DYNAMIC BINARY TRANSLATION FOR DETERMINISTIC REPLAY

PIYUS KEDIA

AMARNATH AND SHASHI KHOSLA SCHOOL OF IT

INDIAN INSTITUTE OF TECHNOLOGY DELHI

NOVEMBER 2017

DYNAMIC BINARY TRANSLATION FOR DETERMINISTIC REPLAY

by

PIYUS KEDIA

Amarnath and Shashi Khosla School of IT

Submitted

in fulfilment of the requirements of the *Doctor of Philosophy*

to the

Indian Institute of Technology Delhi

November 2017

To *Nature*

Certificate

This is to certify that the thesis titled **"Dynamic Binary Translation for Deterministic Replay"** being submitted by **Piyus Kedia** is worthy of consideration for the award of the degree of **Doctor of Philosophy** and is a record of original bonafide research work carried out by him under my guidance and supervision and that the results contained in it have not been submitted in part or full to any other university or institute for award of any degree or diploma.

Dr. Sorav Bansal
Computer Sc. & Engg.
IIT Delhi

Acknowledgements

I would like to thank my loving parents for their love and affection and for the flexibility to do whatever I like. I would like to thank my Grandfather and Grandmother for teaching me to uphold the values they passed on to me at every stage of my life. I want to thank my teachers in school who introduced me to mathematics and science at a very early stage in my life.

I want to thank my supervisor Dr. Sorav Bansal without whom this thesis would not have been possible. If not for him I would perhaps have not done a PhD at all. Through him I was able to discover my true area of interest. He helped me immensely in writing my papers and running through my practice talks. His spending countless hours discussing my problems at the right times was a big driving force behind this thesis. It is a big privilege to be his student. I would also like to thank the anonymous reviewers for their useful comments that made my thesis more comprehensive.

I would like to thank my friends Dipanjan and Kinshuk for making my days memorable during my stay at IIT Delhi. Without them it would not have been so easy to complete this PhD. I want to thank all my lab mates for weird discussions we so often had which made the lab so much more lively and fun to work in. I would also like to thank my committee members for attending my PhD progress talks.

I want to thank MHRD and IBM for supporting me financially during my PhD.

Piyus Kedia

Abstract

We present an efficient software implementation to deterministically record and replay a full multiprocessor virtual machine (VM), including its guest OS kernel and applications. Deterministically replaying a shared-memory monolithic OS kernel (like Linux) presents a significant performance challenge and we demonstrate the use of dynamic binary translation to achieve this objective.

Dynamic binary translation (DBT) is a powerful technique with several important applications. System-level binary translators have been used for implementing a Virtual Machine Monitor [2] and for instrumentation in the OS kernel [28]. In current designs, the performance overhead of binary translation on kernel-intensive workloads is high. e.g., over 10x slowdowns were reported on the syscall nanobenchmark in [2], 2-5x slowdowns were reported on `lmbench` microbenchmarks in [28]. These overheads are primarily due to the extra work required to correctly handle kernel mechanisms like interrupts, exceptions, and physical CPU concurrency. Since the overhead of DBT is itself very high, we can not use it for improving deterministic replay performance. We present a kernel-level binary translation mechanism which exhibits near-native performance even on applications with large kernel activity. Our translator relaxes transparency requirements and aggressively takes advantage of kernel invariants to eliminate sources of slowdown. We have implemented our translator as a loadable module in unmodified Linux, and present performance and scalability experiments on multiprocessor hardware. Although our implementation is Linux specific, our mechanisms are quite general; we only take advantage of typical kernel design patterns, not Linux-specific features.

The biggest challenge in deterministically replaying a multiprocessor system is recording the order of shared memory reads and writes. A potential approach is to use the CREW (Concurrent Read Exclusive Write) protocol at page granularity [27] to track the order of shared memory reads and writes. Page-grained CREW protocol uses hardware page protection techniques (Extended Page Tables/Shadow Page Tables) to restrict the access privilege of the CPUs. CREW dictates that multiple CPUs can read from a page by acquiring shared-access privilege (e.g., reader lock) of that page concurrently, but for writing to a page, they need to acquire exclusive-access (e.g., writer lock) privilege. Every transfer of privilege is recorded in order to reproduce the same transition during replay. This page-granular scheme, has been

demonstrated to work on selected user-level applications, but suffers from false sharing and huge shuttling between processors for workloads having a large amount of sharing (e.g., the Linux kernel). In contrast, we demonstrate an implementation of CREW at byte granularity using DBT to eliminate false sharing and achieve lower tracking overheads. To achieve this, we insert reader/writer locks before/after every shared memory access. We implement shadow memory using DBT, to store reader/writer locks (metadata) for each memory byte (data). This involves CREW-like ownership tracking of memory locations, which involves associating metadata (in shadow memory) with each memory location to store its ownership status. Our reader/writer lock implementation is optimized for the common case when one CPU acquires the same locks repeatedly.

Our implementation exhibits 3-9x recording overhead for several important kernel-intensive benchmarks on a four-processor machine, compared to 2-41x overheads of the best existing comparable approach.

सारांश

हम एक पूर्ण बहुप्रोसेसर आभासी मशीन (वीएम) जिसमें इसके अतिथि ओएस कर्नेल शामिल है, को रिकॉर्ड करने और निर्णायक रूप से पुनः चलाने के लिए एक कुशल सॉफ्टवेयर प्रस्तुत करते हैं| साझा स्मृति अखंड ओएस कर्नेल (जैसे लिनक्स) को निश्चित रूप से फिर से चालू करना एक बड़ी चुनौती है और हम इसे प्राप्त करने के लिए डायनामिक बाइनरी ट्रांसलेशन (डीबीटी) का उपयोग करते हैं|

डीबीटी एक शक्तिशाली तकनीक है, जिसके कई महत्वपूर्ण अनुप्रयोग हैं| सिस्टम-स्तरीय डीबीटी का उपयोग एक आभासी मशीन मॉनिटर [2] को चलाने करने के लिए और ओएस कर्नेल में इंस्ट्रूमेंटेशन [28] के लिए किया गया है। वर्तमान डिजाइनों में, कर्नेल-गहन वर्कलोड पर डीबीटी का प्रदर्शन निम्न स्तर का है। उदाहरण के तौर पर, [2] में सिसकॉल नैनोबेंचमार्क पर 10x से अधिक मंदी की सूचना दी गई थी, [28] एलएमबेंच के 2-5x मंदी के बारे में बताती है। यह मंदी मुख्य रूप से अतिरिक्त कार्य के कारण होती हैं, जो सही ढंग से कर्नेल तंत्र, जैसे कि इंटरप्ट, एक्सेप्षन और भौतिक (सीपीयू) संगामिति को संभालते हैं। चूंकि डीबीटी की लागत बहुत ही उच्च है, इसलिए हम निर्णायक रूप से एक बहुप्रोसेसर प्रणाली को पुनः चलाने के प्रदर्शन को सुधारने के लिए इसका इस्तेमाल नहीं कर सकते। हम एक कर्नेल-स्तरीय डीबीटी प्रस्तुत करते हैं, जो बड़े कर्नेल गतिविधि वाले अनुप्रयोगों पर भी शून्य लागत दर्शाती हैं। हमारे डीबीटी ने पारदर्शिता की आवश्यकताओं को सीमित किया है और मंदी के स्रोतों को खत्म करने के लिए अपरिवर्तनीय कर्नेल का आक्रामक रूप से लाभ उठाया है। हमने मूल लिनक्स में एक लोड करने योग्य मॉड्यूल के रूप में हमारे डीबीटी को निर्मित किया है, और बहुप्रोसेसर हार्डवेयर पर स्केलेबिलिटी प्रयोगों को प्रदर्शित किया है। यद्यपि हमारा कार्यान्वयन लिनक्स विशिष्ट है, हमारे तंत्र काफी सामान्य हैं; हम केवल विशिष्ट कर्नेल डिज़ाइन पैटर्न का लाभ उठाते हैं, लिनक्स-विशिष्ट विशेषताओं नहीं।

निर्णायक रूप से एक बहुप्रोसेसर प्रणाली को फिर से चलाने में सबसे बड़ी चुनौती साझा स्मृति पढ़ने और लिखने का क्रम रिकॉर्ड करना है। साझा स्मृति पढ़ने और लिखने के क्रम को ट्रैक करने के लिए पृष्ठ ग्रैन्युलैरिटी [27] पर सीआरईडब्ल्यू (समवर्ती पढ़ें अनन्य लिखें) प्रोटोकॉल एक संभावित दृष्टिकोण है। सीपीयू के एक्सेस विशेषाधिकार को प्रतिबंधित करने के लिए पृष्ठ ग्रैन्युलैरिटी सीआरईडब्ल्यू प्रोटोकॉल हार्डवेयर पेज सुरक्षा तकनीकों (विस्तारित पृष्ठ सारणी / छाया पृष्ठ सारणी) का उपयोग करता है|

सीआरईडब्ल्यू यह सुझाव देता है कि एकाधिक सीपीयू एक पेज से साझा-एक्सेस विशेषाधिकार प्राप्त कर सकते हैं (उदाहरण के लिए, पाठक ताला), लेकिन किसी पेज पर लिखने के लिए, उन्हें अनन्य प्रवेश (उदाहरण, लेखक ताला) विशेषाधिकार प्राप्त करने की आवश्यकता है| विशेषाधिकार का हर हस्तांतरण, पुनरावृत्ति के दौरान उसी स्थानांतरण को पुन: उत्पन्न करने के लिए दर्ज किया जाता है। यह पृष्ठ-ग्रैन्यूलर स्कीम, चयनित उपयोगकर्ता-स्तरीय अनुप्रयोगों पर कार्य करने के लिए दिखाया गया है, लेकिन यह झूठी साझाकरण से ग्रस्त है, और बड़ी मात्रा में साझाकरण (उदाहरण के लिए, लिनक्स कर्नेल) वाले वर्कलोड के लिए प्रोसेसर के बीच विशेषाधिकार के एक बड़ी संख्या में हस्तांतरण की संभावना है। इसके विपरीत, हम झूठे साझाकरण को खत्म करने और कम रिकॉर्डिंग लागत हासिल करने के लिये डीबीटी का उपयोग करके बाइट ग्रैन्युलैरिटी सीआरईडब्ल्यू प्रोटोकॉल की रचना करते हैं। इसके लिए, हम प्रत्येक साझा स्मृति एक्सेस के पहले/बाद में पाठक/लेखक तालों का उपयोग करते हैं। हम प्रत्येक स्मृति बाइट (डेटा) के लिए पाठक/लेखक तालों (मेटाडेटा) को संग्रह करने के लिए डीबीटी का उपयोग करके छाया मेमोरी की रचना करते हैं। इसमें सीआरईडब्ल्यू की तरह ही स्मृति स्थानों की स्वामित्व ट्रैकिंग शामिल होती है, जिसमें प्रत्येक स्मृति स्थान को उसके स्वामित्व की स्थिति को स्टोर करने के लिए मेटाडेटा (छाया मेमोरी में) के साथ संबद्ध जोड़ना शामिल होता है| हमारे पाठक/लेखक ताले सामान्य परिस्थिति के लिए अनुकूलित हैं, जब एक सीपीयू बार-बार एक ही ताले अधिग्रहित करते हैं|

हमारी रचना, चार-प्रोसेसर मशीन पर कई महत्वपूर्ण कर्नेल-गहन मानक के लिए 3-9x रिकॉर्डिंग लागत प्रदर्शित करती है, जबकि सबसे अच्छा मौजूदा तुलनीय दृष्टिकोण की लागत 2-41x है।

Table of contents

List of figures

List of tables