Operating Systems (IT250) Mini Project

On

Maximizing I/O throughput and Minimizing Performance Variation via Reinforcement Learning based I/O Merging for HDDs

Submitted by

171IT208 ANIKETH ANAGAWADI
171IT211 ARPITHA RAGHUNANDAN
171IT221 MADHUPARNA BHOWMIK
171IT237 SAURABH AGARWALA

Under the Guidance of

SAVITHA S

Date of Submission: $10^{th}\,April, 2019$

Department of Information Technology
National Institute of Technology Karnataka, Surathkal.

2018-2019

DECLARATION

We hereby declare that the Operating Systems (IT250) Report entitled "Maximizing I/O throughput and Minimizing Performance Variation via Reinforcement Learning based I/O Merging for HDDs" which is being submitted to the National Institute of Technology Karnataka Surathkal, in partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology in the Department of Information Technology, is a bonafide report of the work carried out by us. The material contained in this project report has not been submitted to any other University or Institution for the award of any degree.

Signature of the Students

Place : NITK, SURATHKAL

Date : 10 April 2019

CERTIFICATE

This is to certify that the Report entitled "Maximizing I/O throughput and Minimizing Performance Variation via Reinforcement Learning based I/O Merging for HDDs" has been presented by Aniketh Anagawadi , Arpitha Raghunandan , Madhuparna Bhowmik and Saurabh Agarwala , during the even semester of the academic year 2018 - 2019, in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Information Technology.

Place : NITK, SURATHKAL

Date : 10 April 2019

ABSTRACT

In the field of computer science, Machine Learning is one of the fastest growing and expanding domains. Reinforcement Learning is an important type of Machine Learning where an agent learns how to behave in an environment by performing actions and seeing the results. In this project, we apply techniques used in Reinforcement Learning to maximize I/O throughput and minimize performance variation during I/O merging for HDDs.

Hard disk drives (HDDs) are one of the slowest components of computer system and need to be accessed in an efficient manner. Multiple I/O requests may arrive from different processes and only one I/O request can be served at a time by the disk controller. Hence, the other I/O requests need to wait in a waiting queue and need to be scheduled. Two or more request may be far from each other on the HDD and so can result in greater disk arm movement. I/O merging is the merging of I/O requests in a queue, which may help improve throughput and reduce performance variance.

Deciding which I/O requests must be merged for optimal throughput is manually not feasible. Thus, this project aims at using Reinforcement Learning algorithms like Deep Q-Learning and Policy Gradient to automate the process.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The project aims to maximize the I/O throughput and minimize the performance variation during I/O merging for HDDs by proposing the following improvements to existing implementations.

1.1 Determining the storage location of data

A contiguous block of memory for related data increases the I/O throughput while reading data. The same has been implemented and determined.

1.2 Determining I/O Requests to merge

This is a decision problem. Various ML Algorithms under Reinforcement learning such as Q Learning and Policy Gradient exist for the purpose, which are discussed in further sections, and implemented.

1.3 Storing the merged I/O Requests

To solve this problem the Disjoint Set Union Data Structure has been employed. It has been discussed and implemented in further sections.

Chapter 2

Literature Review

2.1 Background

Merge technique is widely adopted by I/O schedulers to maximize the system I/O throughput. However merging operation could also degrade the latency of individual I/O by merging requests which have their corresponding data stored quite apart from each other leading to a greater disk arm movement, thus incurring prolonged I/O latencies and enlarged I/O variations of I/O requests.

These latencies are negligible in case of Solid State Disks (SDDs), because of their fast seek time, but in case of HDDs these result in a significant latency which can't be ignored. Hence, it is necessary that the I/O requests be merged in a way which results in maximum I/O throughput and minimum performance variation.

There are few Machine Learning based implementations to choose the I/O requests to merge in order to decrease the latency. These implementations use Reinforcement Learning Algorithms like Q Learning, Policy Gradient, SARSA ,Deep Q-Network, etc. Out of these, Q Learning and Policy Gradient which are further discussed below, have been used in this project for making decisions on which requests to merge and the results of both has been compared.

Q-learning is an off-policy and model-free Reinforcement Learning algorithm. Model-free means that there is no need to know the details of the environment i.e how and with what probability the next state is being generated, given current state and action. And off-policy means action-value function, Q, directly approximates the optimal action-value function, independent of the policy being followed.

Policy Gradients are a family of model-free reinforcement learning algorithms.In DQN and DRQNs where given a state, the Q-values of the possible actions were found where the Q-values are the expected return for the episode received from that state if that action is

selected. And an epsilon-greedy strategy is used to select the action. However, in Policy Gradients method instead of approximating the action value function, the optimal policy is directly learned.

2.2 Identified Gaps

Several roadblocks exist in current I/O requests merging. They include:

- Non contiguous storage of data of a request leading to increased seek time.
- Merging continuous I/O requests irrespective of the location of the data related to them
- Inappropriate merging Data Structure for storing the merge requests.

2.3 Problem Statement

The main problem statement is to to maximize I/O throughput and minimize performance variation during I/O merging for HDDs.

2.4 Objectives

The key objectives to be realized by this project are:

- Determining contiguous storage locations in HDD to store related data together.
- Merging appropriate I/O requests based on the result from the two Reinforcement learning algorithms namely, Q Learning and Policy Gradient and comparing the results of both.
- Using the Disjoint Set Union (DSU) data structure to store the merged I/O requests.

Chapter 3

Methodology

3.1 Reinforcement Learning Algorithms

Reinforcement Algorithms

3.1.1 Deep Q-Learning

Q-learning is an off-policy and model-free Reinforcement Learning algorithm. Model-free implies that the details of the environment, i.e how and with what probability the next state is being generated, given current state and action, need not be known. Off-policy implies that the action-value function, Q, directly approximates the optimal action-value function, independent of the policy being followed.

3.1.2 Policy Gradient

Policy Gradients are a family of model-free reinforcement learning algorithms. DQN and DRQNs use state to find the Q-values of the possible actions, where the Q-values are the expected return for the episode from that state, if that action is selected. An epsilon-greedy strategy is used to select the action. However, in the Policy Gradients algorithm, instead of approximating the action value function, the optimal policy is directly learned.

3.2 Hard Disk Drives

3.2.1 Seek Time

Seek time is the time taken to locate the disk arm to a specified track where the data is to be read or write.

3.2.2 Rotational Latency

Rotational Latency is the time taken by the desired sector of disk to rotate into a position so that it can access the read/write heads.

3.2.3 Transfer Time

Transfer time is the time to transfer the data. It depends on the rotating speed of the disk and number of bytes to be transferred.

3.2.4 Disk Access Time

$$\text{Disk Access Time} = \text{Seek Time} + \text{Rotational Latency} + \text{Transfer Time}$$

3.2.5 Disk Response Time

Response Time is the average of time spent by a request waiting to perform its I/O operation. Average Response time is the response time of the all requests. Variance Response Time is measure of how individual request are serviced with respect to average response time.

This project aims at reducing disk access time and improving disk response time, which in turn improves throughput.

Chapter 4

Implementation

4.1 Work Done

4.1.1 Disjoint Set Data Structure

This data structure was used to store merged and unmerged I/O requests. It suppor s three major operations makeSet, Union and findSet, making it a suitable data structure to store the I/O requests for this problem statement.

4.1.2 Hard Disk Drives (HDDs)

A class in python was used to simulate a Hard Disk Drive. It stores the position of the read/write head. It further calculates the time required for the various I/O requests. It was populated by randomly generated I/O requests to simulate the data stored on a real HDD.

4.1.3 Reinforcement Learning Algorithms

Reinforcement Learning algorithms were used to check when and where I/O merging was required to maximize throughput. The algorithms learned when I/O requests must be merged for optimal results. As a part of this project two algorithms were used: Q-Learning and Policy Gradient. The characteristics of these algorithms were:

- Environment - Simulation of Hard Disk
- State - Physical address of the I/O requests
- Action - Merge or not Merge
- Reward - Function of time taken to complete the I/O requests in the queue and individual I/O's response time.

The pseudo code for their implementations can be seen below.

Table 4.1: Pseudo Code: Q-Learning

Psuedo Code: Q-Learning
Initialize Q(s,a) arbitrarily
Repeat (for each episode):
Initialize s
Repeat (for each step of episode):
Choose a from s using policy derived from Q (e.g. epsilon-greedy)
Take action a, observe r,s'
Q(s,a) $<$ $--$ Q(s,a) + $alpha$ [r + $gamma$ * max($alpha$).Q(s',a') - Q(s,a)]
s $<$ $--$ s'
until s is terminal

Table 4.2: Pseudo Code: Policy Gradient

Psuedo Code: Policy Gradient
Initialize parameters
For i in number of episodes:
Generate trajectory s0, a0, r0, s1, a1, r1, ... , s(t-1), a(t-1), r(t-1), s(t) using the policy
For t in timestep:
Estimate the return value
Evaluate the policy and update the parameters.

4.2 Results and Analysis

Two reinforcement learning algorithms were used for deciding the merging of I/O requests: Q-Learning and Policy Gradient. The results are summarized in the graphs below.

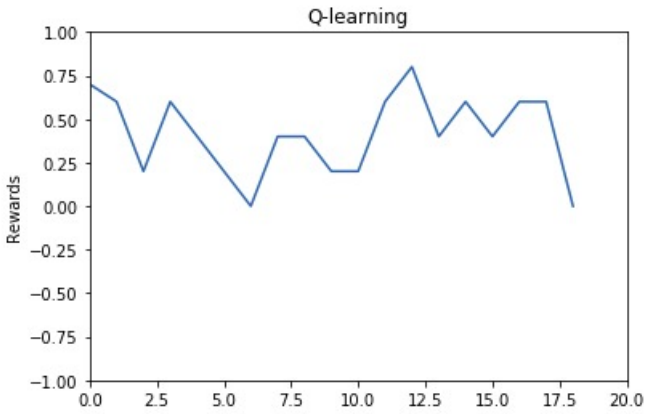

Figure 4.1: Q-Learning

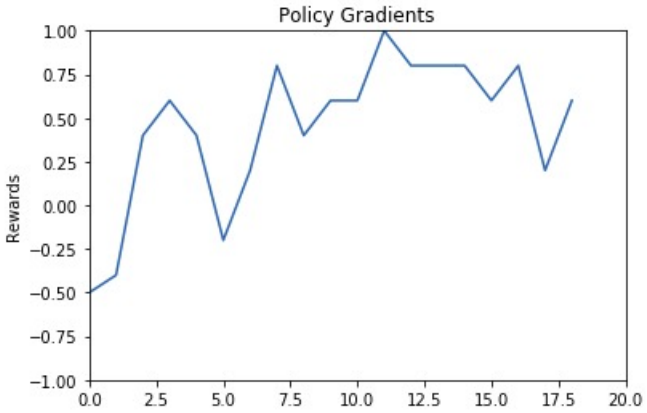

Figure 4.2: Policy Gradient

The graphs plot rewards against the number of episodes. In reinforcement learning, higher reward implies correct learning. As can be seen above, the rewards for Policy Gradient algorithm are higher on an average as compared to rewards for Q-Learning algorithm. After around 10 episodes, the reward for the Q-Learning algorithm is an average of around 0.5, whereas the reward for the Policy Gradient algorithm is an average of around 0.75. This clearly indicates that the I/O merging learned by the Policy Gradient algorithm is better for the given problem statement.

4.3 Innovative Work

The research paper[1] suggested using the Q-Learning Reinforcement Learning algorithm to maximize throughput and minimize performance variations during I/O merging. Our project implements the Policy Gradient Reinforcement Learning algorithm for the same. As can be seen clearly from the above section, Policy Gradient performs better than Q-Learning for I/O merging in Hard Disk Drives.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

Deciding which requests to merge in an I/O queue is not an easy task. Reinforcement Learning simplifies this process. The given algorithms learn by trial and error which kind of I/O merges will help improve performance. This project shows that the Policy Gradient is an efficient Reinforcement Learning algorithm which can be used to maximize throughput and minimize performance variations.

5.2 Future Work

The project can be enhanced with respect to the conditions of the Reinforcement Learning algorithms. The state representation can be improved from normalized values of sector number, track number etc. to representation learned by an one layer Embedding layer. This input for the neural network should result in better convergence and faster learning. The number of I/O requests given at a time can be increased with the use of better hardware support like GPUs. Reward function can be improved by considering the priority of I/O requests. Even though Policy Gradient does worked better than Q-learning in this case, it is prone to getting stuck at a local minimum and therefore more stable algorithms can be used.

Bibliography

[1] Chao wu, Cheng Ji, Qiao Li, Chenchen Fu and Chun Jason Xue. *Maximizing I/O throughput and Minimizing Performance Variation via Reinforcement Learning based I/O Merging for SSDs*. 2018 International Conference on Compilers, Architectures and Synthesis for Embedded Systems (CASES), Turin, Italy, 30 Sept - 5 Oct. 2018. `https://ieeexplore.ieee.org/document/8516832`

[2] Tutorial by OpenAI *Spinning up in Deep RL*. `https://openai.com/blog/spinning-up-in-deep-rl`

[3] Sutton and Barto. *Reinforcement Learning - An Introduction*. `https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf`

[4] Madhuparna Bhowmik *Introduction to Q-Learning*. IEEE-NITK, December 2018

[5] Madhuparna Bhowmik *Deep Recurrent Q-Network*. IEEE-NITK, January 2019

[6] Madhuparna Bhowmik *Policy Gradients*. IEEE-NITK, January 2019