CS330: Getting Started Guide for GemOS

Introduction

`gemOS` is a teaching operating system, will be developed gradually during the course. As part of the assignments, you are required to design and implement several features in `gemOS`. Typically, operating systems execute on hardware (bare metal or virtual). For this course, we will use Gem5 architectural simulator (`http://gem5.org/Main_Page`). An architectural simulator simulates the hardware in software. In other words, all the hardware functionalities and communications are implemented in software using some programming language. For example, Gem5 simulator implements architectural elements for different architectures like ARM, X86, MIPS etc. Advantages of using software simulators for OS development are,

- Internal hardware state and operations at different components (e.g., decoder, cache etc.) can be profiled to better understand the hardware-software interfacing.
- Hardware can be modified for several purposes—make it simpler, understand implications of hardware changes on software design etc.
- Bugs during OS development can be better understood by debugging both hardware code and the OS code.

The getting started guide helps you to setup Gem5 and execute `gemOS` on it.

Preparing Gem5 simulator

System pre-requisites

- Git
- gcc 4.8+
- python 2.7+
- SCons
- protobuf 2.1+

On Ubuntu install the packages by executing the following command.

```
$ sudo apt-get install build-essential git m4 scons zlib1g zlib1g-dev libprotobuf-dev
protobuf-compiler libprotoc-dev libgoogle-perftools-dev python-dev python automake
```

Gem5 installation

Clone the gem5 repository from `https://gem5.googlesource.com/public/gem5`.

```
$ git clone https://gem5.googlesource.com/public/gem5
```

Change the current directory to `gem5` and build it with scons:

```
$ cd gem5
$ scons build/X86/gem5.opt -j9
```

In place of 9 in [-j9], you can assign a number equal to available processors in your system plus one. For example, if your system have 4 cores, then substitute -j9 with -j5. For first time it will take around 10 to 30 minutes depending on your system configuration. After a successful Gem5 is build, test it using the following command,

```
$ build/X86/gem5.opt configs/example/se.py --cmd=tests/test-progs/hello/bin/x86/linux/hello
```

The output should be as follows,

```
gem5 Simulator System.  http://gem5.org
gem5 is copyrighted software; use the --copyright option for details.
gem5 compiled Aug  4 2018 11:00:44
gem5 started Aug  4 2018 17:15:06
gem5 executing on BM1AF-BP1AF-BM6AF, pid 8965
command line: build/X86/gem5.opt configs/example/se.py \
              --cmd=tests/test-progs/hello/bin/x86/linux/hello
/home/user/workspace/gem5/configs/common/CacheConfig.py:50: SyntaxWarning: import * only \
              allowed at module level def config_cache(options, system):
Global frequency set at 1000000000000 ticks per second
warn: DRAM device capacity (8192 Mbytes) does not match the address range assigned (512 Mbytes)
0: system.remote_gdb: listening for remote gdb on port 7000
**** REAL SIMULATION ****
info: Entering event queue @ 0. Starting simulation...
Hello world!
Exiting @ tick 5941500 because exiting with last active thread context
```

Booting GemOS using Gem5

Gem5 execute in two modes—system call emulation (SE) mode and full system (FS) simulation mode. The example shown in the previous section, was a SE mode simulation of Gem5 to study an application. As we want to execute an OS, Gem5 should be executed in FS mode. There are some setup steps before we can execute `gemOS` using Gem5 FS mode. To run OS in full-system mode, where we are required to simulate the hardware in detail, we need to provide the following files,

- `gemOS.kernel`: OS binary built from the `gemOS` source.
- `gemOS.img`: root disk image
- `swap.img`: swap disk image

Gem5 is required to be properly configured to execute the `gemOS` kernel. The configuration requires changing some existing configuration files (in gem5 directory) as follows,

- Edit the `configs/common/FSConfig.py` file to modify the `makeX86System` function where the value of `disk2.childImage` is modified to `(disk('swap.img'))`.
- Edit the `configs/common/FSConfig.py` file to modify the `makeLinuxX86System` function where the value of `self.kernel` is modified to `binary('gemOS.kernel')`.
- Edit the `configs/common/Benchmarks.py` file to update it as follows,

```
elif buildEnv['TARGET_ISA'] == 'x86':
      return env.get('LINUX_IMAGE', disk('gemOS.img'))
```

Create a directory named `gemos` in `gem5` directory and populate it as follows,

```
/home/user/gem5$ mkdir gemos
/home/user/gem5$ cd gemos
/home/user/gem5/gemos$ mkdir disks; mkdir binaries
/home/user/gem5/gemos$ dd if=/dev/zero of=disks/gemOS.img bs=1M count=128
/home/user/gem5/gemos$ dd if=/dev/zero of=disks/swap.img bs=1M count=32
```

Build `gemOS` source (will be provided during assignments) by executing `make` and copy the binary to `gemos/binaries` directory. This step is necessary every time you build the OS and want to test it. For the time being, you can download `gemOS.kernel` from `https://www.cse.iitk.ac.in/users/deba/cs330/resources/gemOS.kernel`. We need to set the `M5_PATH` environment variable to the `gemos` directory path as follows,

```
/home/user/gem5$ export M5_PATH=/home/user/gem5/gemos
```

Now, we are ready to boot GemOS.

```
gem5$ build/X86/gem5.opt configs/example/fs.py
```

Gem5 output will look as follows,

```
gem5 Simulator System.  http://gem5.org
gem5 is copyrighted software; use the --copyright option for details.

gem5 compiled Aug 10 2018 20:24:30
gem5 started Aug 15 2018 18:01:10
gem5 executing on cse-BM1AF-BP1AF-BM6AF, pid 11898
command line: ./build/X86/gem5.debug configs/example/fs.py

Global frequency set at 1000000000000 ticks per second
warn: DRAM device capacity (8192 Mbytes) does not match the address range assigned (512 Mbytes)
info: kernel located at: /home/user/gem5/gemos/binaries/gemOS.kernel
Listening for com_1 connection on port 3456
      0: rtc: Real-time clock set to Sun Jan  1 00:00:00 2012
0: system.remote_gdb.listener: listening for remote gdb #0 on port 7000
warn: Reading current count from inactive timer.
**** REAL SIMULATION ****
info: Entering event queue @ 0.  Starting simulation...
warn: Don't know what interrupt to clear for console.
```

Execute the following command in another terminal window to access the `gemOS` console

```
/home/user$ telnet localhost 3456
```

At this point, you should be able to see the `gemOS` shell. Type help to see the available commands.