# 代码说明

## 1模拟器代码femu

femu

├── backend #从qemu中划分一块内存空间用于模拟SSD

│   ├── dram.c #提供初始化、销毁以及捕获读写请求等接口函数

│   └── dram.h #定义声明有关用于模拟SSD的内存空间

├── bbssd #用于模拟黑盒SSD的工程文件

│   ├── bb.c #定义了黑盒SSD的注册接口以及相关指令处理

│   ├── ftl.c #提供了黑盒SSD的初始化函数以及针对读写操作的相应修改

│   └── ftl.h #声明了黑盒SSD的物理结构以及延时设定

├── dma.c #提供模拟NVMe控制器的部分功能，包括DMA的读写功能等

├── femu.c #实现虚拟 NVMe 控制器，模拟了 NVMe（非易失性内存表达）设备的行为

├── Hybrid\_SSD1 #混合SSD的baseline版本（传统缓存策略和无预取机制）

│   ├── ftl.c #提供了混合SSD的初始化函数以及针对读写操作的相应修改

│   ├── ftl.h #定义了混合SSD的物理结构、延迟信息以及管理结构体

│   └── hb.c #定义了混合SSD的注册函数以及部分延迟设置

├── Hybrid\_SSD3 #混合SSD的HRCMRS版本（提出的缓存策略和无预取机制）

│   ├── ftl.c #提供了混合SSD的初始化函数、读写操作、缓存迁移等

│   ├── ftl.h #定义了混合SSD的物理结构、延迟信息、缓存管理结构体和管理结构体

│   └── hb.c #定义了混合SSD的注册函数以及部分延迟设置

├── Hybrid\_SSD4 #混合SSD的PLGDP版本（传统缓存策略和提出的预取机制）

│   ├── ftl.c #提供了混合SSD的初始化函数、读写操作、预取操作等

│   ├── ftl.h #定义了混合SSD的物理结构、延迟信息、历史记录结构和管理结构等

│   └── hb.c #定义了混合SSD的注册函数以及部分延迟设置

├── Hybrid\_SSD5 #混合SSD的PLGDP+HRCMRS版本（提出的缓存策略和预取机制）

│   ├── ftl.c #提供了混合SSD的初始化函数、读写操作、预取操作、缓存迁移等

│   ├── ftl.h #定义了混合SSD物理结构、延迟信息、缓存管理和数据预取结构等

│   └── hb.c #定义了混合SSD的注册函数以及部分延迟设置

├── inc #

│   ├── pqueue.h #定义了一个优先队列的实现，具体包括优先队列的数据结构、初始化、操作函数等。

│   ├── rte\_atomic\_x86.h #定义了一些基本的原子操作和编译器屏障，主要用于多线程编程中确保操作的原子性和内存顺序一致性。

│   ├── rte\_branch\_prediction.h #用于提高分支预测的效率

│   ├── rte\_ring\_generic.h #对环形缓冲区（ring buffer）操作的一部分实现，专注于环形缓冲区的生产者（入队）和消费者（出队）头部指针的管理。

│   └── rte\_ring.h #定义了一个名为rte\_ring的数据结构，这是一个环形缓冲区（或称为环形队列），并提供了一系列函数来管理和操作这个环形缓冲区

├── intr.c #处理虚拟NVMe控制器的中断管理，包括设置、清理和触发中断

├── Kconfig #系统配置文件

├── lib #提供了两中优先级队列的具体功能实现

│   ├── pqueue.c #实现了优先级队列的具体操作函数接口

│   └── rte\_ring.c #实现了环形队列的具体操作

├── meson.build #指明在编译过程中需要进行编译的文件

├── nand #提供了NAND闪存的基本设置和操作功能

│   ├── nand.c #提供初始化一个模拟的NAND闪存的页面配对和时序

│   └── nand.h #定义了具体闪存介质页面的操作延迟，以及页面类型设定等信息

├── nossd #用于模拟无挂载SSD的工程文件

│   └── nop.c #提供注册无实际存储后端的操作模式

├── nvme-admin.c #提供处理NVMe（Non-Volatile Memory Express）管理命令的功能。

├── nvme.h #定义了NVMe协议的核心数据结构、命令集、状态码、以及与NVMe设备管理相关的一系列函数和操作

├── nvme-io.c #实现了NVMe I/O指令处理、优先队列处理、中断事务等处理

├── nvme-util.c #实现了NVMe协议中的指令提交队列和完成队列以及其相关的操作

├── ocssd #用于模拟open channel SSD的工程文件（不是本研究相关的SSD类型）

│   ├── oc12.c #

│   ├── oc12.h #

│   ├── oc20.c #

│   └── oc20.h #

├── timing-model #定义了闪存硬盘的一些延时信息

│   ├── timing.c #时间计算函数的具体实现

│   └── timing.h #声明时间计算函数

└── zns #用于模拟ZNS SSD的工程文件

├── zns.c #提供ZNS SSD具体操作的修改接口

└── zns.h #设定一些ZNS SSSD的物理配置信息

## 模拟器内部代码hybrid-femu

hybrid-femu #混合模拟器运行过程中用于测试的脚本函数

├── trace\_file #用于复现各种工作负载的脚本

│   ├──auto\_trace2.py #先进行混合SSD数据初始化，初始化后才能进行记录复现

│   ├──auto\_trace\_read2.py #工作负载访问行为复现脚本

│   ├──auto2.py #整合脚本，先后调用执行auto\_trace2.py和auto\_trace\_read2.py

│   └──其他 #工作负载记录

├── workload

│   └── .txt文件 #工作负载记录

└── FEMU\_old.tar.gz #FEMU整体工程文件（由于文件过大，后续附上网盘链接）

# 环境配置与代码运行

步骤一：在ubutnu虚拟机中解压FEMU\_old.tar.gz，并按照CSDN中的教程配置QEMU以及虚拟机镜像文件，并运行FEMU\_old，如以下链接：

[https://blog.csdn.net/bijie1196/article/details/120752319?ops\_request\_misc=%257B%2522request%255Fid%2522%253A%2522171263174116800227444434%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request\_id=171263174116800227444434&biz\_id=0&utm\_medium=distribute.pc\_search\_result.none-task-blog-2~all~top\_positive~default-1-120752319-null-null.142^v100^pc\_search\_result\_base6&utm\_term=FEMU&spm=1018.2226.3001.4187](https://blog.csdn.net/bijie1196/article/details/120752319?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522171263174116800227444434%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=171263174116800227444434&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~top_positive~default-1-120752319-null-null.142%5ev100%5epc_search_result_base6&utm_term=FEMU&spm=1018.2226.3001.4187)。

步骤二：将文件夹中的femu替换掉FEMU\_old工程文件中” /FEMU\_old/hw/femu”文件，并将文件夹中run-HybridSSD.sh加入到FEMU\_old中build-femu中，并运行脚本即可运行混合SSD模拟器。可以通过“lsblk”指令查看虚拟SSD是否挂载到系统中（是一个nvme设备）。由于FEMU\_old占用空间过大，这里附上百度网盘链接：

链接：https://pan.baidu.com/s/1PeUDFuVF8-r-o5mG4jj3TA?pwd=4do9

提取码：4do9

# 实验数据

实验数据放置在文件中的数据文件夹中，其中数据1是运行总结下来的excell数据，并带有数据图。数据2包含了数据处理脚本以及经脚本处理后的数据和原始执行过程中采集到的数据。