Test tools deployment on Black Adder

March 15, 2010

Mark.Deng@Emerson.com

Lawrence.Hu@Emerson.com

Contents

1.CoreMark

1.1 Introduction

CoreMark, developed by EEMBC, is a simple, yet sophisticated, benchmark that is designed specifically to test the functionality of a processor core. Running CoreMark produces a single-number score allowing users to make quick comparisons between processors.

1.2 Build and run

```
# tar xzvf coremark_v1\[1\].0.tgz
# cd coremark_v1.0
# make
```

Full results are available in the files run1.log and run2.log. CoreMark result can be found in run1.log. The results may like this:

```
2K performance run parameters for coremark.
CoreMark Size       : 666
Total ticks         : 14174
Total time (secs): 14.174000
Iterations/Sec      : 2822.068576                    <-------result
Iterations          : 40000
Compiler version : GCC4.3.2
Compiler flags      : -O2 -DPERFORMANCE_RUN=1    -lrt
Memory location     : Please put data memory location here
                                (e.g. code in flash, data on heap etc)
seedcrc             : 0xe9f5
[0]crclist          : 0xe714
[0]crcmatrix        : 0x1fd7
[0]crcstate         : 0x8e3a
[0]crcfinal         : 0x25b5
Correct operation validated. See readme.txt for run and reporting rules.
CoreMark 1.0 : 2822.068576 / GCC4.3.2 -O2 -DPERFORMANCE_RUN=1    -lrt / Heap
```

2.Dhrystone

2.1 Introduction

Dhrystone is a synthetic computing benchmark program developed in 1984 by Reinhold P. Weicker intended to be representative of system (integer) programming. Dhrystone tries to represent the result more meaningfully than MIPS (million instructions per second), because exact instruction count comparisons between different instruction sets (e.g. RISC vs. CISC) are not meaningful. For example, the same high-level task may require many more instructions on a RISC machine, but might execute in the same wall clock time as a single CISC instruction. Thus, Dhrystone score counts only the number of program iteration completions per second, allowing individual machines to perform this calculation in a machine-specific way. Another common representation of the Dhrystone benchmark is the DMIPS (Dhrystone MIPS) obtained when the Dhrystone score is divided by 1,757 (the number of Dhrystones per second obtained on the VAX 11/780, nominally a 1 MIPS machine).

2.2 Build

```
# tar xzvf dhrystone-2.1.tar.tar
# cd Dhrystone-2.1
```

The source code should be modified to match the Embedded Linux Environment on P2020.

```
# vi dhry_1.c
```

In line 48, the “extern int times ()” should be changed to “extern clock_t times ()”. Edit and save it.

```
/* variables for time measurement: */


#ifdef TIMES
struct tms          time_info;
extern    clock_t       times ();
                        /* see library function "times" */
#define Too_Small_Time (2*HZ)
                        /* Measurements should last at least about 2 seconds */
#endif
```

```
# make
```

2.3 Run

```
# ./cc_dry2
# ./cc_dry2reg
```

```
# ./gcc_dry2
# ./gcc_dry2reg
```

3. Memtester

3.1 Introduction

Memtester is a user space utility for testing the memory subsystem for faults. It's portable and should compile and work on any 32- or 64-bit Unix-like system. (Even weird, proprietary Unices, and even Mac OS X.) For hardware developers, memtester can be told to test memory starting at a particular physical address.

3.2 Install

```
# tar xzvf memtester-4.1.3.tar.gz
# cd memtester-4.1.3
# make
```

If you want to install the binary and the manpage to /usr/local/, run

```
# make install
```

And edit INSTALLPATH in the makefile if you prefer a different location.

3.3 Run

```
# cd memtester-4.1.3
# ./memtester <mem> [loops]
```

<memory> is the amount of memory to test, in megabytes by default. You can optionally include a suffix of B, K, M, or G (for bytes, kilobytes, megabytes, and gigabytes respectively).

[loops] is an optional limit to the number of runs through all tests.

For example,

```
# ./memtester 512
```

4.IOzone

4.1 Introduction

IOzone is a filesystem benchmark tool. The benchmark generates and measures a variety of file operations. Iozone has been ported to many machines and runs under many operating systems.

IOzone is useful for performing a broad filesystem analysis of a vendor’s computer platform. The benchmark tests file I/O performance for the following operations: Read, write,

re-read, re-write, read backwards, read strided, fread, fwrite, random read, pread ,mmap, aio_read, aio_write.

4.2 Build

```
# tar xzvf iozone3_338.tar

# cd iozone3_338/src/current

# make linux-powerpc
```

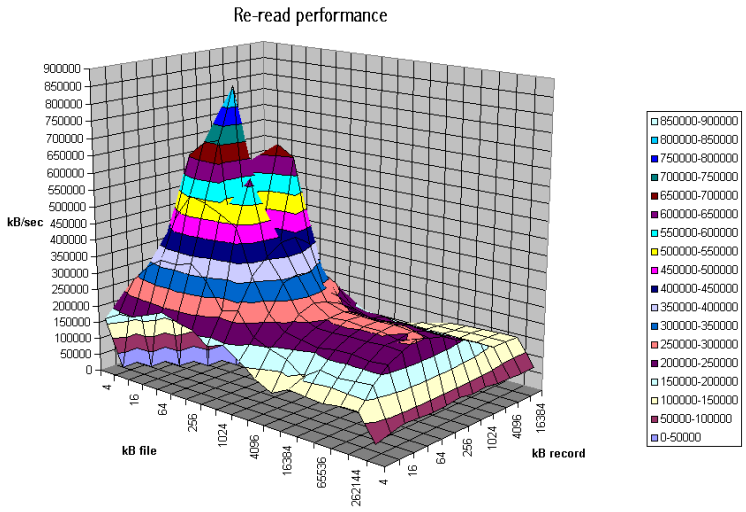
4.3 Run

The simplest way to get started is to try the automatic mode. Output file “output.wks” is a binary format spreadsheet.

```
# ./iozone –Rab results.wks
```

Run “./iozone -h” for more help message, or refer to the guide in /iozone3_338/docs.

Check the results in file results.wks. The results can be imported to the Excel and generate the statistical graphics. The figure below is a graphic generated from IOzone output files.



5.Bonnie++

5.1 Introduction

Bonnie++ is a benchmark suite that is aimed at performing a number of simple tests of hard drive and file system performance. Then you can decide which test is important and decide how to compare different systems after running it. The main program tests database type access to a single file (or a set of files if you wish to test more than 1G of storage), and it tests creation, reading, and deleting of small files which can simulate the usage of programs such as Squid, INN, or Maildir format email.

5.2 Install

```
# tar xzvf bonnie++-1.03e.gz
# cd bonnie++-1.03e
# ./configure
# make
# make install
```

5.3 Run

Run bonnie++ to get help message:

```
# bonnie++
```

```
usage: bonnie++ [-d scratch-dir] [-s size(MiB)[:chunk-size(b)]]
                [-n number-to-stat[:max-size[:min-size][:num-directories]]]
                [-m machine-name]
                [-r ram-size-in-MiB]
                [-x number-of-tests] [-u uid-to-use:gid-to-use] [-g gid-to-use]
                [-q] [-f] [-b] [-D] [-p processes | -y]
```

Test command sample:

```
# bonnie++ -d /tmp –m P2020 –u root
```

The results may like this,

```
Version 1.93c       ------Sequential Output------ --Sequential Input- --Random-
Concurrency   1     -Per Chr- --Block-- -Rewrite- -Per Chr- --Block-- --Seeks--
Machine        Size K/sec %CP K/sec %CP K/sec %CP K/sec %CP K/sec %CP  /sec %CP
P2020          256M   217  86  2220   0  2489   1   656  99 +++++ +++ 974.1   9
Latency             45796us    3022ms    3824ms   13786us     683us     377ms
Version 1.93c       ------Sequential Create------ --------Random Create--------
P2020               -Create-- --Read--- -Delete-- -Create-- --Read--- -Delete--
                files  /sec %CP  /sec %CP  /sec %CP  /sec %CP  /sec %CP  /sec %CP
                   16    92   1   354   2    95   1    83   1   260   1    80   1
Latency             3315ms    1836ms    1638ms    3357ms    1621ms    3425ms
1.93c,1.93c,P2020,1,1268736293,256M,,217,86,2220,0,2489,1,656,99,+++++,+++,974.1
,9,16,,,,,92,1,354,2,95,1,83,1,260,1,80,1,45796us,3022ms,3824ms,13786us,683us,37
7ms,3315ms,1836ms,1638ms,3357ms,1621ms,3425ms
```

6.Netperf

6.1 Introduction

Netperf is a benchmark that can be used to measure various aspect of networking performance. The primary foci are bulk (aka unidirectional) data transfer and request/response performance using either TCP or UDP and the Berkeley Sockets interface. The tests available include:

- TCP and UDP unidirectional transfer and request/response over IPv4 and IPv6 using the Sockets interface.
- TCP and UDP unidirectional transfer and request/response over IPv4 using the XTI interface.
- Link-level unidirectional transfer and request/response using the DLPI interface.
- Unix domain sockets
- SCTP unidirectional transfer and request/response over IPv4 and IPv6 using the sockets interface.

6.2 Install

```
# tar xzvf netperf-2.3.tar.gz
# cd netperf-2.3
# vi makefile
```

Comment out the line 114

```
…
LOG_FILE=DEBUG_LOG_FILE="\"/tmp/netperf.debug\""
#CFLAGS = -O -D$(LOG_FILE) -DNEED_MAKEFILE_EDIT
…
```

```
# make
# make install
```

6.3 Run

Netperf is designed around a basic client-server model. There are two executables - netperf and netserver. Generally you will only execute the netperf program, with the netserver program being invoked by the remote system's inetd or equivalent.

Server:

```
# netserver
```

Starting netserver at port 12865

Starting netserver at hostname 0.0.0.0 port 12865 and family AF_UNSPEC

Clent:

netperf

TCP STREAM TEST to localhost

Recv Socket Size bytes	Send Socket Size bytes	Send Message Size bytes	Elapsed Time secs.	Throughput 10^6bits/sec
256	16384	16384	10.01	75.06

7. Lmbench

7.1 Introduction

lmbench is a suite of simple, portable, ANSI/C microbenchmarks for UNIX/POSIX. In general, it measures two key features: latency and bandwidth. lmbench is intended to give system developers insight into basic costs of key operations.

Lmbench tests,

- Bandwidth benchmarks
  - Cached file read
  - Memory copy (bcopy)
  - Memory read
  - Memory write
  - Pipe
  - TCP
- Latency benchmarks
  - Context switching.
  - Networking: connection establishment, pipe, TCP, UDP, and RPC hot potato
  - File system creates and deletes.
  - Process creation.
  - Signal handling
  - System call overhead
  - Memory read latency
- Miscellanious
  - Processor clock rate calculation

7.2 Install and run

tar xzvf lmbench-3.0-a9.gz

```
# cd lmbench-3.0-a9
# make results see                    //build and run the benchmark
```

After selecting some configuration options, have a  while it goes to work. The benchmark takes about a half hour to run. The results will be in **Results/ os_name/host_name**. It's a good idea to do several runs and compare the output like so

```
# make results
# make rerun
# make rerun
# make rerun
```

8.LTP

8.1 Introduction

The Linux™ Test Project (LTP) is a joint project started by SGI™ and maintained by IBM®, that has a goal to deliver test suites to the open source community that validate the reliability, robustness, and stability of Linux. The LTP test suite contains a collection of automated and semi-automated tools for testing the Linux kernel and related features. The goal of LTP is to deliver a suite of automated testing tools for Linux as well as publishing the results of tests we run.

8.2 Install

Install info-zip:

```
# tar xzvf zip30.tar.gz
# cd zip30
# make -f unix/Makefile generic_gcc
# make -f unix/Makefile install
```

Install LTP:

```
# tar xzvf ltp-full-20100228.gz
# cd ltp-full-20100228
# tar xzf ltp-XXXXXXXX.tar.gz
# cd ltp
# ./configure
# make all
```

make install

8.3 Run

cd /opt/ltp/runltp

./runltp -p -l p2020.log

The command executes the default set of test case. The results will be logged in p2020.log. The running process may take several hours, thus you'd better run it in midnight. The results log file is as below. For more usage, please refer to http://ltp.sourceforge.net/documentation/how-to/ltp.php

```
Test Start Time: Wed Mar 17 10:19:14 2010
----------------------------------------
Testcase                  Result     Exit Value
--------                  ------     ----------
abort01                   PASS       0
accept01                  PASS       0
accept4_01                PASS       0
access01                  PASS       0
access02                  PASS       0
access03                  PASS       0
access04                  PASS       0
access05                  PASS       0
acct01                    PASS       0
acct02                    PASS       0
…
…
pty01                     FAIL       2
ptem01                    PASS       0
hangup01                  PASS       0
Containers                PASS       0
BindMounts                FAIL       1

------------------------------------------------
Total Tests: 1226
Total Failures: 57
Kernel Version: 2.6.32-rc3
Machine Architecture: ppc
Hostname: P2020RDB
```

9.Summary

The summary of the test tools on P2020 is as below. Other versions and more information could be found via the URL.

Tool name	Version	Cover	URL
CoreMark	coremark_v1[1].0	CPU	http://www.coremark.org/home.php
Dhrystone	dhrystone-2.1	CPU	http://netsw.org/benchmark/system/
Memtester 4	Memtester 4	Memory	http://pyropus.ca/software/memtester/
IOZone	iozone3_338	Storage	http://www.iozone.org/
Bonnie++	bonnie++-1.03e	Storage	http://sourceforge.net/projects/bonnie/
Netperf	netperf-2.3	Network	http://www.netperf.org/netperf/
Lmbench	lmbench-3.0-a9	System	http://lmbench.sourceforge.net/
LTP	ltp-full-20030206	System	http://ltp.sourceforge.net/

Notes,

All of the software introduced in this guide could be downloaded at \\10.163.40.195\cncdebaodom02\ECC\Lawrence\P2020.