# 安装

Ubuntu 14.04

内核4.2.0

Gcc 4.8

pin-2.14-71313-gcc.4.4.7-linux

首先安装必要的软件：sudo apt-get install libconfig++-dev libhdf5-dev libelf-dev

1. build/opt/config.cpp:32:25: fatal error: libconfig.h++: 没有那个文件或目录

sudo apt-get install libconfig++-dev

1. fatal error: hdf5.h: no such file or directory

把源代码中的#include<hdf5.h>改成 #include<hdf5/serial/hdf5.h>

把源代码中的#include<hdf5\_hl.h>改成 #include<hdf5/serial/hdf5\_hl.h>

1. error: #error This kit requires gcc 3.4 or later

打开文件/home/zhangqianlong/MyPHD/pin-2.14-71313-gcc.4.4.7-linux/source/include/pin/compiler\_version\_check2.H，注释掉相关报错语句

1. fatal error: gelf.h:

sudo apt-get install libelf-dev

1. Pin2.14出问题：./build/opt/zsim tests/simple.cfg运行时报错 4.4 not support

https://github.com/s5z/zsim/issues/109

根据这里的介绍，一开始我在pin里面查找pin\_bin.cpp或者pin\_cmd.cpp  发现没有这些文件，我开始在pycharm中，全局搜索zsim\_nvmain，终于在pin外面的src下找到了pin\_cmd.cpp，然后根据博客中的介绍，在52行添加

args.push\_back("-injection");

args.push\_back("child");

这时候编译不再报错内核不支持了。

实验平台终于可以接着搭建了，接着执行测试，真是一波三折，新的错误又来了：

unexpected AUX VEC type 26

经过查找，需要加-ifeellucky到pin的配置里面。我尝试着再次修改pin\_cmd.cpp文件，这里还是pin2.13版本不支持4.\*内核的问题。

args.push\_back("-ifeellucky");

1. 安装gcc4.8.5

Sudo apt-get install gcc-4.8

Sudo apt-get install g++-4.8

1. 内核降级：

<https://blog.csdn.net/u013431916/article/details/82530523>

1. /usr/include/asm/unistd.h: No such file or directory

sudo apt-get install build-essential

sudo ln -s /usr/include/asm-generic /usr/include/asm

1. Panic on build/opt/zsim.cpp:696: Loaded vDSO not valid

把内核降级为3.13

1. gm\_create failed shmget: Invalid argument

zsim目录的README.md中提到：

Host Configuration: The system configuration may need some tweaks to support

zsim. First, it needs to allow for large shared memory segments. Second, for

Pin to work, it must allow a process to attach to any other from the user, not

just to a child. Use sysctl to ensure that `kernel.shmmax=1073741824` (or larger)

and `kernel.yama.ptrace\_scope=0`. zsim has mainly been used in

Ubuntu 11.10, 12.04, 12.10, 13.04, and 13.10, but it should work in other Linux

distributions. Using it in OSs other than Linux (e.g,, OS X, Windows) will be

non-trivial, since the user-level virtualization subsystem has deep ties into

the Linux syscall interface.

解决办法：根据运行时错误提示：[H] Creating global segment, 8192 MBs，选择下面命令执行：

sudo sysctl -w kernel.shmmax=1073741824

sudo sysctl -w kernel.shmmax=8589934592

1. SetupArgumentBranchTarget: 2846: assertion failed: INS\_CallOrBranchIsMemoryIndirect(ins) #

参考：<https://github.com/s5z/zsim/issues/154>

解决办法：修改zsim.cpp：

566 // Instrument only conditional branches

! 567 // For kernel version larger than 4.0, there will be some assertion failed

+ 568 // see https://github.com/s5z/zsim/issues/154

+ 569 // workaround is add INS\_IsXend(ins)

+ 570 //if (INS\_Category(ins) == XED\_CATEGORY\_COND\_BR) {

+ 571 if (INS\_Category(ins) == XED\_CATEGORY\_COND\_BR && !INS\_IsXend(ins)) {

重要参考：

<https://blog.csdn.net/qqq_11101/article/details/78481377>

1. 解决如下问题：g++ -o build/opt/fftoggle --static build/opt/fftoggle.o build/opt/config.o build/opt/galloc.o build/opt/log.o build/opt/pin\_cmd.o -lconfig++ -lpthread

/usr/lib/gcc/x86\_64-linux-gnu/4.7/../../../x86\_64-linux-gnu/libconfig++.a(libconfig\_\_\_la-libconfigcpp.o): In function `void std::\_\_cxx11::basic\_string<char, std::char\_traits<char>, std::allocator<char> >::\_M\_construct<char\*>(char\*, char\*, std::forward\_iterator\_tag)':

/usr/include/c++/5/bits/basic\_string.tcc:223: undefined reference to `std::\_\_cxx11::basic\_string<char, std::char\_traits<char>, std::allocator<char> >::\_M\_create(unsigned long&, unsigned long)'

把/home/zhangqianlong/MyPHD/Simulator/Banshee/ext\_lib/libconfig/lib中的库文件替换系统库文件/usr/lib/x86\_64-linux-gnu

# 代码阅读

1. 打开log：打开src/log.h，添加#define \_LOG\_TRACE\_ ,在trace定义里，修改if ( LOG\_##type == LOG\_Cache)
2. Zsim如何使用gdb？卡死在文件zsim.cpp的函数1475的main中的while判断中，

继续往下追，发现是init.cpp中的SimInit函数卡主了;

再往下追，发现是函数中的sleep(1)卡住了。

95 void notifyHarnessForDebugger(int harnessPid) {

+ 96 info("In func %s, 1", \_\_func\_\_);

97 kill(harnessPid, SIGUSR1);

+ 98 info("In func %s, 2", \_\_func\_\_);

99 sleep(1); //this is a bit of a hack, but ensures the debugger catches us

+ 100 info("In func %s, 3", \_\_func\_\_);

101 }

1475 bool masterProcess = false;

1476 if (procIdx == 0 && !gm\_isready()) { // process 0 can exec() without fork()ing first, so we must check gm\_isready() to ensure we don't initialize twice

+ 1477 info("procIdx = %d\n", procIdx);

1478 masterProcess = true;

1479 SimInit(KnobConfigFile.Value().c\_str(), KnobOutputDir.Value().c\_str(), KnobShmid.Value());

1480 } else {

+ 1481 info("In else1\n");

1482 while (!gm\_isready()) usleep(1000); // wait till proc idx 0 initializes everything

+ 1483 info("In else2\n");

1484 zinfo = static\_cast<GlobSimInfo\*>(gm\_get\_glob\_ptr());

1485 }