HQEMU v2.5.2
Technical Report

Virtualization Development Team

September 26, 2018

Contents

1 Introduction

HQEMU (Hybrid QEMU) is a cross-ISA, retargetable and multi-threaded dynamic binary translator on multicores. HQEMU integrates QEMU and LLVM as its building blocks, and supports process-level emulation and full-system virtualization. Figure 1 shows the organization of HQEMU. For more design details, please refer to the papers: (a) http://dl.acm.org/citation.cfm?id=2259016.2259030, and (b) https://ieeexplore.ieee.org/abstract/document/6471968/.

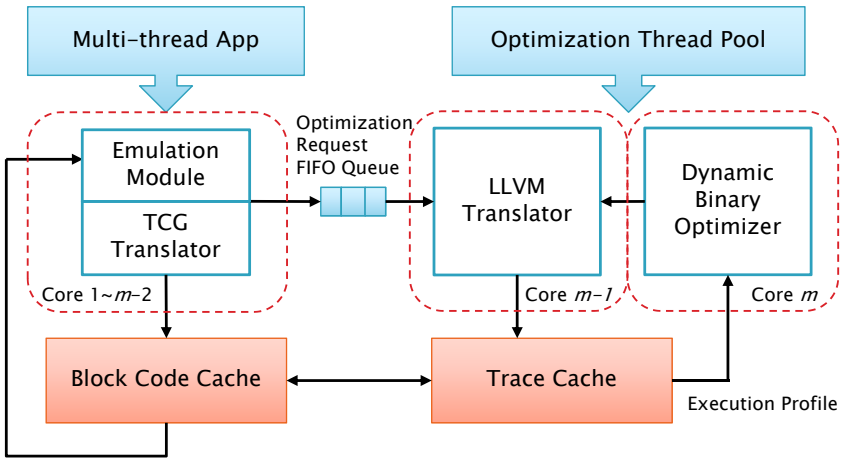

Figure 1: Major components of HQEMU on an m-core platform.

1.1 ISAs supported in HQEMU v2.5.2

Guest ISA: x86, x86-64, arm, aarch64.
Host ISA: x86, x86-64, aarch64, ppc64le.

1.2 Requirements

- Linux with recent kernel version. We use 2.6.30 and later, but earlier versions should also work.
- LLVM version 3.5 and later. Earlier versions are not supported.

HQEMU v2.5.2 is based on QEMU version 2.5. Table 1 lists the LLVM versions that are successfully integrated for the supported host ISAs. Please select the correct LLVM version when building HQEMU. HQEMU supports both LLVM MCJIT and the legacy JIT. MCJIT is supported with all LLVM versions, and the legacy JIT is supported with LLVM v3.5 only.

Table 1: Supported LLVM versions for the host ISA.

Host ISA	**LLVM version**				
	3.5	3.8	3.9	5.0	6.0
x86/x86-64	Yes	Yes	Yes	Yes	Yes
aarch64			Yes	Yes	Yes
ppc64le				Yes	Yes

2 Installing HQEMU on the x86 host

In the following instructions, we use $HQEMU to denote the top directory of the HQEMU source tree.

2.1 Install dependent packages

The following software packages are required by HQEMU:

- LLVM 3.5 or newer
- Clang 3.5 or newer
- GCC 4.8 or newer

You can use prebuilt x86 Clang provided on the official LLVM website. Because patching LLVM[1] is required to work with HQEMU, you **MUST** build LLVM from source. The following example assumes that you will build LLVM 6.0 and Clang 6.0 from source, and install them in the /usr/local directory. x86-64 host machine is used in this example.

2.1.1 Build LLVM and Clang

```
$ ls
cfe-6.0.0.src.tar.xz  compiler-rt-6.0.0.src.tar.xz
llvm-6.0.0.src.tar.xz

$ tar -xf llvm-6.0.0.src.tar.xz
$ cd llvm-6.0.0.src
$ patch -p1 < $HQEMU/patch/llvm/llvm-6.0.patch
$ mkdir build
$ cd tools
$ tar -xf ../../cfe-6.0.0.src.tar.xz
$ mv cfe-6.0.0.src clang
$ cd ../projects
$ tar -xf ../../compiler-rt-6.0.0.src.tar.xz
$ mv compiler-rt-6.0.0.src compiler-rt
$ cd ../build

$ cmake -G "Unix Makefiles" \
        -DCMAKE_INSTALL_PREFIX=/usr/local/llvm-6.0 \
        -DCMAKE_BUILD_TYPE=Release ..

$ make && make install
$ export PATH=/usr/local/llvm-6.0/bin:$PATH
```

Note that the patch file "llvm-VERSION.patch" is placed in the $HQEMU/patch/llvm directory.

[1] Patch to reserve registers required by QEMU.

2.2 Build HQEMU

The building of HQEMU requires LLVM tool *llvm-config* to locate LLVM header files and libraries, and *clang* to generate LLVM bitcode file. Therefore, make sure the LLVM installation path has been added to your PATH environment variable. The default steps for building HQEMU are as follows:

```
$ tar -zxf hqemu-2.5.2.tar.gz
$ mkdir build
$ cd build
$ ../hqemu-2.5.2/configure --prefix=`pwd` \
                           --target-list=i386-linux-user \
                           --enable-llvm
$ make && make install
```

Here are some configuration options which may be of interest:

- --prefix=<path>: install all files in the specified directory. /usr/local/ is the default path if --prefix is not specified.
- --target-list=<list>: set target ISA lists. Use comma to separate targets if multiple targets are specified, for example, --target-list=i386-linux-user,arm-linux-user for process-level emulation; --target-list=i386-softmmu,arm-softmmu for full-system emulation.

2.3 Run HQEMU

You can download user mode test programs "linux-user-test-0.3.tar.gz" from http://itanium.iis.sinica.edu.tw/hqemu/download/linux-user-test-0.3.tar.gz. This tarball consists of test programs and runtime libraries for Linux user-mode emulation. Additional test programs can be found from the QEMU official website http://wiki.qemu.org/Testing.

```
$ cd build
$ 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
```

Users can use the environment variable, LLVM_MODE, to control HQEMU's execution mode, e.g.,

```
$ export LLVM_MODE=<mode>
```

The available modes are:

- `none`: use the QEMU/TCG translator only. No LLVM optimization is used.
- `block`: use the LLVM translator only, with block-level translation.
- `hybrids`: HQEMU running emulation and LLVM optimizations on one single thread.
- `hybridm`: HQEMU running emulation and LLVM optimizations on different threads. (the default mode, same as LLVM_MODE unset)

3 Installing HQEMU on the ARM64 host

3.1 Install dependent packages

The following software packages are required by HQEMU:

- LLVM 3.9 or newer
- Clang 3.9 or newer
- GCC 4.8 or newer

The following example assumes that you will build LLVM 6.0 and Clang 6.0 from source, and install them in the /usr/local directory. $HQEMU denotes the top directory of the HQEMU source tree. ARM64 host machine is used in this example.

3.1.1 Build LLVM and Clang

```
$ ls
cfe-6.0.0.src.tar.xz  compiler-rt-6.0.0.src.tar.xz
llvm-6.0.0.src.tar.xz

$ tar -xf llvm-6.0.0.src.tar.xz
$ cd llvm-6.0.0.src
$ patch -p1 < $HQEMU/patch/llvm/llvm-6.0.patch
$ mkdir build

$ cd tools
$ tar -xf ../../cfe-6.0.0.src.tar.xz
$ mv cfe-6.0.0.src clang
$ cd ../projects
$ tar -xf ../../compiler-rt-6.0.0.src.tar.xz
$ mv compiler-rt-6.0.0.src compiler-rt
$ cd ../build

$ cmake -G "Unix Makefiles" \
        -DCMAKE_INSTALL_PREFIX=/usr/local/llvm-6.0 \
        -DCMAKE_BUILD_TYPE=Release ..

$ make && make install
$ export PATH=/usr/local/llvm-6.0/bin:$PATH
```

Note that the patch file “llvm-VERSION.patch” is placed in the $HQEMU/patch/llvm directory.

3.2 Build HQEMU

The building of HQEMU requires LLVM tool *llvm-config* to locate LLVM header files and libraries, and *clang* to generate LLVM bitcode file. Therefore, make sure the LLVM installation path has been added to your PATH environment variable. The default steps for building HQEMU are as follows:

```
$ tar -zxf hqemu-2.5.2.tar.gz
$ mkdir build
$ cd build
$ ../hqemu-2.5.2/configure --prefix=`pwd` \
                           --target-list=i386-linux-user \
                           --enable-llvm
$ make && make install
```

Here are some configuration options which may be of interest:

- --prefix=<path>: install all files in the specified directory. /usr/local/ is the default path if --prefix is not specified.
- --target-list=<list>: set target ISA lists. Use comma to separate targets if multiple targets are specified, for example, --target-list=i386-linux-user,arm-linux-user.

3.3 Run HQEMU

You can download user mode test programs "linux-user-test-0.3.tar.gz" from http://itanium.iis.sinica.edu.tw/hqemu/download/linux-user-test-0.3.tar.gz. This tarball consists of test programs and runtime libraries for Linux user-mode emulation.

```
$ cd build
$ 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
```

Users can use the environment variable, LLVM_MODE, to control HQEMU's execution mode, e.g.,

```
$ export LLVM_MODE=<mode>
```

The available modes are:

- `none`: use the QEMU/TCG translator only. No LLVM optimization is used.
- `block`: use the LLVM translator only, with block-level translation.
- `hybrids`: HQEMU running emulation and LLVM optimizations on one single thread.
- `hybridm`: HQEMU running emulation and LLVM optimizations on different threads. (the default mode, same as LLVM_MODE unset)

4 Cross-compiling HQEMU for the ARM64 host

4.1 Install dependent packages

The following software packages are required by HQEMU:

- LLVM 3.9 or newer
- Clang 3.9 or newer
- x86-64 GCC 4.8 or newer
- ARM64 GCC toolchain 4.8 or newer (Linaro Toolchain aarch64-linux-gnu is recommended)

The following example assumes that you will build LLVM 6.0 and Clang 6.0 from source, and install them in the /usr/local directory. $HQEMU denotes the top directory of the HQEMU source tree. x86-64 host machine is used as the platform for cross-compilation. Assume the Linaro ARM64 Toolchain is installed in /opt/gcc-linaro-aarch64/.

4.1.1 Build the Clang cross-compiler for ARM64

```
$ ls
cfe-6.0.0.src.tar.xz  compiler-rt-6.0.0.src.tar.xz
llvm-6.0.0.src.tar.xz

$ tar -xf llvm-6.0.0.src.tar.xz
$ cd llvm-6.0.0.src
$ mkdir build

$ cd tools
$ tar -xf ../../cfe-6.0.0.src.tar.xz
$ mv cfe-6.0.0.src clang
$ cd ../projects
$ tar -xf ../../compiler-rt-6.0.0.src.tar.xz
$ mv compiler-rt-6.0.0.src compiler-rt
$ cd ../build

$ cmake -G "Unix Makefiles" \
        -DCMAKE_INSTALL_PREFIX=/usr/local/llvm-6.0 \
        -DCMAKE_BUILD_TYPE=Release ..

$ make && make install
$ cp ./bin/clang-tblgen /usr/local/llvm-6.0/bin/
```

4.1.2 Build the ARM64 LLVM library

```
$ tar -xf llvm-6.0.0.src.tar.xz
$ cd llvm-6.0.0.src
$ patch -p1 < $HQEMU/patch/llvm/llvm-6.0.patch
$ mkdir build-aarch64
$ cd build-aarch64

$ cmake -G "Unix Makefiles" \
        -DCMAKE_INSTALL_PREFIX=/usr/local/llvm-6.0-aarch64 \
        -DCMAKE_BUILD_TYPE=Release \
        -DCMAKE_CROSSCOMPILING=True \
        -DLLVM_TABLEGEN=/usr/local/llvm-6.0/bin/llvm-tblgen \
        -DCLANG_TABLEGEN=/usr/local/llvm-6.0/bin/clang-tblgen \
        -DLLVM_DEFAULT_TARGET_TRIPLE=aarch64-linux-gnu \
        -DCMAKE_C_COMPILER=aarch64-linux-gnu-gcc \
        -DCMAKE_CXX_COMPILER=aarch64-linux-gnu-g++ \
        ..

$ make && make install
$ rm -f /usr/local/llvm-6.0-aarch64/bin/*
$ cp -r /usr/local/llvm-6.0/bin/* /usr/local/llvm-6.0-aarch64/bin/
$ cp -r /usr/local/llvm-6.0/lib/clang /usr/local/llvm-6.0-aarch64/lib/
$ export PATH=/usr/local/llvm-6.0-aarch64/bin:$PATH
```

Note that the patch file "llvm-VERSION.patch" is placed in the $HQEMU/patch/llvm directory.

4.2 Build HQEMU

The building of HQEMU requires LLVM tool *llvm-config* to locate LLVM header files and libraries, and *clang* to generate LLVM bitcode file. Therefore, make sure the LLVM installation path has been added to your PATH environment variable. The default steps for building HQEMU are as follows:

```
$ export SYSROOT=/opt/gcc-linaro-aarch64/aarch64-linux-gnu/libc/

$ tar -zxf hqemu-2.5.2.tar.gz
$ mkdir build
$ cd build
$ ../hqemu/configure --prefix=`pwd` \
                     --target-list=i386-linux-user \
                     --enable-llvm \
                     --cross-prefix=aarch64-linux-gnu- \
                  --clang-flags="-target aarch64 --sysroot $SYSROOT"
$ make && make install
```

Here are some configuration options which may be of interest:

- --prefix=<path>: install all files in the specified directory. /usr/local/ is the default path if --prefix is not specified.
- --target-list=<list>: set target ISA lists. Use comma to separate targets if multiple targets are specified, for example, --target-list=i386-linux-user,arm-linux-user.

4.3 Run HQEMU

You can download user mode test programs "linux-user-test-0.3.tar.gz" from http://itanium.iis.sinica.edu.tw/hqemu/download/linux-user-test-0.3.tar.gz. This tarball consists of test programs and runtime libraries for Linux user-mode emulation.

```
Copy the installation directory 'build/bin' to the ARM64 machine.

$ cd build
$ 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
```

Users can use the environment variable, LLVM_MODE, to control HQEMU's execution mode, e.g.,

```
$ export LLVM_MODE=<mode>
```

The available modes are:

- `none`: use the QEMU/TCG translator only. No LLVM optimization is used.
- `block`: use the LLVM translator only, with block-level translation.
- `hybrids`: HQEMU running emulation and LLVM optimizations on one single thread.
- `hybridm`: HQEMU running emulation and LLVM optimizations on different threads. (the default mode, same as LLVM_MODE unset)

5 Installing HQEMU on the PPC64 LE host

5.1 Install dependent packages

The following software packages are required by HQEMU:

- LLVM 5.0 or newer
- Clang 5.0 or newer
- GCC 4.8 or newer

The following example assumes that you will build LLVM 6.0 and Clang 6.0 from source, and install them in the /usr/local directory. $HQEMU denotes the top directory of the HQEMU source tree. ppc64le host machine is used in this example.

5.1.1 Build LLVM and Clang

```
$ ls
cfe-6.0.0.src.tar.xz  compiler-rt-6.0.0.src.tar.xz
llvm-6.0.0.src.tar.xz

$ tar -xf llvm-6.0.0.src.tar.xz
$ cd llvm-6.0.0.src
$ patch -p1 < $HQEMU/patch/llvm/llvm-6.0.patch
$ mkdir build

$ cd tools
$ tar -xf ../../cfe-6.0.0.src.tar.xz
$ mv cfe-6.0.0.src clang
$ cd ../projects
$ tar -xf ../../compiler-rt-6.0.0.src.tar.xz
$ mv compiler-rt-6.0.0.src compiler-rt
$ cd ../build

$ cmake -G "Unix Makefiles" \
        -DCMAKE_INSTALL_PREFIX=/usr/local/llvm-6.0 \
        -DCMAKE_BUILD_TYPE=Release ..

$ make && make install
$ export PATH=/usr/local/llvm-6.0/bin:$PATH
```

Note that the patch file “llvm-VERSION.patch” is placed in the $HQEMU/patch/llvm directory.

5.2 Build HQEMU

The building of HQEMU requires LLVM tool *llvm-config* to locate LLVM header files and libraries, and *clang* to generate LLVM bitcode file. Therefore, make sure the LLVM installation path has been added to your PATH environment variable. The default steps for building HQEMU are as follows:

```
$ tar -zxf hqemu-2.5.2.tar.gz
$ mkdir build
$ cd build
$ ../hqemu-2.5.2/configure --prefix=`pwd` \
                           --target-list=x86_64-linux-user \
                           --enable-llvm
$ make && make install
```

Here are some configuration options which may be of interest:

- --prefix=<path>: install all files in the specified directory. /usr/local/ is the default path if --prefix is not specified.
- --target-list=<list>: set target ISA lists. Use comma to separate targets if multiple targets are specified, for example, --target-list=i386-linux-user,arm-linux-user.

5.3 Run HQEMU

You can download user mode test programs "linux-user-test-0.3.tar.gz" from http://itanium.iis.sinica.edu.tw/hqemu/download/linux-user-test-0.3.tar.gz. This tarball consists of test programs and runtime libraries for Linux user-mode emulation.

```
$ cd build
$ tar -zxf linux-user-test-0.3.tar.gz
$ ./bin/qemu-x86_64 -L ./linux-user-test-0.3/gnemul/qemu-x86_64 \
                    ./linux-user-test-0.3/x86_64/busybox echo Hello

Hello
```

Users can use the environment variable, LLVM_MODE, to control HQEMU's execution mode, e.g.,

```
$ export LLVM_MODE=<mode>
```

The available modes are:

- `none`: use the QEMU/TCG translator only. No LLVM optimization is used.
- `block`: use the LLVM translator only, with block-level translation.
- `hybrids`: HQEMU running emulation and LLVM optimizations on one single thread.
- `hybridm`: HQEMU running emulation and LLVM optimizations on different threads. (the default mode, same as LLVM_MODE unset)

6 Cross-compiling HQEMU for the PPC64 LE host

6.1 Install dependent packages

The following software packages are required by HQEMU:

- LLVM 5.0 or newer
- Clang 5.0 or newer
- x86-64 GCC 4.8 or newer
- ppc64le GCC toolchain (IBM advance-toolchain-atX.X-cross-ppc64le is recommended. Link: https://developer.ibm.com/linuxonpower/advance-toolchain/)

The following example assumes that you will build LLVM 6.0 and Clang 6.0 from source, and install them in the /usr/local directory. $HQEMU denotes the top directory of the HQEMU source tree. x86-64 host machine is used as the platform for cross-compilation. Assume the IBM Advance Toolchain is with version 11.0 and is installed in /opt/at11.0/.

6.1.1 Build the Clang cross-compiler for ppc64le

```
$ ls
cfe-6.0.0.src.tar.xz  compiler-rt-6.0.0.src.tar.xz
llvm-6.0.0.src.tar.xz

$ tar -xf llvm-6.0.0.src.tar.xz
$ cd llvm-6.0.0.src
$ mkdir build

$ cd tools
$ tar -xf ../../cfe-6.0.0.src.tar.xz
$ mv cfe-6.0.0.src clang
$ cd ../projects
$ tar -xf ../../compiler-rt-6.0.0.src.tar.xz
$ mv compiler-rt-6.0.0.src compiler-rt
$ cd ../build

$ cmake -G "Unix Makefiles" \
        -DCMAKE_INSTALL_PREFIX=/usr/local/llvm-6.0 \
        -DCMAKE_BUILD_TYPE=Release ..

$ make && make install
$ cp ./bin/clang-tblgen /usr/local/llvm-6.0/bin/
```

6.1.2 Build the ppc64le LLVM library

```
$ tar -xf llvm-6.0.0.src.tar.xz
$ cd llvm-6.0.0.src
$ patch -p1 < $HQEMU/patch/llvm/llvm-6.0.patch
$ mkdir build-ppc64le
$ cd build-ppc64le

$ cmake -G "Unix Makefiles" \
        -DCMAKE_INSTALL_PREFIX=/usr/local/llvm-6.0-ppc64le \
        -DCMAKE_BUILD_TYPE=Release \
        -DCMAKE_CROSSCOMPILING=True \
        -DLLVM_TABLEGEN=/usr/local/llvm-6.0/bin/llvm-tblgen \
        -DCLANG_TABLEGEN=/usr/local/llvm-6.0/bin/clang-tblgen \
        -DLLVM_DEFAULT_TARGET_TRIPLE=powerpc64le-linux-gnu \
        -DCMAKE_C_COMPILER=powerpc64le-linux-gnu-gcc \
        -DCMAKE_CXX_COMPILER=powerpc64le-linux-gnu-g++ \
        ..

$ make && make install
$ rm -f /usr/local/llvm-6.0-ppc64le/bin/*
$ cp -r /usr/local/llvm-6.0/bin/* /usr/local/llvm-6.0-ppc64le/bin/
$ cp -r /usr/local/llvm-6.0/lib/clang /usr/local/llvm-6.0-ppc64le/lib/
$ export PATH=/usr/local/llvm-6.0-ppc64le/bin:$PATH
```

Note that the patch file "llvm-VERSION.patch" is placed in the $HQEMU/patch/llvm directory.

6.2 Build HQEMU

The building of HQEMU requires LLVM tool *llvm-config* to locate LLVM header files and libraries, and *clang* to generate LLVM bitcode file. Therefore, make sure the LLVM installation path has been added to your PATH environment variable. The default steps for building HQEMU are as follows:

```
$ export SYSROOT=/opt/at11.0/ppc64le/

$ tar -zxf hqemu-2.5.2.tar.gz
$ mkdir build
$ cd build
$ ../hqemu/configure --prefix=`pwd` \
                     --target-list=x86_64-linux-user \
                     --enable-llvm \
                     --cross-prefix=powerpc64le-linux-gnu- \
                  --clang-flags="-target ppc64le --sysroot $SYSROOT"
$ make && make install
```

Here are some configuration options which may be of interest:

- --prefix=<path>: install all files in the specified directory. /usr/local/ is the default path if --prefix is not specified.
- --target-list=<list>: set target ISA lists. Use comma to separate targets if multiple targets are specified, for example, --target-list=i386-linux-user,arm-linux-user.

6.3 Run HQEMU

You can download user mode test programs "linux-user-test-0.3.tar.gz" from http://itanium.iis.sinica.edu.tw/hqemu/download/linux-user-test-0.3.tar.gz. This tarball consists of test programs and runtime libraries for Linux user-mode emulation.

```
Copy the installation directory 'build/bin' to the ppc64le machine.

$ cd build
$ tar -zxf linux-user-test-0.3.tar.gz
$ ./bin/qemu-x86_64 -L ./linux-user-test-0.3/gnemul/qemu-x86_64 \
                    ./linux-user-test-0.3/x86_64/busybox echo Hello

Hello
```

Users can use the environment variable, LLVM_MODE, to control HQEMU's execution mode, e.g.,

```
$ export LLVM_MODE=<mode>
```

The available modes are:

- `none`: use the QEMU/TCG translator only. No LLVM optimization is used.
- `block`: use the LLVM translator only, with block-level translation.
- `hybrids`: HQEMU running emulation and LLVM optimizations on one single thread.
- `hybridm`: HQEMU running emulation and LLVM optimizations on different threads. (the default mode, same as LLVM_MODE unset)

7 HQEMU options

HQEMU provides several options for users to control the execution of HQEMU, such as printing debug information, profiling, and turning optimizations on/off. Options are passed via the LLVM_CMD environment variable. Multiple options can also be passed at the same time by separating them with space. For example:

```
$ export LLVM_CMD='option1=val1 option2=val2'
$ export LLVM_CMD='-debuglv=debug -profile=basic,trace -threads=4'
```

To display HQEMU specific options, run the command before running HQEMU:

```
$ export LLVM_CMD='-help'
```

To display all available options including HQEMU and LLVM builtin options, use the command:

```
$ export LLVM_CMD='-help-hidden'
```

7.1 Debug HQEMU

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 is assigned by setting the "`-debuglv=`" option in the LLVM_CMD environment variable. The default debugging level is set to `none`. You can enable multiple debugging levels at the same time with desired debug masks 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 before HQEMU specific optimizations.
- `ir_opt`: LLVM IRs after HQEMU specific optimizations.
- `entry`: translation functions that are called.
- `verify`: check whether the LLVM function is well-formed.
- `asm`: turn on debugging levels of `in_asm`, `op` and `out_asm`.
- `debug`: turn on debugging levels of `llvm`, `ir_opt` 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 messages with debug masks, `llvm` and `ir_opt`, being set:

```
$ export LLVM_CMD='-debuglv=llvm,ir_opt'
$ ./bin/qemu-i386 -L ./linux-user-test-0.3/gnemul/qemu-i386/ \
                     ./linux-user-test-0.3/i386/sha1

[00:00:00.001080] Initializing LLVM Environment.
[00:00:00.001309] Creating 1 translator(s).
[00:00:00.001322] Starting LLVM Translator 0.
[00:00:00.014509] LLVM initialized for target (x86_64-linux-gnu).
[00:00:00.019975] Use LLVM disassembler for guest (i386).
[00:00:00.019985] Use LLVM disassembler for host (x86_64).
[00:00:00.020002] LLVM IR Factory initialized.
[00:00:00.020070] LLVM Translator 0 initialized.
[00:00:00.042496] LLVM MCJIT initialized.
[00:00:00.042519] Requested trace info: pc 0x40802a46 length 1
[00:00:00.049942]     - Found a direct branch to pc 0x40802a53
[00:00:00.050006]     - Found a direct branch to pc 0x40802a46
[00:00:00.050026] Compile: start...
[00:00:00.050054] PostProcess: pc 0x40802a46 length 1 is_loop 1

define i64 @"40802a46"() nounwind naked {
init:
  %cpu = call i8* asm sideeffect "", "={r14}"() nounwind
  %cpu.struct = bitcast i8* %cpu to %struct.CPUX86State*
  %0 = getelementptr i8* %cpu, i32 8
  %1 = getelementptr i8* %cpu, i32 0
  %2 = getelementptr i8* %cpu, i32 48
  %3 = getelementptr i8* %cpu, i32 40
  %4 = getelementptr i8* %cpu, i32 44
  %edx = bitcast i8* %0 to i32*
  %eax = bitcast i8* %1 to i32*
  %cc_op = bitcast i8* %2 to i32*
  %cc_src = bitcast i8* %3 to i32*
  %cc_dst = bitcast i8* %4 to i32*
  %env = bitcast i8* %1 to i64*
  (......)
  br label %entry

entry:                                ; preds = %loopback, %init
  %eax.a.0 = phi i32 [ %5, %init ], [ %13, %loopback ]
  %10 = shl i32 %eax.a.0, 2
  %11 = add i32 %6, %10
  %12 = inttoptr i32 %11 to i32*
  store volatile i32 0, i32* %12
```

```
  %13 = add i32 %eax.a.0, 1
  %14 = sub i32 %13, 97
  %15 = add i32 %14, 97
  %16 = icmp ule i32 %15, 97
  br i1 %16, label %loopback, label %false_dest.block

loopback:                                 ; preds = %entry
  br label %entry, !exit !0
  (......)
}
```

7.2 Profile HQEMU

Developers who wish to profile a program's execution behavior can use the "`-profile=`" option in the LLVM_CMD environment variable to control the profiling levels of a target program. The default profiling level is set to `none`. You can enable multiple profiling levels at the same time with desired values separated by comma. The available profiling mask values are listed below:

- `none`: no profiling message will be displayed. This is the default profiling level.
- `basic`: static information. This level includes amount of basic blocks, 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.
- `trace`: trace's runtime information. This level includes the values of trace counters.
- `cache`: addresses/sizes of basic block cache and trace caches.
- `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_CMD='-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)
-------------------------
Trace information:
-------------------------
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
    (......)
  24 (0: 1)        0/  35832: (16.0) 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 0 355 355 21951
  26 (1: 2)     1736/   2013: ( 7.3) 277
  27 (0: 4)        0/     93: ( 2.4) 0 56 37
```

8 Frequently asked questions

1. Why the program is terminated with the message "Error: cannot find bitcode file llvm_helper_X.bc"? HQEMU searches for the bitcode file 'llvm_helper_X.bc' in the order of the following directories: (a) /etc/hqemu/, (b) $HOME/.hqemu/, and (c) the bin/ folder under the configured installation path (e.g., build/bin/). You need to manually copy the bitcode file to any of the aforementioned directories if it is not correctly installed.