Replication of Concurrent Applications in a Shared Memory Multikernel

Yuzhong Wen

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Science and Application

Binoy Ravindran, Chair
Ali R. Butt Co-Chair
Dongyoon Lee

June 17, 2016
Blacksburg, Virginia

Keywords: blah, blah

Replication of Concurrent Applications in a Shared Memory Multikernel

Yuzhong Wen

(ABSTRACT)

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc nec elit molestie, mattis mi a, consequat arcu. Fusce venenatis rhoncus elit. Morbi ornare, libero a bibendum pretium, nibh orci tristique mauris, in suscipit mauris nibh ac metus. Nullam in sem vitae nisi aliquet iaculis in a nibh. Aliquam lobortis quis turpis ut tempus. Sed eu sapien eu nisi placerat viverra pharetra eu turpis. Mauris placerat massa mi, auctor facilisis sem consequat in. Pellentesque sollicitudin placerat mi quis rhoncus. In euismod lorem semper, scelerisque leo et, dapibus diam. Suspendisse augue dui, placerat at finibus a, cursus vitae erat. Nam accumsan magna vitae lorem tincidunt, et rhoncus elit consequat.

Suspendisse ut tellus at ex suscipit sollicitudin ut ut elit. Nam malesuada molestie elit eget luctus. Donec id quam ullamcorper, aliquam mauris at, congue felis. Nunc dapibus dui sit amet nisl laoreet, eget rhoncus est tempor. Mauris in blandit mauris. Aenean vitae ipsum lacinia, blandit turpis et, feugiat purus. Mauris in finibus quam, ac dictum lorem. Nam dignissim luctus ante. Suspendisse risus felis, imperdiet a lobortis sed, suscipit ac dui. Nullam fermentum velit eu congue dictum. Pellentesque tempor dui vel nisl tristique, non sollicitudin odio elementum. In ultricies elementum mattis.

Vestibulum eget imperdiet eros. Proin bibendum sit amet felis quis dignissim. Aliquam convallis mauris ut sapien gravida, eu consequat lacus dignissim. Vivamus porttitor hendrerit nisl, sit amet suscipit lorem vestibulum ut. Donec id tellus condimentum, sollicitudin sapien vel, lobortis nulla. Donec et elit quis est tempor semper. Aliquam erat volutpat. In nec consectetur dui. Nullam aliquam diam at eros ultrices vehicula. Nulla nibh ex, condimentum vitae nisl sed, aliquet ultricies sapien. Suspendisse potenti. Suspendisse pellentesque tincidunt facilisis. Morbi sodales vulputate ex malesuada molestie. Vestibulum eget placerat nunc.

This work is supported by AFOSR under the grant FA9550-14-1-0163. Any opinions, findings, and conclusions expressed in this thesis are those of the author and do not necessarily reflect the views of AFOSR.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

William Shakespeare has profoundly affected the field of literature worldwide. In the United States there was a surge of Shakespearean literature starting in the 1960s, with the opening of the Montgomery Shakespearean festival and continuing into the present ...

Table 1.1: The Graduate School wants captions above the tables.

x	1	2
1	1	2
2	2	4

Chapter 2

Related Work

Chapter 3

Popcorn Linux Background

3.1 Multikernel Boot

3.2 Inter-Kernel Messaging Layer

3.3 Popcorn Namespace

3.3.1 FT PID

3.4 Network Stack Replication

Chapter 4

Deterministic Execution

4.1 Logical Time Based Deterministic Scheduling

Inspired by Kendo[**?**] and Conversion[**?**], this scheduling policy maintains a logical time for each task inside the Determinator's runtime. When __det_start is called, the system will only let the one which holds the "token" to proceed, the token is decided in the following way: only the task holds the minimal logical time can have the token, if several tasks have the same logical time, the one who has the smallest PID number gets the token. When __det_end is called, the logical time is increased by 1. As long as the replicated application updates logical time in a same way on both primary and secondary, they will sure end up with the same thread interleaving.

In this algorithm, if one running thread holds the token for too long, i.e. doesn't update its logical time frequently during its execution, other threads will be waiting for the token for a long time. In the worst case this will kill the parallelism of a multithreaded program. We provide __det_tick syscall to manually increase the logical time with a predefined value, in order to break the logical time imbalance. In the evaluation part we will show that we are able to achieve reasonable overhead with minimal effort of instrumenting application's code.

4.2 Balance the Logical Time

4.2.1 Execution Time Profiling

4.2.2 Non-deterministic External Events

4.3 Implementation

Chapter 5

Schedule Replication

5.1 Execute-Follow Execution Model

5.2 Implementation

Chapter 6

Additional Runtime Support

6.1 Synchronization Skipping

6.2 Syscall Synchronization

6.3 Modified Pthread Library

Chapter 7

Evaluation

7.1 Correctness Evaluation

7.1.1 Racey Benchmarks

7.2 PBZip2

7.2.1 Overhead Profiling

7.2.2 Results

7.3 Mongoose Webserver

7.3.1 Overhead Profiling

7.3.2 Results

7.4 Nginx Webserver

7.4.1 Overhead Profiling

7.4.2 Results

7.5 Redis Database Server

7.5.1 Overhead Profiling

7.5.2 Results

Chapter 8

Conclusion

Chapter 9

Bibliography