RESEARCH

"Intelligent Task Mapping using Machine Learning," by Tetzlaff and Glesner, discussed a method to use machine learning to determine runtime behavior for processes in heterogeneous multiprocessor systems by analyzing static code features [1]. "Applying Machine Learning Techniques to Improve Linux Process Scheduling," by Negi and Kumar used ML techniques to learn CPU time-slice utilization, and minimize task turnaround time in Linux [2]. "Task Scheduling for Heterogeneous Computing based on Learning Classifier System," by Yang, Xu, and Jia, used the XCS algorithm to determine the optimal task assignment and execution sequence for heterogeneous systems [3].

3.3.1 Tetzlaff and Glesner - Advantages and Disadvantages

Although they introduced methodology for a complete approach to ML task scheduling, Tetzlaff and Glesner implemented only supervised learning of loop bounds. Using an unspecified algorithm, they were able to correctly classify the boundaries of loops within tasks within a mean absolute error of 0.94 for 115 static code features. Given their planned implementation for complete scheduling, these results are promising for developing efficient task mapping to reduce interprocessor communication. However, the results are limited, and there is no determination whether or not this approach actually does result in reduced turnaround time or throughput.

3.3.2 Negi and Kumar – Advantages and Disadvantages

Negi and Kumar used Machine Learning classifiers "Trees" and "Lazy" to learn about the CPU time slice utilization behavior of known programs in a linux system. They modified linux kernel to extract various process attributes that are inputs to learning programs. With machine learning they were able to predict CPU burst times, which minimizes the process turn around time. In paper they evaluated which process parameters help in better decision making and compared two learning algorithms. Major advantage of their approach is that given a known program, our learning algorithm knows its behavior. So we can predict and tune scheduler time slicing dynamically to improve efficiency by reducing process context switches involved at execution time. However method

discussed limits us to known programs only. In a real environment we need not always have prior idea on executing programs.

3.3.3 Yang, Xu, and Jia – Advantages and Disadvantages

Yang, et al., used the XCS algorithm, which combines reinforcement learning with genetic algorithms, to develop efficient task mapping to heterogeneous systems, similar to Tetzlaff and Glesner. In this paper, they were able to develop a complete implementation, finding the optimal task assignment for each processor as well as the task execution sequence. They compared their implementation to the well-known HEFT (Heterogeneous Earliest Finish Time) algorithm, and found that their approach provided a speedup in all test cases. The limitations of this paper are that they only provided a comparison to one other scheduling algorithm, and that their approach used on-line methods, which requires additional computation at runtime.

3.4 OUR SOLUTION

In this project, support vector machines will be implemented using the SVMRank program in the Minix 3 operating system. SVMRank is an SVM instance specifically designed for efficient training of SVMs to rank data [6]. Minix 3 is a small and simple Unix-like system, well-suited for experimentation and modification [7]. An algorithm will be generated by testing many different example programs set with different priority levels to determine which order of tasks provides the most efficient scheduling. By applying the SVM algorithm to rank the tasks, the default scheduler in Minix, a multi-priority round robin scheduler, will be modified to choose the next process based on the results of the algorithm. The results of this implementation will be compared to the results provided by the default scheduler.

3.4.1 Differences from Existing Solutions

The main difference from existing solutions is the implementation of an SVM to create efficient scheduling. While SVMs have been applied in some scheduling applications [8] [9], indicating the possibility for use in process scheduling, there is little information on their implementation in this area.

3.4.2 Advantages over Existing Solutions

The main advantage over deterministic approaches is that this implementation should have little effect on runtime speed, as the training is beforehand. In this case, a ranking SVM is used in this solution which should provide higher accuracy over other non-deterministic ML techniques, because it is specifically tailored for efficient ranking. This combination of accuracy and efficiency should provide advantages over existing solutions.

4 HYPOTHESIS

Using an algorithm generated by the RankSVM program, with training data provided by custom tests, more efficient process scheduling will be achieved than the existing round robin scheduler in Minix 3.

5 METHODOLOGY

Minix3 OS is chosen for this project implementation. Minix3 is a free and popular open source OS for academic research. Its kernel is lightweight and has support to run on virtual hardware. Minix3 is based on a tiny microkernel running in kernel mode and the rest of the OS runs as a collection of isolated, protected processes in user mode. It has a scheduler implemented in user mode allowing users to have multiple scheduling algorithms.

5.1 DATA COLLECTION

This project will modify the scheduler and clock routines to print process information at regular intervals to be used as features and corresponding output for the SVM algorithm. User space scheduling algorithm will be modified to provide information about process execution, time quanta and current queues.

5.2 PROPOSED SOLUTION

Specifically, the support vector machine (SVM) model will be used to help the scheduler decide which process to pick from a given queue. The SVM model takes a process queue, and associated features, as input and recommends next best process to schedule based on the ranking result.

5.2.1 Algorithm Design

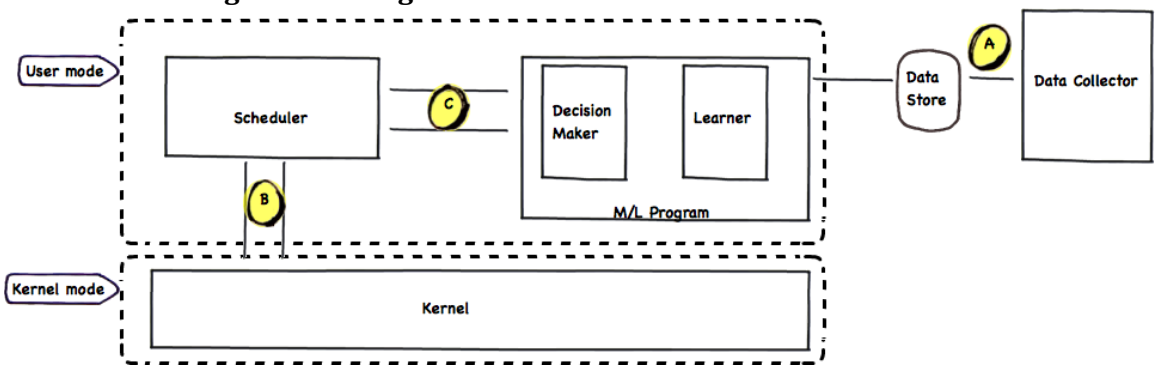

Figure 2 - Implementation of an ML program in Minix 3.

A. Data collector collects data about process information and stores in data store. This is an independent program that is triggered manually at initial stage to feed input data. When there is sufficient data this program runs as system idle process to keep collecting more data about completed/finished processes when system is idle.
B. Kernel: Scheduler communication: System calls to query next scheduled process.
C. Scheduler: Passes a process queue to decision maker. Decision maker picks best process from process queue and replies.

The steps that the overall system will perform are as follows:

1. Execute computationally heavy programs and collect initial sample data.

2. Develop and train the SVM algorithm.
3. Implement a method in scheduler to call the SVM algorithm using the features for the process that is to be scheduled next.
4. SVM algorithm will rank best process and return the ID for that process.
5. Selected process is scheduled.

5.2.2 Programming Environment

C/C++ with Minix3 implementation of ACK ANSI-C compiler.

5.2.3 Other Tools Used

SVMRank is used to train and implement the aforementioned SVM algorithm, a virtual box is used for development and Git is used for collaboration.

5.3 Output Generation

Run computationally heavy programs (P1…Pn) on two environments.

Environment 1: Unassisted scheduler/Default MINIX3

Environment 2: Modified scheduler with SVM

Both setups have traces to print required information as described above in section 5.1. This data is the output to verify the hypothesis.

5.4 Testing Methodology

Environment 1: Unmodified MINIX3 with default scheduler and traces enabled.

Environment 2: Modified MINIX3 with SVM assisted scheduler and traces enabled.

Computationally heavy programs are executed on above environments under identical conditions. Data is tabulated and compared against each run. This comparative analysis will test our hypothesis.

6 Implementation

6.1 Code Design

7 DATA ANALYSIS

7.1 OUTPUT

7.1.1 Output Analysis

7.1.2 Hypothesis Comparison

7.1.3 Abnormal Cases

7.2 STATISTIC REGRESSION

7.3 DISCUSSION

8 CONCLUSION

8.1 SUMMARY

8.2 FUTURE STUDIES

9 BIBLIOGRAPHY

[1] Tetzlaff, D.; Glesner, S., "Intelligent Task Mapping Using Machine Learning," *Computational Intelligence and Software Engineering (CiSE), 2010 International Conference on* , vol., no., pp.1,4, 10-12 Dec. 2010

[2] Negi, A.; Kishore, K.P., "Applying Machine Learning Techniques to Improve Linux Process Scheduling," *TENCON 2005 2005 IEEE Region 10* , vol., no., pp.1,6, 21-24 Nov. 2005

[3] Jiadong Yang; Hua Xu; Peifa Jia, "Task Scheduling for Heterogeneous Computing Based on Learning Classifier System," *Artificial Intelligence and Computational Intelligence, 2009. AICI '09. International Conference on* , vol.3, no., pp.370,374, 7-8 Nov. 2009

[4] Andrew S. Tanenbaum. 2007. *Modern Operating Systems* (3rd ed.). Prentice Hall Press, Upper Saddle River, NJ, USA.

[5] "Introduction to Support Vector Machines," OpenCV 2.4.9.0 Documentation, [online] 2014, http://docs.opencv.org/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html

[6] T. Joachims, "Training Linear SVMs in Linear Time," *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*, 2006

[7] "More about MINIX 3," MINIX 3, [online] 2014, http://www.minix3.org/other/read-more.html

[8] Yi-Hung Liu; Han-Pang Huang; Yu-Sheng Lin, "Dynamic scheduling of flexible manufacturing system using support vector machines," *Automation Science and Engineering, 2005. IEEE International Conference on* , vol., no., pp.387,392, 1-2 Aug. 2005

[9] Ci Chen; Binghai Zhou; Lifeng Xi, "A support vector machine based scheduling approach for a material handling system," *Natural Computation (ICNC), 2010 Sixth International Conference on* , vol.7, no., pp.3768,3772, 10-12 Aug. 2010

10 APPENDICES

10.1 APPENDIX A – PROGRAM FLOWCHART

10.2 APPENDIX B – SOURCE CODE

10.3 APPENDIX C – I/O LISTING