BOSTON UNIVERSITY

COLLEGE OF ENGINEERING

Dissertation

BIG DATA PROCESSING FOR MACHINE LEARNING TASKS WITH RUST

by

SHINSAKU OKAZAKI

B.S., Seikei University, 2018

Submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

2020

Approved by

First Reader

Kia Teymourian, PhD
Professor of Computer Science

Second Reader

First M. Last
Associate Professor of . . .

Third Reader

First M. Last
Assistant Professor of . . .

Facilis descensus Averni;
Noctes atque dies patet atri janua Ditis;
Sed revocare gradum, superasque evadere ad auras,
Hoc opus, hic labor est. Virgil (from Don's thesis!)

Acknowledgments

Here go all your acknowledgments. You know, your advisor, funding agency, lab mates, etc., and of course your family.

As for me, I would like to thank Jonathan Polimeni for cleaning up old LaTeX style files and templates so that Engineering students would not have to suffer typesetting dissertations in MS Word. Also, I would like to thank IDS/ISS group (ECE) and CV/CNS lab graduates for their contributions and tweaks to this scheme over the years (after many frustrations when preparing their final document for BU library). In particular, I would like to thank Limor Martin who has helped with the transition to PDF-only dissertation format (no more printing hardcopies – hooray !!!)

The stylistic and aesthetic conventions implemented in this LaTeX thesis/dissertation format would not have been possible without the help from Brendan McDermot of Mugar library and Martha Wellman of CAS.

Finally, credit is due to Stephen Gildea for the MIT style file off which this current version is based, and Paolo Gaudiano for porting the MIT style to one compatible with BU requirements.

Janusz Konrad

Professor

ECE Department

BIG DATA PROCESSING FOR MACHINE LEARNING TASKS WITH RUST

SHINSAKU OKAZAKI

Boston University, College of Engineering, 2020

Major Professors: First M. Last, PhD
Professor of Electrical and Computer Engineering
Secondary appointment

First M. Last, PhD
Professor of Computer Science

ABSTRACT

Have you ever wondered why this is called an *abstract*? Weird thing is that its legal to cite the abstract of a dissertation alone, apart from the rest of the manuscript.

Contents

List of Tables

List of Figures

List of Abbreviations

The list below must be in alphabetical order as per BU library instructions or it will be returned to you for re-ordering.

CAD	....	Computer-Aided Design
CO	....	Cytochrome Oxidase
DOG	....	Difference Of Gaussian (distributions)
FWHM	....	Full-Width at Half Maximum
LGN	....	Lateral Geniculate Nucleus
ODC	....	Ocular Dominance Column
PDF	....	Probability Distribution Function
$\mathbb{R}^2$	....	the Real plane

Chapter 1

Introduction

1.1 Problem Description

Many of popular open source cluster computing frameworks for large scale data analysis, such as Hadoop and Spark, allow programmers to define objects in a host languages, such as Java. The objects are then managed in RAM by the language and its runtime, Java Virtual Machine in the case of Java and Scala. Storing objects in memory enables machine to process iterative computation. One of the fundamental tasks for recent big data analysis is analysis using Machine Learning Algorithms, which require iterative process. As the amount of data increases, memory is required to keep many objects. Therefore, memory management plays a critical role in this task.

Memory management in Java and Scala is performed by garbage collection. The garbage collection brings a significant advantage for programmers by removing responsibility for planning memory management by themselves. Instead, JVM monitors the state of memory and performs garbage collection at certain points. However, these monitoring and auto-execution of garbage collection cost additional computation and might consume computation resources which should be used for data processing. This can significantly decrease performance of the computation.

In contrast, memory management in system language, such as C++, relies on programmers' decision for when to allocate and deallocate memory. The functions, malloc/free consume most of the memory management. Proper implementation of system language for big data processing can be overperform the implementation in host language. Nevertheless, implementing C++ performing proper memory management and guaranteeing security can be unproductive and complicated.

Considering the issue of memory management, we introduce solution based on unique memory management methods implemented in Rust, ownership and borrowing. This unique concepts in Rust secure codes and perform memory management without monitoring memory or calling functions. We introduce implementations of machine learning algorithms in both Java and Rust to assess performances of each memory management system for iterative big data processing tasks.

1.2 Memory Management

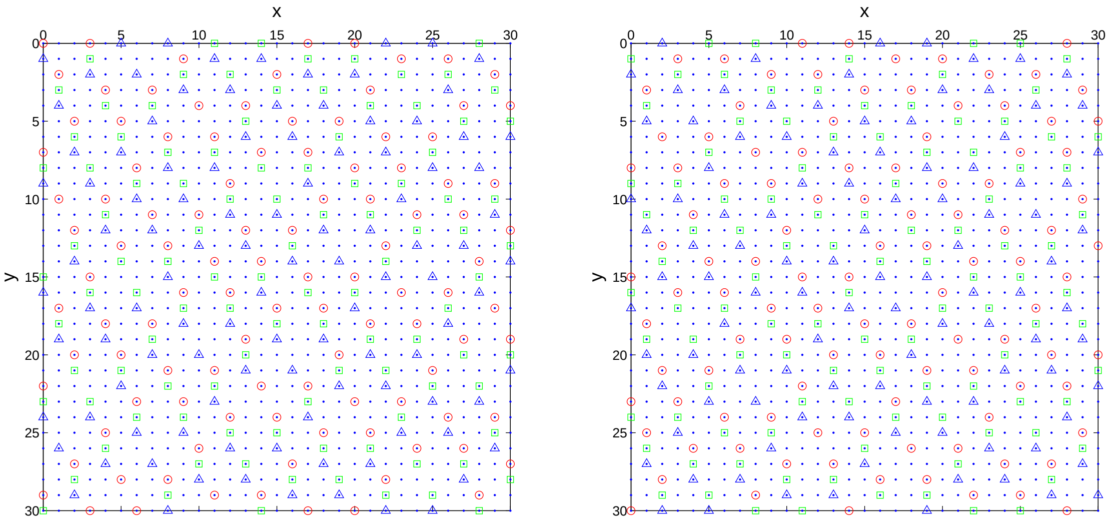
Each value in Rust has a variable called its owner. This owner has information about the value, such as location in memory, length and capacity of the value. This owner can live on the scope associated with its life time. When the owner goes out of it's scope, the value will be dropped. When a value already assigned to a variable is assigned to another variable, if the value is allocated on heap its information is copied to the new owner and drop the old owner disabling old variable. Similar thing happens when we pass variable to parameter of function. After passing a variable to a parameter, all the information is copied to new owner through the parameter and old owner is no longer available. The new owner can only live in the function and the object will be dropped. In this case, we have no longer access to the object after the function. To avoid this, Rust has a concept called borrowing. We can set reference for the parameter of function and use the reference for operation within function and drop the reference, but not the ownership.

Chapter 2

Body of my thesis

2.1 Some results

Here goes all the important stuff, likely with a lot of graphics like this:



(a)

(b)

Figure 2·1: Assignment of single-view intensities to RGB components: (a) view #1; and (b) view #2.

You will also be using a lot of citations. Here is the format required in the dissertation: (Lamport, 1985),(Debreuve et al., 2001).

In all likelihood, you will need to insert tables. See one example on the next page.

Table 2.1: Absolute disparity error per pixel for the test data from Fig. 2·1 and different parameter values. In each experiment one parameter is adjusted while other parameters are unchanged.

$\eta = 6000$, $\mu = 2000$

K	u_1	u_2
3	0.52	0.46
7	0.47	0.43
10	0.35	0.36
12	0.37	0.36

$K = 10$, $\mu = 2000$

η	u_1	u_2
1000	0.54	0.45
3000	0.43	0.40
6000	0.35	0.36
9000	0.37	0.37

$K = 10$, $\eta = 6000$

μ	u_1	u_2
100	1.00	1.16
1000	0.53	0.47
2000	0.35	0.36
3000	0.44	0.43

Of course, there must be a Table of Contents at the beginning of the thesis.

Chapter 3

Conclusions

3.1 Summary of the thesis

Time to get philosophical and wordy.

IMPORTANT: In the references at the end of thesis, all journal names must be spelled out in full, except for standard abbreviations like IEEE, ACM, SPIE, INFOCOM, ...

Appendix A

Proof of xyz

This is the appendix.

References

Debreuve, E., Barlaud, M., Aubert, G., Laurette, I., and Darcourt, J. (2001). Space-time segmentation using level set active contours applied to myocardial gated SPECT. *IEEE Trans. Med. Imag.*, 20(7):643–659.

Lamport, L. (1985). *LaTeX—A Document Preparation System—User's Guide and Reference Manual*. Addison-Wesley.

CURRICULUM VITAE

Joe Graduate

Basically, this needs to be worked out by each individual, however the same format, margins, typeface, and type size must be used as in the rest of the dissertation.