A Retargetable System-Level DBT Hypervisor

Tom Spink, Harry Wagstaff, Björn Franke
Overview

- Host Physical Machine (x86)
- Guest Virtual Machine (Arm)
- Guest Virtual Machine (MIPS)
- Guest Virtual Machine (RISC-V)
- Captive
- Operating System
- Host Physical Machine (x86)
Use Cases

● Architectural exploration
  ○ New instruction sets
  ○ Instruction set extensions
  ○ Rapid prototyping

● Fast performance modelling

● Running unmodified systems
  ○ Behavioural analysis

● Host agnostic virtualisation services
Exploiting the Host VM

Host Virtual Machine (x86)

Captive DBT Unikernel

Arm Guest Page Tables

MMU Mapping Engine

x86 VM Page Tables
Offline Stage: GenSim

Arm-v8 Architecture Description

GenSim

Optimiser

Arm-v8 Guest Architecture Module
execute (addi)
{
    uint64 imm = decode_imm(inst.imm12, inst.shift);
    uint64 op1 = read_gpr_sp(inst.sf, inst.rn);

    if (inst.S) {
        if (inst.sf == 0) {
            write_gpr32(inst.rd, __builtin_adc32_flags((uint32)op1, (uint32)imm, 0));
        } else {
            write_gpr64(inst.rd, __builtin_adc64_flags(op1, imm, 0));
        }
    } else {
        write_gpr_sp(inst.sf, inst.rd, op1 + imm);
    }
}
action void add (Instruction sym_1_3_parameter_inst) [] {
    uint64 sym_14_0_rd
    uint64 sym_5_0_rn
    uint64 sym_9_0_rm
} {
    block b_0 {
        s_b_0_0 = struct sym_1_3_parameter_inst a;
        s_b_0_1 = bankregread 7 s_b_0_0;
        s_b_0_2 : write sym_5_0_rn s_b_0_1;
        s_b_0_3 = struct sym_1_3_parameter_inst b;
        s_b_0_4 = bankregread 7 s_b_0_3;
        s_b_0_5 : write sym_9_0_rm s_b_0_4;
        s_b_0_6 = read sym_5_0_rn;
        s_b_0_7 = read sym_9_0_rm;
        s_b_0_8 = binary + s_b_0_6 s_b_0_7;
        s_b_0_9 : write sym_14_0_rd s_b_0_8;
        s_b_0_10 = struct sym_1_3_parameter_inst a;
        s_b_0_11 = read sym_14_0_rd;
        s_b_0_12 : bankregwrite 0 s_b_0_10 s_b_0_11;
        s_b_0_13 : return;
    }
}

action void add (Instruction sym_1_3_parameter_inst) [] {
    block b_0 {
        s_b_0_0 = struct sym_1_3_parameter_inst a;
        s_b_0_1 = bankregread 7 s_b_0_0;
        s_b_0_2 = struct sym_1_3_parameter_inst b;
        s_b_0_3 = bankregread 7 s_b_0_2;
        s_b_0_4 = binary + s_b_0_1 s_b_0_3;
        s_b_0_5 = struct sym_1_3_parameter_inst a;
        s_b_0_6 : bankregwrite 0 s_b_0_5 s_b_0_4;
        s_b_0_7 : return;
    }
}
bool generator::translate_add (const test_decode_test_F1& insn, dbt_emitter& emitter) {
    basic_block *__exit_block = emitter.create_block();
    goto fixed_block_b_0;

    fixed_block_b_0: {
        auto s_b_0_1 = emitter.load_register(emitter.const_u32((uint32_t)(256 + (16 * insn.a))), dbt_types::u64);
        auto s_b_0_3 = emitter.load_register(emitter.const_u32((uint32_t)(256 + (16 * insn.b))), dbt_types::u64);
        auto s_b_0_4 = emitter.add(s_b_0_1, s_b_0_3);
        emitter.store_register(emitter.const_u32((uint32_t)(0 + (8 * insn.a))), s_b_0_4);
        goto fixed_done;
    }

    fixed_done:
    emitter.jump(__exit_block);
    emitter.set_current_block(__exit_block);
    if (!insn.end_of_block) emitter.inc_pc(emitter.const_u8(4));
    return true;
}
Online Stage: Captive

Arm-v8 Guest Architecture Module

Arm-v8 Guest System Image
- Guest User-space
- Guest OS Kernel

Captive

Guest Machine (Arm)
Execution Loop

Guest (Physical) PC

Is translation cached?

YES

Execute Translation

NO

Translate Basic-block

Store in cache
Basic-block Translation

(a) add x0, x0, #1
(b) mov x1, x0
(c) bl 0x100c
DAG Lowering

\texttt{add \ x0, x0, \ #1}
\texttt{mov (%rbp), \ %VREG0} \quad ; \text{Load guest reg. into temporary}
\texttt{add \ \$1, \ %VREG0} \quad ; \text{Add one.}
\texttt{mov \ %VREG0, (\%rbp)} \quad ; \text{Store temporary to guest reg.}

\texttt{mov \ x1, x0}
\texttt{mov (%rbp), \ %VREG1} \quad ; \text{Load guest reg. into temporary}
\texttt{mov \ %VREG1, \ 8(\%rbp)} \quad ; \text{Store temporary to guest reg.}

\texttt{bl \ 0x100c}
\texttt{lea \ 4(\%r15), \ %VREG2} \quad ; \text{Load PC+4 into temporary}
\texttt{mov \ %VREG2, \ 0xf8(\%rbp)} \quad ; \text{Store into guest reg.}
\texttt{add \ \$12, \ %r15} \quad ; \text{Increment PC by 12}
Exploiting Architectural Features

- Full control of the host MMU
- Use of privilege levels, e.g. user mode (ring 3) and kernel mode (ring 0)
- Use of privileged instructions
  - `in/out` for fast I/O
  - `syscall/sysret` for fast mode switching
  - Manipulation of control registers
- Use of special registers (`fs/gs`)
- Complete control over the ABI
  - Reservation of `r15` to represent guest machine PC
- Software and Hardware Interrupts
Evaluation

Speed-up over QEMU

Speed-up

400: perlbench
401: bzip2
403: gcc
429: mcf
445: gbmk
456: hminer
458: sjeng
462: libquantum
464: h264ref
471: omnietpp
473: astar
483: xalancbmk
Geo. Mean.
Questions?

https://gensim.org

The Edinburgh Simulation Group

Tom Spink, Senior Researcher: Captive
tspink@inf.ed.ac.uk

Kuba Kaszyk, PhD Student: GPUSim

Martin Kristien, PhD Student: Multicore Simulation

Harry Wagstaff, Former Member: GenSim

Björn Franke, Reader: BDFL