Why Use Simulation?

- Understanding real systems
- More inspectable
- Less dangerous
- Fault injection
- Debugging
- Prototype HW before expensive implementations

What is a Simulator?

- What to simulate? Instructions, memory etc.
- What happens when an operating system boots, traps, disk accesses etc.
- Simulation vs. Emulation

Full-System Simulation

- User program
  - Libraries
  - Database
  - Middleware
- Operating system
  - CPU
  - RAM
  - ROM
  - PCI
  - IDE
  - ISA
  - (Flash)
  - Disk
  - Network
  - LCD
  - Devices
- Real OS & Software
- Simulated hardware
Advantages with Full System Simulation

- OS debugging (such as, single stepping during boot, break on traps etc.).
- Performance tuning of applications and operating systems
- Fault injection
- Teaching and research
- Checkpointing
- Determinism
- Non intrusive – inspect without effecting state

Possible Problems

- Can I trust the results that my simulator have produced?
- How do I verify my results?
- Wisconsin experiments

Wisconsin Experiments
### Speed vs. Accuracy

- **Spice, etc.**
- **Gate/RTL model (~1,000,000x)**
- **"Cycle-accurate" architecture simulator (~10,000x)**
- **Emulator (~5x)**
- **Simics™**

### Virtutech AB - Simics

- Virtutech is a small Swedish company. 25-30 persons. Office in Stockholm, Uppsala and U.S.
- Simics is the only full system simulator that supports multiple targets and is capable of booting unmodified operating systems.

### Simics’s Three Modes

- **Fast mode**: No cache simulation. Just in time compilation (code name turbo).
- **Normal mode**: Simple cache simulation.
- **Out-of-order mode**: MAI (Micro Architecture Interface). Supports speculative execution, such as, branch and value prediction. Cache simulation etc.

### Supported architectures

- X86 (from 486 to Pentium4)
- AMD-64 (Hammer/Opteron)
- Itanium
- SUN SunFire & Serengeti (USII & USIII)
- PowerPC
- Alpha
- ARM, MIPS + lots of devices
Unmodified OSs tested/supported
- Windows (XP, NT, 98, 95 and more)
- Solaris 9 (and more)
- Linux
- Tru64
- VxWorks

Simulating Multiprocessors
- Arbitrary configurations
- Multiple processors
- Multiple nodes
- Multiple address spaces
- Arbitrary device configurations

Simics Architecture
Target vs. Host

- The target is the simulated system
- The host is the computer that runs Simics
- The different prompts:
  - `target#` - the target prompt: root on the target system
  - `host$` - the host prompt: user on the host system (xterm etc.).
  - `simics>` - the Simics prompt

Simics Dir Structure

Most of the files in the user dir install dir are soft links:
- LICENSE
- Makefile.in
- README
- README.SOURCE
- README.previous
- README.proxy
- Version
- configure
- doc
- licenses
- master-install

Real but for this course uninteresting files:
- import
- scripts
- config

Interesting dirs:
- home - home for different configurations. We will use the `sarek` configuration
- src - simics module source code
- v9-sol8-64 - binaries and build place

Simics Commands

- `simics> c`
  - Causes Simics to continue its simulation.
  - Pressing `Ctrl-c` when Simics is running causes it to stop.
- `simics> c 1000`
  - Causes Simics to run another 1000 machine instructions and then stop.
- `simics> help`
  - Gives help on selected topics. Try it out!
- `simics> help dark2-cache`
  - This help command will list all commands created for this assignment.

Simics Commands (cont)

- `simics> q`
  - Quit Simics.
- `simics> load-module`
  - Load a module into Simics.
- `simics> list-modules`
  - Prints a list of available modules and their current status, loaded or not.
hostfs

- **hostfs – accessing host files**
  - hostfs is a Virtutech module that makes it possible to mount the host’s file system in the simulated target’s operating system.

- **target# mount /host**
  - This will mount the host’s file system under /host on the target. The hostfs module has to be loaded (module-load hostfs).

- **target# ls /host/it/kurs/dark2**
  - This will list the contents on the host machine’s file system at /stud/it/dark2

---

Simics API

- A module can register multiple classes
- A class can register multiple attributes
- The configuration system uses classes and attributes for:
  - Checkpointing
  - Python command wrapping
  - And more...

---

Register a class

```c
conf_class_t *
SIM_register_class(const char *name,
                   class_data_t *class_data);
```

Example:
```c
static conf_class_t *my_class;
static class_data_t my_class_data;
```

```c
my_class = SIM_register_class("my-class-name",
                            &my_class_data);
```

---

Register an attribute

```c
int
SIM_register_attribute(conf_class_t *class_struct,
                       const char *attr_name,
                       get_attr_t get_attr,
                       lang_void *get_attr_data,
                       set_attr_t set_attr,
                       lang_void *set_attr_data,
                       attr_attr_t attr,
                       const char *doc);
```
Simics Command Files

The file dark2-cache.simics contains the following:

read-configuration /it/kurs/dark2/simics-checkpoints/booted-sarek-1p

@SIM_new_object("dark2-cache", "dc0")
@conf.dc0.size = 512
@conf.dc0.block_size = 64
@conf.phys_mem0.timing_model = conf.dc0

Command files can contain any Simics commands
Lines that starts with @ are interpreted by the Simics Python virtual machine.

"@conf.dc0.size = 512" calls the set-attribute function

Adding New Commands

Edit the commands.py file in the dark2-cache dir

def dc_resize_cmd(dc, size, block_size):
    dc.size = size;
    dc.block_size = block_size;

    new_command("resize", dc_resize_cmd,
                [arg(integer_t, "size"), arg(integer_t, "block-size")],
                namespace = "dark2-cache",
                type = "dark2-cache commands",
                short = "resize the cache",
                doc = "Resizes the cache. The long version ???");

STC

- Simics uses a fast address cache called ISTC and DSTC for instructions resp. data accesses.
- Direct mapped SW cache containing "safe" address translations
  - No traps are possible, that is, TLB and cache hit is guaranteed.
- This caching have to be disabled when cache simulations are performed.

HAP (happening)

- There are multiple haps that a user can register callbacks for.
- Initial-configuration – makes it possible to get a callback when the configuration phase is over.
- Magic-instruction – makes it possible to get a callback when such an instruction is executed. This is for example done in the magic-break-enable and magic-break-disable commands.
The Step- and Time-Queue

- Simics uses two queues that events can be posted on.
- The STEP QUEUE: incremented on instruction completions and interrupt events
- The TIME QUEUE: incremented each cycle
- The time queue is used when the out-of-order mode are used.

The Assignment

- Questions?