How to use the BigDataBench simulator versions

Zhen Jia

Institute of Computing Technology, Chinese Academy of Sciences

BigDataBench Tutorial
MICRO 2014 Cambridge, UK
Objectives

- Facilitate architecture researchers’ study of Big Data in:
  - Obtaining performance characteristics for new architectures
  - Architectural exploration
  - Experimentally determining the benefits of new designs
  - Processor’s innovation using existing hardware
Big Data Simulation Requirements

- Big Data workloads’ characteristics:
  - Large input: data set can not fit into memory
  - Communicate among nodes
  - Deep software stacks

- Simulator running Big Data workloads should support:
  - Multiple peripheral devices
  - Networks of systems
  - Executing real OS and applications
Simulators used

- We provide two simulator versions benchmark for Big Data applications:
  - MARSSx86 version: use X86 as the instruction set architecture and run Linux.
  - Simics version: use SPARC as the instruction set architecture and run Solaris.
Simulator Characteristics

- Both of MARSSx86 and Simics:
  - Support full-system simulation
  - Provide accurate simulation interfaces
  - Get detailed performance data
  - Fast and easy evaluation
  - Adjust hardware parameters in a small overhead
  - Have a wide range of user groups
## Simulator version workloads

<table>
<thead>
<tr>
<th>No.</th>
<th>Workload name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Hadoop-WordCount</td>
</tr>
<tr>
<td>2</td>
<td>Hadoop-Grep</td>
</tr>
<tr>
<td>3</td>
<td>Hadoop-NaiveBayes</td>
</tr>
<tr>
<td>4</td>
<td>Cloud-OLTP-Read</td>
</tr>
<tr>
<td>5</td>
<td>Hive-Differ</td>
</tr>
<tr>
<td>6</td>
<td>Hive-TPC-DS-query3</td>
</tr>
<tr>
<td>7</td>
<td>Spark-WordCount</td>
</tr>
<tr>
<td>8</td>
<td>Spark-Sort</td>
</tr>
<tr>
<td>9</td>
<td>Spark-Grep</td>
</tr>
<tr>
<td>10</td>
<td>Spark-Pagerank</td>
</tr>
<tr>
<td>11</td>
<td>Spark-Kmeans</td>
</tr>
<tr>
<td>12</td>
<td>Shark-Project</td>
</tr>
<tr>
<td>13</td>
<td>Shark-Orderby</td>
</tr>
<tr>
<td>14</td>
<td>Shark-TPC-DS-query8</td>
</tr>
<tr>
<td>15</td>
<td>Shark-TPC-DS-query10</td>
</tr>
<tr>
<td>16</td>
<td>Impala-Orderby</td>
</tr>
<tr>
<td>17</td>
<td>Impala-SelectQuery</td>
</tr>
</tbody>
</table>

Our architecture subset workloads:
Use Case---MARSSx86
MARSSx86’s brief introduction

- MARSS86 is an open source, full system simulation tool built on QEMU and PTLsim
- Multicore simulation environment for the x86-64 ISA
- Detailed models for Coherent Caches and On-Chip Interconnections
MARSS Framework

- Framework includes
  - Cycle accurate simulation models for CPUs, cache, interconnect, DRAM controller etc.
  - Emulation models for CPU, Disk, NIC, etc. from QEMU
- Simulate *unmodified* software stack
- Runs on top of unmodified Linux Kernel and x86-64 hardware

This slide is from MARSSx86 tutorial 2012
Two Modes of MARSS

- MARSS has two modes:
  - QEMU mode: functional simulation
  - MARSS mode: cycle accurate simulation

<table>
<thead>
<tr>
<th>Mode</th>
<th>Emulation (QEMU)</th>
<th>Simulation (MARSS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scope</td>
<td>Full-system simulator</td>
<td>Full-system simulator</td>
</tr>
<tr>
<td>Detail</td>
<td>Functional</td>
<td>Timing (performance)</td>
</tr>
<tr>
<td>Speed</td>
<td>Fast</td>
<td>Slow</td>
</tr>
<tr>
<td>Checkpoint</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Open source</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Pre-requirements of MARSSx86

MARSS can run on a Linux system with the following minimum requirements:

- X86 64 CPU cores with a minimum 2GHz clock and 2GB RAM (4GB RAM is preferred)
- C/C++ compiler, gcc or icc
- SCons compilation tool version 1.2 or later
- SDL Development Libraries (required for QEMU)
Compiling MARSS

Download the package and then:

- $ tar xf marss-0.4.tar.gz
- $ cd marss-0.4
- $ scons –Q [c=8]

By default it will compile single simulated core. To simulate more than one core, add an option `c=NUM_CORES`.
Preparing workloads for MARSS

1. Create qemu-image
2. Boot a simulation machine
3. Install OS
4. Install applications
5. Create workload and dataset
6. Run applications

We have done:  

Users need to do:
Simulated Cluster

- One master + one slave
We provide four QEMU images:
- marss-1.img: Master node of Impala-based workloads
- marss-2.img: Slave node of Impala-based workloads
- marss-3.img: Master node of Hadoop & Spark workloads
- marss-4.img: Slave node of Hadoop & Spark workloads

We provide two QEMU network config scripts:
- qemu-ifup: qemu-network-config-script for slaver node
- qemu-ifup2: qemu-network-config-script for master node
Start Master Node

- For master: 
  ```bash
  $ qemu-system-x86_64 -m 8192 -hda [path-to-marss-1.img] -monitor stdio -net nic,macaddr=52:54:00:12:34:55 -net tap,ifname=tap1,script=[path-to-qemu-ifup2]
  ```
Start Master Node
Start Slave Node

- For slave: `\$ qemu-system-x86_64 -m 8192 -hda [path-to-marss-2.img] -monitor stdio -net nic -net tap,ifname=tap0,script=[path-to-qemu-ifup]`
QEMU monitor:

```
[root@localhost marss-0.4]# qemu/qemu-system-x86_64 -m 8192 -hda /home/linux/inlong/marss-0.4/marss-1.img -vnc 172.16.50.5:3 -monitor stdio -net nic,macaddr=52:54:00:12:34:55 -net tap,ifname=tap1,script=/etc/qemu-ifup2 bring tap1 up
QEMU 0.14.1 monitor - type 'help' for more information
(qemu)
```
QEMU monitor:

(qemu) help
help|? [cmd] -- show the help
commit device|all -- commit changes to the disk images (if -snapshot is used) or backing files
q|quit -- quit the emulator
block_resize device size -- resize a block image
eject [-f] device -- eject a removable medium (use -f to force it)
drive_del device -- remove host block device
change device filename [format] -- change a removable medium, optional format
 screendump filename -- save screen into PPM image 'filename'
logfile filename -- output logs to 'filename'
log item[,...] -- activate logging of the specified items to '/tmp/qemu.log'
 savevm [tag|id] -- save a VM snapshot. If no tag or id are provided, a new snapshot is created
loadvm tag|id -- restore a VM snapshot from its tag or id
 delvm tag|id -- delete a VM snapshot from its tag or id
 singlestep [on|off] -- run emulation in singlestep mode or switch to normal mode
 stop -- stop emulation
Command: simconfig

- Switch to or configure MARRSS simulation mode

(qemu) simconfig
simulation options received:
syntax: simconfig <arguments...>
In the monitor mode give the above command with options given below

Options are:
PTLmon Control:
- help          Print this message [disabled]
Action [specify only one]:
- run           Run under simulation [disabled]
- stop          Stop current simulation run and wait for command [disabled]
- kill          Kill PTLsim inside domain (and ptlmon), then shutdown domain [disabled]
- flush         Flush all queued commands, stop the current simulation run and wait [disabled]
General Logging Control:
- quiet         Do not print PTLsim system information banner [disabled]
- logfile       Log filename (use /dev/fd/1 for stdout, /dev/fd/2 for stderr) [ptlsim.log]
- loglevel      Log level (0 to 99) [0]
- startlog      Start logging after iteration <startlog> [0]
- startlogrip   Start logging after first translation of basic block starting at rip [18446744073709551615]
- consolelog    Replicate log file messages to console [disabled]
- logbufsize    Size of PTLsim ptl_logfile buffer (not related to -ringbuf) [524288]
- logfilesize   Size of PTLsim ptl_logfile [67108864]
- dump-state-now Dump the event log ring buffer and internal state of the active core [disabled]
- screenshot    Takes screenshot of VM window at the end of simulation []
- log-user-only Only log the user mode activities [disabled]
- dump-config-file Dump Simulated Machine Configuration into Specified file instead of log file []
Simulated Machine Configuration

- In marss-0.4/config/default.conf

```python
shared_l2:
    description: Shared L2 Configuration
    min_contexts: 2
    cores: # The order in which core is defined is used to assign
           # the cores in a machine
    - type: ooo
      name_prefix: ooo_
    caches:
    - type: l1_128K_mesi
      name_prefix: L1_I_
      insts: $NUMCORES # Per core L1-I cache
      option:
        private: true
        last_private: true
    - type: l1_128K_mesi
      name_prefix: L1_D_
      insts: $NUMCORES # Per core L1-D cache
      option:
        private: true
        last_private: true
    - type: l2_2M
      name_prefix: L2_
      insts: 1 # Shared L2 config
```
Run BigDataBench

- Issue following commands in QEMU monitor console to configure MARSS mode simulation:
  - simconfig -logfile bench.log -stats bench.stats
    --machine $MACHINE_NAME

- Run BigDataBench
  - $ ./start-sim; ./runMicroBenchmark.sh; ./stop-sim

Switch to MARSS mode
Switch back to QEMU mode
E.g. Impala Workloads

Mysql has been started when the simulated OS was booting

Firewall has been stopped by default

Run Impala Workload

$ cd /home/linxinlong/BigDataBench_Impala_V3.0/InteractiveMicroBenchmark

$ ./start-sim; ./runMicroBenchmark.sh; ./stop-sim

We have done

Users need to do
Performance Data

- Performance characteristics file called `bench.stats` at `$MARSS_HOME`

```python
simulator:
  tags: [shared_12, localhost.localdomain.(none), 2014-12-09, kernel]
version:
  git_commit: 00000000
  git_branch: UNKNOWN
  git_timestamp: UNKNOWN
  build_timestamp: Nov 14 2014 14:39:45
  build_hostname: BUILDHOST
  build_compiler: gcc-4.4
run:
  timestamp: 1418121398
  hostname: localhost.localdomain.(none)
  native_hz: 2400000000
  seconds: 0
  performance:
    cycles_per_sec: 0
    commits_per_sec: 0
base_machine:
  oo5_0.0:
    cycles: 63364
    iq_reads: 29436
    iq_writes: 16574
    iq_fp_reads: 0
    iq_fp_writes: 0
    dispatch:
      width: [56600, 426, 419, 366, 3553]
      opclass:
        logic: 3240
        addsub: 4750
        addsubc: 0
        addshift: 227
```
Use case---Simics
Brief Introduction of Simics

- A full-system simulator used to run unchanged production binaries of the target hardware.
- It can simulate a wider range of ISA, e.g., SPARC, MIPS, x86 CPUs.
Simics version

- We provide images for SPARC and deploy Solaris operation system.
- Save checkpoint for each workload.
- Run workloads by:

<table>
<thead>
<tr>
<th>Workload</th>
<th>Master</th>
<th>Slaver</th>
</tr>
</thead>
</table>
| Wordcount | cd /master  
  ./simics -c Hadoopwordcount_L  
  bin/hadoop jar  
  ${HADOOP_HOME}/hadoop-examples-* .jar  
  wordcount /in /out/wordcount | cd /slaver  
  ./simics -c Hadoopwordcount_L |
Wrap-Up

- Using BigDataBench simulator version just two steps:
  - Deploy and run simulator
  - Run applications in the images

- For more information access "
  
  http://prof.ict.ac.cn/BigDataBench/simulatorversion"
Images of BigDataBench simulator version:
  - http://prof.ict.ac.cn/BigDataBench/simulatorversion/

We hope that our subsetting approach and resulting benchmark suite will facilitate architecture researchers in studying alternative organizations and technologies for big data systems.

If there is any problem just let us know.
  - E.g. It is too slow to download (we can mail the CDs).