08441 Abstracts Collection
Emerging Uses and Paradigms for Dynamic Binary Translation
— Dagstuhl Seminar —

Bruce R. Childers[1], Jack Davidson[2], Koen De Bosschere[3] and Mary Lou Soffa[4]

[1] University of Pittsburgh, USA
childers@cs.pitt.edu
[2] University of Virginia, USA
jwd@virginia.edu
[3] Gent University, B
Koen.DeBosschere@elis.ugent.be
[4] University of Virginia, USA
soffa@virginia.edu

Abstract. From 26.10. to 31.10.2008, the Dagstuhl Seminar 08441 "Emerging Uses and Paradigms for Dynamic Binary Translation " was held in Schloss Dagstuhl – Leibniz Center for Informatics. During the seminar, several participants presented their current research, and ongoing work and open problems were discussed. Abstracts of the presentations given during the seminar as well as abstracts of seminar results and ideas are put together in this paper. The first section describes the seminar topics and goals in general. Links to extended abstracts or full papers are provided, if available.

Keywords. Dynamic binary translation, Virtual machines

08441 Final Report – Emerging Uses and Paradigms for Dynamic Binary Translation

Software designers and developers face many problems in designing, building, deploying, and maintaining cutting-edge software applications-reliability, security, performance, power, legacy code, use of multi-core platforms, and maintenance are just a few of the issues that must be considered. Many of these issues are fundamental parts of the grand challenges in computer science such as reliability and security.

Keywords: Dynamic binary translation, Virtual machines

Joint work of: Altman, Erik; Childers, Bruce R.; Cohn, Robert; Davidson, Jack; De Brosschere, Koen; De Sutter, Bjorn; Ertl, Anton M.; Franz, Michael; Gu, Yuan; Hauswirth, Matthias; Heinz, Thomas; Hsu, Wei-Chung; Knoop, Jens; Krall, Andreas; Kumar, Naveen; Maebe, Jonas; Muth, Robert; Rival, Xavier; Rohou, Erven; Rosner, Roni; Soffa, Mary Lou; Troeger, Jens; Vick, Christopher

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2009/1888

Challenges in Dynamic Binary Translation

Erik Altman (IBM TJ Watson Research Center, US)

The talk will start by looking at some of the many and varied successes of Dynamic Binary Translation (DBT). It will then examine and quantify some of DBT's shortcomings, and just how short it is.

Finally, the talk will look at several ways these shortcomings might be overcome. These ways include (1) identifying domains with the best chance for DBT success and (2) techniques for helping DBT achieve that success.

Keywords: Application-Specific, Semantic Knowledge, TLP, Oracle Parallelism

Pin: Binary Translation for Dummies

Robert Cohn (Intel - Hudson, US)

Binary translation and dynamic code generation are powerful techniques for enabling and accelerating tasks such as program analysis, performance modeling, emulation, and virtualization. For example, vmware and qemu incorporate custom binary translators. In contrast, there are systems that provide a generic infrastructure for binary translation such as Pin, Strata, and Valgrind and users build tools on top.

Pin's target audience is tool developers who are experts in domains such as program analysis, cache modeling, or workload characterization, but may not be familiar with compilation techniques. The challenge is to provide a programming model for dynamic code generation that allows domain experts to build their tool and not get bogged down in the details of instruction set semantics, isolation, and just in time compilation. The talk takes its title from a popular book series, with "Excel for Dummies" being a prime example. Users of Excel don't need to know how a spreadsheet program is implemented; they just want to solve a problem.

The presentation examines issues that make it easy or hard to write binary translation tools in Pin, such as programming models, ISA and OS independence, and performance. We discuss the things we got right and the things we wish we had done differently.

Using C for the Back End

M. Anton Ertl (TU Wien, AT)

If we want to implement a translator easily and portably, but with with good code quality, then translating through C is a good option.

While C is a static language, we can also use this technique for a dynamic translator with the help of a dynamic linker. The disadvantage of this approach is the large startup time; this disadvantage can be partially overcome by caching, batching, and seeding the cache. Some challenges for this technique in binary translation are modeling the (arbitrary) control flow in C and the compilation granularity of C. One example of using dynamic translation through C, although not in a binary translation context is the implementation of a foreign function interface.

The Surprising Versatility of the Trace Compilation Paradigm: How We Set Out To Build A Better Just-In-Time Compiler And Suddenly Found Ourselves At The Center Of The Second Browser War

Michael Franz (Univ. California - Irvine, US)

We have been investigating the use of Dynamo-style trace based optimization beyond traditional binary translation. Surprisingly, the paradigm is far more versatile than its original inventors imagined.

Our key breakthrough for making trace compilation so versatile is a novel intermediate representation, the Trace Tree, which is constructed lazily on-demand while a program is simultaneously executed, incrementally compiled, and optimized.

Our compilation technique is surprisingly competitive at much lower implementation complexity. Our academic prototype Java compiler has been able to attain a performance similar to commercial production compilers while using only about 1/7th of the memory footprint, 1/30th of the compile time, and 1/100th of the actual compiler size. Early experiments showed an even greater benefit for dynamically-typed languages such as JavaScript.

Our academic experiments are now being validated in one of the largest "real world" trials imaginable. Mozilla recently selected our Trace Tree compiler as the new JavaScript engine for Firefox 3.1 (due out in November 2008), with an expected 300 Million installations to come. Even in the first alpha prototype, the Trace Tree compiler's JavaScript performance is a surprising 700% higher than that of FireFox's previous compiler, and a staggering 15 times (1500%) faster than that of Internet Explorer. On the other hand, Google recently launched their Chrome browser, which uses a superbly well engineered but "traditional" control-flow based JavaScript compiler. As both browsers mature, the performance competition between them is likely to settle the question whether or not one should base future compilers on Trace Trees.

Keywords: Trace-based compilation, trace trees

Software Security Challenges: Direct Attack and Self-Protection

Yuan Gu (Cloakware/Irdeto - Ottawa, CA)

First, a brief introduction to Cloakware; and then, discuss software attacks and security and Cloakware security technology and solutions; and finally, present a number of software security hard problems that may be addressed by using Dynamic Binary Translation, as well DBT may raise new software security challenges.

My Dynamic Binary Instrumentation Wish List

Matthias Hauswirth (Universität Lugano, CH)

Dynamic binary rewriting has many different applications, including the translation between different instruction sets, the online optimization of executing programs, and the instrumentation of code at runtime.

In this talk we focus on dynamic binary instrumentation.

We show one specific use, the measurement of the perceptible performance of interactive applications, and, based on our experience of using different instrumentation systems, we present four wishes for future binary instrumentation tools:

1. the provision of a vertical view of behavior across system layers,
2. the minimization of the overhead through persistence, sampling, and parallelization,
3. the reduction of measurement perturbation, in particular with respect to time, and
4. the ability for the declarative specification of instrumentation.

Keywords: Dynamic binary instrumentation

(Maybe?) Existing Dynamic Binary Translation, Instrumentation, and Optimization Systems

Matthias Hauswirth (Universität Lugano, CH)

This is an idea which may or may not make sense. It is not necessarily a talk (although I could "present" the idea at some point if it is deemed useful).

Given that this seminar is looking at emerging trends, it might make sense to review what is already out there. We could use the seminar Wiki to collaboratively put together a survey of existing dynamic binary rewriting (aka software dynamic translation) systems. A coarse classification could be whether "rewriting" means "translation" (e.g. between ISAs), "instrumentation" (e.g. for profiling), or "optimization" (e.g. to reduce execution time). If I can find time in the next couple of days, I will seed that Wiki structure (see http://www.dagstuhl.de/wiki/index.php/Dynamic_Binary_Rewriting_Survey) by shamelessly reusing a section of Nicolas Nethercote's (Valgrind) dissertation, which compares 11 different dynamic binary instrumentation systems.

Towards automatically generating device emulation code

Thomas Heinz (Robert Bosch GmbH - Stuttgart, DE)

Software maintenance is an important issue for automotive suppliers as the life cycle of electronic control units can span up to 30 years. Binary translation is a promising approach to enable automatic porting of ECU software to a new hardware platform replacing an obsolete, original architecture. As legacy ECU software contains a significant amount of low level, device related code which is often intermingled with application code, it is necessary to provide full-system binary translation to be useful, i.e. binary translation incorporating device emulation.

The talk is based on the scenario of replacing an obsolete microcontroller (source) by a new one (target) while attempting to emulate the source peripheral devices directly on corresponding target devices with similar functionality. The talk sketches some ideas for an approach to automatically generate device emulation code based on a semantic description of device operations.

Keywords: Binary translation, device emulation, code generation

ADORE: An Adaptive Object Code Re-Optimization System

Wei-Chung Hsu (University of Minnesota, US)

ADORE (Adaptive Object Re-Optimization) is a dynamic binary optimizer. It can be automatically loaded into memory at a program's start-up. During the program's execution, ADORE monitors the performance and dynamically optimizes the code based on collected runtime profiles.

ADORE has several major components, including self-monitor, profiler, phase detector, trace builder, and optimizer. The self-monitor collects performance information from sampling the hardware PMUs and sends processed data to the profiler for further analysis. The profiler identifies the performance bottleneck,

such as cache misses or mis-speculations, as they happen in a running application. The trace builder locates the most frequently executed program regions and reshapes the code to execute more efficiently. ADORE has been implemented on two platforms: the IA-64 (Itanium-1 and Itanium-2) and SPARC (UltraSparc IV+). The ADORE/Itanium is good at dynamically inserting cache prefetches for delinquent loads, and ADORE/Sparc can further generate a helper thread on-the-fly to prefetch for the main thread.

Dynamic binary translation and optimization are often constrained because insufficient semantic information available for the executable. We are currently seeking support from compiler annotations to increase the power of dynamic binary optimization.

Towards Real-Time Dynamic Binary Translation: Some Notes on Recent Activities in the WCET and RT Field

Jens Knoop (TU Wien, AT)

In this talk I will report on recent activities in the field of Real-Time and Worst-Case Execution Time Analysis, most notably on the continuing efforts towards mastering the Worst-Case Execution Time Tool Challenge and the closely related Annotation Language Challenge which have recently been launched at the ISoLA 2006 symposium and the WCET 2007 workshop, respectively. Starting from my own work pursued in the ALL-TIMES and the CoSTA project on advanced compiler support for timing analyses of embedded real-time systems, and an overview of the SATIrE system used as a common and unifying infrastructure within these projects, I will show how these projects and the activities around them are embedded into the broader field of research on analysing real-time systems, and which connecting links this might provide for upcoming approaches towards Real-Time Dynamic Binary Translation.

This work has been partially supported by the 7th EU R&D Framework Programme under contract No 215068, "Integrating European Timing Analysis Technology (ALL-TIMES)", and by the Austrian Science Fund (FWF) under contract No P18925-N13, "Compiler Support for Timing Analysis (CoSTA)".

Dynamic Binary Translation for Generation of Cycle Accurate Architecture Simulators

Andreas Krall (TU Wien, AT)

In this talk we discuss our experiences in using the LLVM just-in-time compiler as code generator in a cycle accurate architecture simulator for pipelined architectures. The architecture simulator is generated from an architecture specification in a mostly structural architecture description language. The simulator contains an interpreter and dynamically translates first heavy executed basic blocks and later traces to machine code. Spreading of pipelined instructions over basic block boundaries is solved by basic block duplication. We present detailed results for a MIPS and a VLIW simulator. Simulation speeds of up to 500 MHz on a 2200MHz Athlon 64 processor are reached.

Joint work of: Fellnhofer, Andreas; Krall, Andreas; Riegler, David

Vertical Instrumentation

Jonas Maebe (Gent University, BE)

High level instrumentation allows access to rich semantic information. Low level instrumentation on the other hand provides insight into many things that are not always visible at higher levels. Vertical instrumentation connects the different levels and enables combining the best of both worlds.

Keywords: Vertical instrumentation, dynamic binary instrumentation, aspect-oriented programming

security and binary rewriting

Robert Muth (Google - New York, US)

In this talk we discuss a handful of applications of dynamic binary rewriting in the area of security. Both black hat and white hat applications are discussed.

Keywords: Dynamic binary rewriting, security, sandboxing, side channels

Certifying Compilation Correctness

Xavier Rival (ENS - Paris, FR)

Bugs in compilers may turn out extremely difficult to isolate, and can have a great impact on users. We will review a range of techniques that aim at certifying compilation correctness.

First, we will discuss the semantics of compilation (with or without optimizations), so as to choose what property should be proved to be preserved by compilation.

Then, we will consider several techniques that can be used to prove properties about compilation, including the translation of program invariants or types during the compilation process, the automated proof of equivalence between the source and the compiled code, and the formal proof of correctness of a compiler. For each of these techniques, we will discuss the advantages and drawbacks.

Keywords: Compilation, Compilation Correctness, Optimizations

Combining Processor Virtualization and Split Compilation for Heterogeneous Multicore Embedded Systems

Erven Rohou (INRIA - Rennes Bretagne Atlantique, FR)

Complex embedded systems have always been heterogeneous multicore systems. Because of the tight constraints on power, performance and cost, this situation is not likely to change any time soon. As a result, the software environments required to program those systems have become very complex too.

We propose to apply instruction set virtualization and just-in-time compilation techniques to program heterogeneous multicore embedded systems, with several additional requirements:

– the environment must be able to compile legacy C/C++ code to a target independent intermediate representation;
– the just-in-time (JIT) compiler must generate high performance code;
– the technology must be able to program the whole system, not just the host processor.

Advantages that derive from such an environment include, among others, much simpler software engineering, reduced maintenance costs, reduced legacy code problems... It also goes beyond mere binary compatibility by providing a better exploitation of the hardware platform.

We also propose to combine processor virtualization with split compilation to improve the performance of the JIT compiler. Taking advantage of the two-step compilation process, we want to make it possible to run very aggressive optimizations online, even on a very constraint system.

Keywords: Heterogeneous multicore, virtualization, compilation, bytecode, annotations

Extended Abstract: http://drops.dagstuhl.de/opus/volltexte/2009/1887

Dynamic Binary Translation Beyond Just Dynamic Binary Translatio

Roni Rosner (Intel Israel, IL)

(D)BT is everywhere, it takes many forms, it is used for many purposes in many ways. DBT is a powerful, mature technology, but not (yet?) a silver bullet. Can we make significantly more out of DBT (i.e., breakthrough)?

Keywords: Dynamic Binary Tranlation and Optimization

Dynamic Binary Translation for System Emulation

Jens Troeger (Microsoft Research - Redmond, US)

System emulation is an approach to boot and run an operating system and its applications inside a sandboxed execution environment: the emulator. In order to run the guest operating system, the emulator must provide transparent implementations for the guest hardware features used by the guest operating system. The code of the guest can be interpreted or Just-in-Time compiled; the state of the guest processor is usually implemented by memory data structures and possibly mapped to the host processor (eg register mapping); and the greater environment like guest memory, devices, exceptions and interrupts, timing, etc, are implemented in different fashions depending on need of the emulator and ability of the host system. Because system emulators run entire operating systems, performance is often critical.

In this presentation I share my experience with working on different system emulators, the challenges we faced, and where the performance bottle necks are. DBT is an approach to compile guest machine code into host machine code, usually on demand following the trace of execution. I briefly formalize system emulation and DBT, and then focus on performance and accuracy of the system emulator. Optimizations can be applied on both levels, instruction compilation and state implementation of the guest. As it turns out, however, most performance improvements for system emulation are gained not by optimizing code, but by optimizing the implementation of the guest machine state.

Keywords: Dynamic Binary Translation, System Emulation

The Portmeirion Project: Architecting Systems to Support Binary Translation

Christopher Vick (Sun Microsystems - Menlo Park, US)

The Portmeirion project seeks to extend the application of Sun's virtualization technologies to enable the implementation of virtual instruction set architectures. By co-designing a hardware architecture and a system virtual machine, Portmeirion creates the ability to efficiently run code written to multiple ISA's (in particular, SPARC, Java byte code and X86) on a single system, while allowing strong innovation in the development of the hardware platform because that platform is no longer encumbered with a massive burden of backward compatibility. One critical portion of the Portmeirion approach is the system virtual machine, called the Portmeirion Virtual Machine (PVM), and perhaps its most critical component is the Portmeirion Optimizing Binary Translator. This translator is an outgrowth and major extension of dynamic compilation technologies developed for the HotSpot Java Virtual Machine, applied to a whole new set of problems. Its purpose is to, along with targeted specialized hardware support, enable the efficient execution of code written to the virtual ISA (vISAs) such as SPARC on hardware with a different native ISA (nISA) that does not directly implement the vISA. The other critical portion of Portmeirion approach is the co-designed CPU architecture, which is designed to enhance the efficiency of the binary translator.

Keywords: Computer Architecture, Dynamic Binary Translation, Virtual Machines