# 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 xterm gdb

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

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("parent"); //不能用child，会导致zsim的debug功能无法使用**

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");

! 54 › //Global pin options

! 55 › args.push\_back("-follow\_execv"); //instrument child processes

! 56 › args.push\_back("-tool\_exit\_timeout"); //don't wait much of internal threads

! 57 › args.push\_back("1");

+ 58 › args.push\_back("-ifeellucky");

+ 59 args.push\_back("-injection");

+ 60 args.push\_back("parent");

1. 安装gcc4.8.5

Sudo apt-get install gcc-4.8

Sudo apt-get install g++-4.8

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. 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

1. File "../../../../scripts/ipc.py", line 2, in <module>

import h5py

ImportError: No module named h5py

sudo apt-get install python-h5py

1. /usr/bin/ld: cannot find -lhdf5

/usr/bin/ld: cannot find -lhdf5\_hl

sudo ln -s /usr/lib/x86\_64-linux-gnu/libhdf5\_serial\_hl.so /usr/lib/libhdf5\_hl.so

1. Unable to load xxx/libzsim.so: xxx/libzsim.so: undefined symbol: \_ZN10LEVEL\_BASE9StringDecB5cxx11Emjc

解决办法：

diff --git a/SConstruct b/SConstruct

index 66f3a74..9eea125 100644

--- a/SConstruct

+++ b/SConstruct

@@ -47,7 +47,7 @@ def buildSim(cppFlags, dir, type, pgo=None):

# NOTE: Original Pin flags included -fno-strict-aliasing, but zsim does not do type punning

# NOTE (dsm 16 Apr 2015): Update flags code to support Pin 2.14 while retaining backwards compatibility

env["CPPFLAGS"] += " -g -std=c++0x -Wall -Wno-unknown-pragmas -fomit-frame-pointer -fno-stack-protector"

- env["CPPFLAGS"] += " -MMD -DBIGARRAY\_MULTIPLIER=1 -DUSING\_XED -DTARGET\_IA32E -DHOST\_IA32E -fPIC -DTARGET\_LINUX"

+ env["CPPFLAGS"] += " -MMD -DBIGARRAY\_MULTIPLIER=1 -DUSING\_XED -DTARGET\_IA32E -DHOST\_IA32E -fPIC -DTARGET\_LINUX -D\_GLIBCXX\_USE\_CXX11\_ABI=0"

## 其他参考

1. 内核降级：

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

# 代码阅读

## 打开Log

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)

## Gdb的bug

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 }

## MemReq访问路径

TimingCore::loadAndRecord[timing\_core.cpp]

Load()[filter\_cache.h]

Replace()[filter\_cache.h]

Access()[filter\_cache.h]

Cache::access()[cache.cpp]

cc->processAccess()[cache.cpp]

processAccess()[coherence\_ctrls.h]

InitSystem()[init.cpp]初始化时cache和memory之间互相连接起来了，L1的parent是L2，LLC的parent是内存控制器，如果内存控制器大于等于2，则创建SplitAddrMemory()，以后LLC的parent就是SplitAddrMemory，然后通过SplitAddrMemory再分别按照interleave访问内存控制器。

## \_mcdram和\_ext\_dram区别是什么？

Cache hit时只调用\_mcdram->access，否则要先后调用\_mcdram->access和\_ext\_dram->access，因此前者是混合存储做Cache时的dram

## Off-package的内存大小没有指定？

## On-package的内存大小如何初始化？

MemoryController::MemoryController函数中初始化\_cache\_size，

默认每个片上内存有4个MC channel：

\_mcdram\_per\_mc = config.get<uint32\_t>("sys.mem.mcdram.mcdramPerMC", 4);

请求地址到达MC后，会被处理用于选择去哪个片上内存的channel执行：

uint32\_t mcdram\_select = (address / 64) % \_mcdram\_per\_mc;

……

req.cycle = \_mcdram[mcdram\_select]->access(req, 0, 4);

## AlloyCache的命中hitway信息没有

因为AlloyCache的granularity必须等于64，并且只有一个way

## 片上内存的带宽哪里指定的？

默认是四个channel，是普通DRAM的四倍

## Cache-only的配置是几路，size如何指定成无限大小？

不用指定，因为不会使用cache结构体，在mc.cpp的access中只去访问mcdram就返回了

## Tagless和Alloy-0.1比hybrid要性能好，什么原因？

有的程序，不需要大带宽，但是局部性比较好，在hybrid中有FBR，在FBR不起作用之前，Tagless这种以footprint为粒度的，且miss后立刻换入的方法是最好的。