Hybrid QEMU
Technical Report

Virtualization Development Team

November 4, 2011

Contents

1 Introduction

N/A

A Installing and running Hybrid QEMU for x86 host

Follow the information at http://not-yet-available/hqemu/document.html. More detailed instruction for building and installing Hybrid QEMU are provided below.

Notations
$QEMU: top directory of the Hybrid QEMU source tree

A.1 Installing dependent packages

The following software packages are currently required by Hybrid QEMU:

- LLVM 2.8 or newer
- LLVM-GCC 4.2 or newer
- Pthreads
- GCC 3.x or newer (GCC 4.x recommended)
- libpfmon 2.0 or newer (optional)

You can download prebuilt x86 binaries of LLVM and LLVM-GCC from the official LLVM website. For other architectures, you may need to build them from source if prebuilt binary is not available. Because patching LLVM[1] is required to work with Hybrid QEMU, you **MUST** build LLVM from source.

The following example assumes that you will be building LLVM and LLVM-GCC from source and installing them in the /usr/local directory. x86 machine is used in this example; the configure arguments to other architectures may be different.

A.1.1 LLVM

```
$ tar -zxf llvm-2.8.tgz
$ cd llvm-2.8
$ patch -p1 < llvm-2.8.patch
$ mkdir build
$ cd build
$ ../configure --prefix=/usr/local/llvm-2.8 --enable-optimized
$ make
$ make install
$ export PATH=/usr/local/llvm-2.8/bin:$PATH
```

NOTE: llvm-*${VERSION}*.patch is placed in the $QEMU/patch directory.

A.1.2 LLVM-GCC

```
$ tar -zxf llvm-gcc-4.2-2.8.source.tgz
$ mkdir llvm-gcc-4.2-2.8.source/build
$ cd llvm-gcc-4.2-2.8.source/build
$ ../configure --prefix=/usr/local/llvm-gcc --program-prefix=llvm- \
    --enable-llvm=/usr/local/llvm-2.8 --enable-languages=c,c++,fortran
$ make
$ make install
$ export PATH=/usr/local/llvm-gcc/bin:$PATH
```

[1]patch to reserve registers required by QEMU

A.1.3 libpfmon (Optional)

For installation guides, please see http://perfmon2.sourceforge.net/.

A.2 Building and installing Hybrid QEMU

The building of Hybrid QEMU requires LLVM tool `llvm-config` to locate LLVM header files and libraries, and LLVM-GCC tools to generate required LLVM bitcode file. Thus, make sure that paths of these tools have been added to your PATH environment variable (see section A.1.1 and A.1.2) before the building process. The default steps for building and installing Hybrid QEMU are as follows:

```
$ tar -zxf qemu-hybrid.tar.gz
$ mkdir qemu
$ cd qemu
$ ../qemu-hybrid/configure --prefix=`pwd` --target-list=i386-linux-user \
    --enable-llvm=hybridm
$ make
$ make install
```

Here are some configure arguments which may be of interest:

- --prefix=<path>: install all files in the specified directory (/usr/local/ is the default if –prefix is not specified)
- --target-list=<list>: set target lists separated by comma (e.g. –target-list=i386-linux-user,arm-linux-user)
- --enable-llvm=<mode>: enable LLVM support. <mode> can be *llvm*[1], *hybrids* or *hybridm*[2] (original QEMU will be used if –enable-llvm is not specified)
- --enable-dbo: enable dynamic binary optimization (libpfmon required)
- --enable-debug: enable common debug build options

A.3 Running Hybrid QEMU

You can download user mode test programs 'linux-user-test-0.3.tar.gz' from the QEMU official website http://wiki.qemu.org/Download. This tarball consists of Linux user mode emulation test programs and their associated runtime libraries for different architectures.

```
$ cd qemu
$ tar -zxf linux-user-test-0.3.tar.gz
$ ./bin/qemu-i386 -L./linux-user-test-0.3/gnemul/qemu-i386/ \
    ./linux-user-test-0.3/i386/sha1

SHA1=15dd99a1991e0b3826fede3deffc1feba42278e6
```

If error message 'Error: Bitcode file <path>/llvm_helper.bc: No such file or directory' shows up, you need to manually create directory <path> and move bitcode file llvm_helper.bc from your Hybrid QEMU installation path to directory <path>.

[1] llvm denotes using LLVM translator only

[2] hybrids denotes both TCG and LLVM translators are run by one single thread; hybridm uses both translators run by different threads

A.4 Debugging Hybrid QEMU

Developers who wish to find more verbose information about how the LLVM translator is working can use the debug masks to select different debugging level. The debugging level can be gathered by setting LLVM_DEBUG environment variable before running a program. The default debugging level is set to *none*. You can turn on multiple debugging levels at the same time with desired debug mask values separated by comma. The available debug mask values are listed below:

- *none*: no debug message will be displayed. This is the default debugging level
- *llvm*: status of the LLVM translator
- *in_asm*: display assembly codes of guest program
- *op*: display TCG IRs of guest program
- *out_asm*: display generated host assembly codes
- *ir*: LLVM IRs
- *entry*: translation functions that are called
- *dbo*: display information generated by dynamic binary optimizer
- *verify*: check if an LLVM function is well-formed. This verification will not cause the execution to terminate, only warning messages are displayed
- *asm*: turn on debugging levels of *in_asm*, *op* and *out_asm*
- *debug*: turn on debugging levels of *llvm*, *ir* and *out_asm*
- *all*: turn on all debugging levels

NOTE: debug messages are directed to 'stderr' and will be displayed on the screen.

The example below shows parts of the debug messages with debug mask values, *llvm* and *ir*, being set:

```
$ export LLVM_DEBUG=llvm,ir
$ ./bin/qemu-i386 -L./linux-user-test-0.3/gnemul/qemu-i386/ \
    ./linux-user-test-0.3/i386/sha1

[00:00:00.004389] LLVM Event Listener registered.
[00:00:00.015953] LLVM Environment initialized.
[00:00:00.016568] LLVM IR Factory initialized.
[00:00:00.016578] LLVM Translator initialized.
[00:00:00.016751] TraceInit: pc 0x40802a46 length 1 is_loop 1
[00:00:00.016768] TraceSetContext entered pc 40802a46 (head,end).
[00:00:00.016836] Compile: start...

define void @"40802a46"() noreturn nounwind naked {
loop:
  %r14 = call i8* asm sideeffect "", "={r14}"() nounwind
  %0 = bitcast i8* %r14 to %struct.CPUX86State*
  %cc_dst = getelementptr i8* %r14, i32 44
  %cc_src = getelementptr i8* %r14, i32 40
  %cc_op = getelementptr i8* %r14, i32 48
  %eax = getelementptr i8* %r14, i32 0
  %edx = getelementptr i8* %r14, i32 8
  br label %entry

entry:                                           ; preds = %loop
  %1 = bitcast i8* %edx to i32*
  %2 = load i32* %1, !edx !0
  %3 = bitcast i8* %eax to i32*
  %4 = load i32* %3, !eax !1
  %5 = shl i32 %4, 2
  %6 = add i32 %2, %5
  %7 = inttoptr i32 %6 to i32*
  %8 = getelementptr i32* %7, i32 0
  store i32 0, i32* %8
  %9 = add i32 %4, 1
  %10 = sub i32 %9, 97
  %11 = add i32 %10, 97
  %12 = bitcast i8* %cc_op to i32*
  %13 = bitcast i8* %cc_src to i32*
  %14 = bitcast i8* %cc_dst to i32*
  %15 = bitcast i8* %eax to i32*
  %16 = icmp ule i32 %11, 97
  br i1 %16, label %true_dest, label %false_dest
  (......)
```

A.5 Profiling Hybrid QEMU

Users who wish to profile a program's execution behavior can use the LLVM_PROFILE environment variable to control the profiling levels of a target program. Currently, the profiling method is instrumentation-based and might slow down the execution of a program significantly. The default profiling level is set to *none*. To turn profiling on, set LLVM_PROFILE environment variable before running a program. You can turn on multiple profiling levels at the same time with desired values separated by comma. The available profiling mask values are listed below: (the brace denotes the mode that supports this profiling level)

- *none*: no profiling message will be displayed. This is the default profiling level
- *basic*: static information. This level includes the basic blocks' amount, guest and host assembly code size, guest instruction count and translation time; traces' amount, thresholds of trace generation, expected length, the number of trace exits, the number of indirect branch in a trace, guest and host assembly code size, translation time (llvm, hybrid)
- *edge*: edge profiling. This level includes guest program's control flow graph and edge counts (llvm)
- *trace*: trace's runtime information. This level includes the values of trace counters (hybrid)
- *cache*: addresses/sizes of basic block cache and trace caches (llvm, hybrid)
- *all*: turn on all profiling levels

NOTE: profiling messages are directed to 'stderr' and will be displayed on the screen.

The following example demonstrates the profiling results of program *sha1*.

```
$ export LLVM_PROFILE=basic,trace
$ ./bin/qemu-i386 -L./linux-user-test-0.3/gnemul/qemu-i386/ \
    ./linux-user-test-0.3/i386/sha1

SHA1=15dd99a1991e0b3826fede3deffc1feba42278e6
```

```
-------------------------
Block statistic:
Num of Blocks  : 1856
G/H Code size  : 40378/422012 bytes
Guest iCount   : 11435
Trans. Cycles  : 179473023 cycles

Trace statistic:
Num of Traces  : 32
Profile Thres. : 50
Predict Thres. : 16
G/H Code size  : 7499/27626 bytes
Trans. Cycles  : 605679246 cycles
Average Len    : 4.1 (max=16)
Average # Exit : 3.5 (max=12)
Average # IB   : 0.4 (max=2)
TB covered     : 118/131 (Duplicate 13)
Guest Size     : 7269/7499 (Duplicate 230) bytes
Guest iCount   : 2143/2203 (Duplicate 60)
Length distribution: (1-16)
    11 7 4 4 2 0 0 0 0 0 1 1 0 1 1 1
-------------------------
Trace information:
 Id (Len) PC of TBs
  0 ( 4): 0x4114ecc9 0x4114ed4c 0x4114ed59 0x4114ed5d
  1 ( 1): 0x4115bc47
    (......)
 23 ( 1): 0x8049c90
 25 (15): 0x80490df 0x80491cd 0x8049288 0x8049356 0x8049417 0x80494e9 ...
    (......)
-------------------------
Thread 0:
Trace used: 17/32
  Id (C:Ex)  LoopCnt/   Count: ( Len) Accumulated Exit Cnt
   2 (1: 4)        0/      12: ( 1.0) 12 0 0 0 0
   3 (1: 3)       31/      35: (16.5) 4 0
   4 (1: 2)        9/      11: ( 5.5) 2
   5 (0: 2)        0/       8: ( 1.0) 8
    (......)
  22 (0: 6)        0/       4: ( 3.0) 2 0 0 0 2
  24 (0: 1)        0/   35832: (16.0) 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 35832
  25 (0: 3)        0/   22661: (15.0) 0 0 0 0 0 0 0 0 0 0 0 0 355 355 21951
  26 (1: 2)     1736/    2013: ( 7.3) 277
  27 (0: 4)        0/      93: ( 2.4) 0 56 37
```

B Cross-compiling Hybrid QEMU for ARM host

B.1 Installing dependent packages

The following software packages are currently required by Hybrid QEMU:

- LLVM 2.8 or newer
- LLVM-GCC 4.2 or newer
- Pthreads
- GCC toolchain for ARM 3.x or newer (GCC 4.x recommended)

The following example assumes that you will be building LLVM and LLVM-GCC from source and installing them in the /usr/local directory. X86 machine is used as the platform for cross-compilation.

B.1.1 ARM GCC toolchain

Prebuilt ARM GCC toolchain can be downloaded from http://www.codesourcery.com/sgpp/lite/arm.
Put the toolchain in the /usr/arm directory.

B.1.2 LLVM for x86

```
$ tar -zxf llvm-2.8.tgz
$ cd llvm-2.8
$ mkdir build
$ cd build
$ ../configure --prefix=/usr/local/llvm-2.8-x86 --enable-optimized
$ make
$ make install
$ export PATH=/usr/local/llvm-2.8-x86/bin:$PATH
```

B.1.3 LLVM-GCC Cross-compiler for ARM

```
$ tar -zxf llvm-gcc-4.2-2.8.source.tgz
$ mkdir llvm-gcc-4.2-2.8.source/build
$ cd llvm-gcc-4.2-2.8.source/build
$ ../configure --prefix=/usr/local/llvm-gcc-arm --program-prefix=llvm- \
    --enable-llvm=/usr/local/llvm-2.8-x86 --enable-languages=c,c++ \
    --disable-multilib \
    --target=arm-none-linux-gnueabi \
    --with-sysroot=/usr/arm/arm-none-linux-gnueabi/libc/
$ make
$ make install
$ export PATH=/usr/local/llvm-gcc-arm/bin:$PATH
```

NOTE: Since LLVM-GCC will be executed on x86 machine, you need to specify the path of x86 version LLVM whiling configuring LLVM-GCC.

B.1.4 LLVM for ARM

```
$ tar -zxf llvm-2.8.tgz
$ cd llvm-2.8
$ patch -p1 < llvm-2.8.patch
$ mkdir build
$ cd build
$ ../configure --prefix=/usr/local/llvm-2.8-arm --enable-optimized \
    --build=i686-pc-linux-gnu \
    --host=arm-none-linux-gnueabi \
    --target=arm-none-linux-gnueabi
$ make
$ make -i install        # add -i to ignore installation errors
$ export PATH=/usr/local/llvm-2.8-arm/bin:$PATH
```

NOTE: llvm-*${VERSION}*.patch is placed in the $QEMU/patch directory.
NOTE: Add this ARM version LLVM to $PATH so that it can be linked by the Hybrid QEMU correctly.

B.2 Building and installing Hybrid QEMU

```
$ tar -zxf qemu-hybrid.tar.gz
$ mkdir qemu
$ cd qemu
$ ../qemu-hybrid/configure --prefix=`pwd` --target-list=arm-linux-user \
    --enable-llvm=llvm \
    --cross-prefix=arm-none-linux-gnueabi- \
    --cpu=armv7l \
    --disable-strip
$ make
$ make install
```

B.3 Running Hybrid QEMU

You can download user mode test programs ‘linux-user-test-0.3.tar.gz’ from the QEMU official website http://wiki.qemu.org/Download. This tarball consists of Linux user mode emulation test programs and their associated runtime libraries for different architectures.

```
$ cd qemu
$ tar -zxf linux-user-test-0.3.tar.gz
$ ./bin/qemu-arm -L./linux-user-test-0.3/gnemul/qemu-arm/ \
    ./linux-user-test-0.3/arm/hostname
```