**小矮人程序使用手册**

Version 1

2016.01

**摘要**

小矮人程序使用手册，包括小矮人程序概述、安装配置说明、物理平台使用方法和模拟器平台(Gem5)使用方法。

目录

一、 概述 3

二、 小矮人程序概述 3

1. 小矮人程序介绍 3

2. 小矮人实现方法介绍 4

三、 物理平台使用 7

1. 编译小矮人程序 7

2. 运行小矮人程序 7

四、 模拟器平台使用 9

1. GEM5 配置说明 9

2. GEM5运行介绍 9

1) FS模式 9

2) SE模式 10

3. 在gem5 fs模式下运行小矮人程序 10

1) 挂载gem5镜像 10

2) 传入测试程序及数据集 10

3) 卸载镜像 11

4) 启动gem5 11

5) 编译小矮人程序 11

6) 运行小矮人程序 11

7) 获取运行结果 13

4. 在gem5 se模式下运行小矮人程序 14

1) 安装交叉编译工具 14

2) 交叉编译小矮人程序 14

3) 运行小矮人程序 15

4) 获取运行结果 16

**小矮人程序使用手册**

# 概述

本手册包括：小矮人程序的概述、物理平台使用介绍、模拟器平台使用介绍。主要介绍小矮人程序在物理平台和模拟器平台的安装使用。

# 小矮人程序概述

## 小矮人程序介绍

小矮人程序集一共包含七个程序，通过openmp实现多线程处理，分别为md5，sort，wordcount，union，bfs，matrixMult和fft。

md5负载测试程序实现的功能为：对输入文件的每一行计算出md5值并输出到输出文件中。

sort负载测试程序实现的功能为：将输入的文件按照递增的顺序排序并输出到输出文件中。

wordcount负载测试程序实现的功能为：统计输入文件中各个单词的出现次数并输出到输出文件中。

union负载测试程序实现的功能为：将两个文件合并在一起，参数有两个输入文件、一个输出文件和线程数。

bfs负载测试程序实现的功能为：在输入的图文件中采用广度优先搜索算法搜索整个图文件，在程序中自带了图文件的生成程序。

matrixMult负载测试程序实现的功能为：计算输入矩阵的乘法。

fft负载测试程序实现的功能为：对输入数据进行快速DFT运算。

## 小矮人实现方法介绍

本节介绍md5，sort，wordcount，union，bfs，matrixMult和fft七个小矮人程序的实现方法。

1) md5

数据输入：单个或多个文件，每个文件行看作一个元素。

数据输出：每个输入文件会输出到一个结果文件中。

逻辑上，对每个文件的每一行计算其md5值，并输出到结果文件的相应行。

具体的，main()函数主要是参数处理和文件/文件夹处理。对每个文件，调用do\_one\_file()来处理。

do\_one\_file()主要是：单线程读取文件内容，每次读取合适的大小，放入内存做md5计算，内存中的数据分成n份，n为线程个数；多线程各自做md5计算，每个线程调用md5()对前述内存数据中的一份进行md5计算；最后单线程将前述的结果写入结果文件。

2) sort

数据输入：每个文件行看作一个元素，对这些元素进行比较排序。

数据输出：所有文件会被归并到一个输出文件中。

逻辑上，将所有输入文件看作一个大的输入文件。首先，对这个大的输入文件切分成split，这些split可以读入内存来做内存排序。将每个已排序的split输出为一个中间结果文件，然后对所有的中间结果文件做多路归并，输出最后的排序结果。

具体的，main.c文件主要是参数处理和文件/文件夹处理。对每个文件，调用do\_one\_file()来处理。

do\_one\_file()主要是：单线程读取文件内容，每次读取合适的大小，放入内存做内存排序，内存中的数据分成n份，n为线程个数；多线程各自排序，每个线程调用系统库的qsort()对前述内存数据中的一份进行排序，所以排序的结果是内存数据变成了n个有序的数据；单线程多路归并写入中间结果文件，将前述的n个有序的数据，归并到中间结果文件中。

最后，main()调用do\_merge\_sort()对所有中间结果文件做单线程多路归并。

在模拟器版本的开发中，我们对第一阶段开发的小矮人进行调优，包括结构体的对齐，分片大小的处理，以及排序算法的改进。

3) wordcount

数据输入：每个单词看作一个元素，对这些元素进行计数。

数据输出：所有文件会被归并到一个输出文件中。

逻辑上，将所有输入文件看作一个大的输入文件。首先，对这个大的输入文件切分成split，这些split可以读入内存来做wordcount。将每个已处理的split输出为一个中间结果文件，然后对所有的中间结果文件做多路归并，输出最后的计数结果。

具体的，main.c文件主要是参数处理和文件/文件夹处理。对每个文件，调用do\_one\_file()来处理。

do\_one\_file()主要是：单线程读取文件内容，每次读取合适的大小，放入内存，对内存数据做英文分词；内存中的数据分成n份，n为线程个数；多线程各自排序，排序的元素是单词，每个线程调用系统库的qsort()对前述内存数据中的一份进行排序，所以排序的结果是内存数据变成了n个有序的数据；单线程多路归并写入中间结果文件，将前述的n个有序的数据，先压缩成“单词-计数”对，然后归并到中间结果文件中。

最后，main()调用do\_merge\_sort()对所有中间结果文件做单线程多路归并。

4) union

数据输入：每个文件行看作一个单词的集合，对两个对应的集合做union。

数据输出：每两个相应输入文件会输出到一个结果文件中。

逻辑上，每个文件行看作一个单词的集合，将两个相应输入文件的相应行做union，并将结果输出到相应的结果文件中。

具体的，main.c文件主要是参数处理和文件/文件夹处理。对每个文件，调用do\_one\_file()来处理。

do\_one\_file()主要是：单线程读取文件内容，每次读取合适的大小，放入内存；若两个文件行数相同，对内存数据中的相应行做union，若不同，则其中一部分文件行将只做单词去重；内存中的数据分成n份，n为线程个数；多线程各自做union，每个线程调用union\_two\_line()或union\_one\_line()对前述内存数据中的一份进行按行union；最后单线程写入结果文件。

5) bfs

数据输入：graph500的图数据文件。

数据输出：graph500运行后的统计信息。

首先，生成文本表示的图数据。修改 ../../data\_gen/BigDataGeneratorSuite/Graph\_datagen/gen\_unweighted\_and\_split.sh中的参数，详见其中的注释和readme文件，运行之，得到文本数据文件。

然后，运行 ./ run\_my\_make\_edgelist.sh 将文本数据文件转换成graph500的dump文件。

最后，运行graph500的单线程和多线程程序。

6) matrixMult

数据输入：两个文件，分别表示两个稀疏矩阵A和B，矩阵风格类似于matrix market，但数据不一定是按行或列排序的。

数据输出：两个稀疏矩阵的乘积会输出到一个结果文件中。

具体的，main.c文件主要是参数处理。对输入文件，调用do\_one\_file()来处理。

do\_one\_file()主要是：单线程读取文件内容，将前一个文件读入一个CRS格式的稀疏矩阵中，将后一个文件读入一个CCS格式的稀疏矩阵中；读入稀疏矩阵时，由于输入文件中数据不一定有序，所以会进行计数和排序；多线程对前述两个稀疏矩阵做矩阵乘法，每个线程对每次负责计算结果矩阵的一行，即A的一行和B的所有列相乘，对CRS和CCS而言，相当于多次的merge；最后单线程写入结果文件。

7) fft

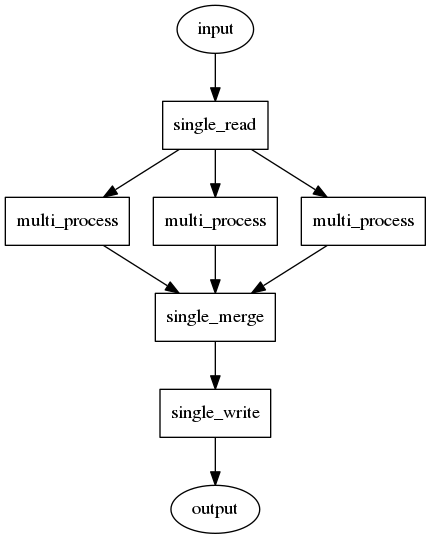
数据输入：一个稀疏矩阵文件，矩阵风格类似于matrix market，但数据不一定是按行或列排序的。

数据输出：计算输入矩阵的二维fft，将结果写入输出文件。

具体的，main.c文件主要是参数处理。对输入文件，调用do\_one\_file()来处理。

do\_one\_file()主要是：单线程读取文件内容，将输入文件读到一个二维数组矩阵中，置实部为矩阵数据，置虚部为0；多线程计算输入矩阵的二维fft，调用fftw3库中的ffw3\_execute()执行；最后单线程写入结果文件。

md5、sort、wordcount、union的逻辑可以抽象成下面的流程图。



# 物理平台使用

## 编译小矮人程序

以编译sort程序为例。进入到sort/src目录中，执行make all命令后就开始编译sort程序，生成的可执行文件single\_thread和multi\_thread在sort/src/run目录下，在编译完成并且没有报错的情况下就可以运行测试程序了。

其余负载测试程序编译过程和sort程序一样，在此不做赘述。

## 运行小矮人程序

1）运行sort程序。进入到sort/src/run目录中，执行如下命令：

./multi\_thread ../dataset/test.data.01 ./output/test.data.01 3

./single\_thread ../dataset/test.data.01 ./output/test.data.01

其中“multi\_thread”为多线程，“3”为线程数，“../dataset/test.data.01”为测试数据集，“./output/test.data.01”为输出结果。Sort程序第一个参数可以是目录或者单个文件名，第二个参数是文件名，所有的数据合并到同一个文件。

2）运行wordcount程序的过程和sort程序一样。

3）运行md5程序的过程和sort程序一样。

4）运行union程序的步骤为：进入到run目录中，执行如下命令：

./multi\_thread ../dataset/test.data.01 ../dataset/test.data.02 ./output 3

./single\_thread ../dataset/test.data.01 ../dataset/test.data.02 ./output

其中“multi\_thread”为多线程，“3”为线程数，“../dataset/test.data.01”为测试数据集1，“../dataset/test.data.02”为测试数据集2，“./output/test.data.01”为输出结果。

5）运行bfs程序的步骤为：首先生成测试数据集，使用命令如下：

cd ../../data\_gen/BigDataGeneratorSuite/Graph\_datagen/

vim gen\_unweighted\_and\_split.sh

./gen\_unweighted\_and\_split.sh

之后将生成好的输入数据放到bfs/dataset/文件夹中。之后运行如下命令：

make my\_make\_edgelist

./run\_my\_make\_edgelist.sh ./dataset/gen\_graph\_10\_2640.txt

准备好输入数据集之后就可以运行负载测试了，运行负载测试的命令如下：

./multi\_thread 10 3 ../dataset/g.10\_2640.dump 4

./single\_thread 10 3 ../dataset/g.10\_2640.dump

其中“multi\_thread”为多线程，“4”为线程数，“../dataset/g.10\_2640.dump”为测试数据集。

6）运行matrixMult程序。进入到matrixMult/src/run目录中，执行如下命令：

./multi\_thread ../dataset/part-00000 ../dataset/part-00000 ./output/out.s.mtx 3

./single\_thread ../dataset/part-00000 ../dataset/part-00000 ./output/out.m.mtx

其中“multi\_thread”为多线程，“3”为线程数，“../dataset/part-00000”为测试数据集，“./output/out.m.mtx”为输出结果。

7）运行fft程序的方法如下：

因为fft程序需要用到fftw库，所以要先安装fftw库。首先需要下载fftw库，下载地址为：http://fftw.org/fftw-3.3.4.tar.gz。之后解压缩，进入到文件夹中。

./configure –enable-openmp

make

make install

运行小矮人程序fft的命令如下：

./multi\_thread ../dataset/part-00000 ./output/out.s.mtx 3

./single\_thread ../dataset/part-00000 ./output/out.m.mtx

其中“multi\_thread”为多线程，“3”为线程数，“../dataset/part-00000”为测试数据集。

# 模拟器平台使用

## GEM5 配置说明

在启动gem5模拟器时可以根据用户需要设置不同的cpu数量、cache以及内存大小。

如果需要使用多个cpu启动gem5模拟器，可以在启动gem5时加入参数：--num-cpus=2。

如果需要设置gem5模拟器的内存大小，可以在启动gem5时加入参数：--mem-size=2048MB。

如果需要设置gem5模拟器的cache值，可以在启动gem5时加入参数：--caches --l2cache --l1d\_size=32kB --l1i\_size=32kB --l2\_size=256kB --l3\_size=20MB --l1d\_assoc=4 --l1i\_assoc=4 --l2\_assoc=8 --l3\_assoc=16

## GEM5运行介绍

### FS模式

FS（full system）模式模拟完整的全系统，包括OS，运行在用户态和核心态的线程调度以及各种设备。能够精确模拟系统时间等开销。在FS模式下能够启动完整的系统，建模硬件设备，能够执行中断、例外、特权指令、故障处理函数，运行速度慢。

### SE模式

SE（syscall emulation）模式能够仿真大部分操作系统级服务，能够取得很好功能模拟加速比。SE模式可以用来运行单个应用以及在MP/SMT上的一系列指令，可以建模用户可见的ISA和常见的系统调用，通过调用主机操作系统来模拟系统调用，能够简化地址翻译模型。

## 在gem5 fs模式下运行小矮人程序

在gem5 fs模式下运行需要的步骤为：1）挂在gem5镜像文件，2）将需要运行的程序以及输入数据集复制到挂在的gem5镜像文件中，3）卸载之前挂载的gem5镜像文件，4）启动gem5模拟器并进入到启动好的模拟器中，5）在模拟器中编译测试程序，6）在模拟器中运行之前编译好的测试程序，7）测试程序运行完成后，关闭gem5模拟器，并进入到相应文件夹中获取测试结果。

### 挂载gem5镜像

首先需要挂载gem5镜像，将需要运行的小矮人程序以及输入数据集传入到gem5镜像中。使用命令为：

mount -o,loop,offset=32256 /disk\_path/disks/ aarch32-ubunt-natty-headless.img /mount\_path

其中“/disk\_path/disks/aarch32-ubunt-natty-headless.img”为镜像路径及镜像名称。

### 传入测试程序及数据集

挂载好gem5镜像后，可以使用如下命令查看镜像中的文件，并可以使用cp命令将小矮人程序及输入数据集传入到gem5镜像中。

cp /dwarf\_path /mount\_path

### 卸载镜像

当需要使用镜像时，需要先执行卸载操作，命令如下：

umount /mount\_path

### 启动gem5

将小矮人程序以及输入数据传入到镜像中后，就可以启动gem5模拟器，使用命令如下：

./build/ARM/gem5.opt ./configs/example/fs.py --disk-image=/disk\_path/disks/aarch32-ubuntu-natty-headless.img

当执行上述命令时，需要在gem5的根目录中执行，其中“/disk\_path/disks/aarch32-ubunt-natty-headless.img”为镜像的路径以及镜像名称。

执行上述命令启动gem5后，重新打开一个命令行窗口，进入gem5安装目录下的util/term/中执行：

./m5term localhost 3456

进入gem5中。

### 编译小矮人程序

以编译sort程序为例。进入到sort/src目录中，执行make all命令后就开始编译sort程序，因为在gem5中模拟的是arm环境，所以必须对测试程序进行重新编译。在编译完成并且没有报错的情况下就可以运行测试程序了。

其余负载测试程序编译过程和sort程序一样，在此不做赘述。

### 运行小矮人程序

1）运行sort程序。进入到src/omp/sort-openmp/run目录中，执行如下命令：

./multi\_thread 3 ../dataset/test.data.01 ./output/test.data.01

./single\_thread ../dataset/test.data.01 ./output/test.data.01

其中“multi\_thread”为多线程，“3”为线程数，“../dataset/test.data.01”为测试数据集，“./output/test.data.01”为输出结果。

2）运行wordcount程序的过程和sort程序一样。

3）运行md5程序的过程和sort程序一样。

4）运行union程序的步骤为：进入到run目录中，执行如下命令：

./multi\_thread ../dataset/test.data.01 ../dataset/test.data.02 ./output 3

./single\_thread ../dataset/test.data.01 ../dataset/test.data.02 ./output

其中“multi\_thread”为多线程，“3”为线程数，“../dataset/test.data.01”为测试数据集1，“../dataset/test.data.02”为测试数据集2，“./output/test.data.01”为输出结果。

5）运行bfs程序的步骤为：首先生成输入数据集，使用命令如下：

cd ../../data\_gen/BigDataGeneratorSuite/Graph\_datagen/

vim gen\_unweighted\_and\_split.sh

./gen\_unweighted\_and\_split.sh

之后将生成好的输入数据放到bfs/dataset/文件夹中。之后运行如下命令：

make my\_make\_edgelist

./run\_my\_make\_edgelist.sh ./dataset/gen\_graph\_10\_2640.txt

准备好输入数据集之后就可以运行负载测试了，运行负载测试的命令如下：

./multi\_thread 10 3 ../dataset/g.10\_2640.dump 4

./single\_thread 10 3 ../dataset/g.10\_2640.dump

其中“multi\_thread”为多线程，“4”为线程数，“../dataset/g.10\_2640.dump”为测试数据集。

6）运行matrixMult程序。进入到matrixMult/src/run目录中，执行如下命令：

./multi\_thread ../dataset/part-00000 ../dataset/part-00000 ./output/out.s.mtx 3

./single\_thread ../dataset/part-00000 ../dataset/part-00000 ./output/out.m.mtx

其中“multi\_thread”为多线程，“3”为线程数，“../dataset/part-00000”为测试数据集，“./output/out.m.mtx”为输出结果。

7）运行fft程序的方法如下：

因为fft程序需要用到fftw库，所以要先安装fftw库。首先需要下载fftw库，下载地址为：http://fftw.org/fftw-3.3.4.tar.gz。之后解压缩，进入到文件夹中。因为在模拟器中编译fftw库特别慢，所以需要使用交叉编译工具在物理平台上编译ARM下的fftw库。

交叉编译方式：

首先下载arm-linux-gcc-4.4.3.tar安装包，下载地址为：“http://arm9download.cncncn.com/mini2440/linux/arm-linux-gcc-4.4.3-20100728.tar.gz”，并解压到相应的目录中。解压完成之后需要配置环境变量，将交叉编译工具的安装目录配置到环境变量中。

export PATH＝$PATH:/usr/local/arm/4.4.3/bin

其中“/usr/local/arm/4.4.3”为交叉编译工具的安装目录(此目录可修改)。

在导入环境变量之后执行命令：arm-linux-gcc –v当看到出现版本信息并且没有报错时，说明交叉编译工具安装成功。

执行命令编译arm-fftw3：

./configure --prefix=/mnt/sdi/gwl/gem5/dwarf\_exp/dependency/ARM-fftw/ --host=arm-unknown-linux-gnueabi --target=arm-linux-gnueabi CC=/usr/local/arm/4.4.3/bin/arm-none-linux-gnueabi-gcc --enable-openmp

make

make install

编译后将inlude/和lib/中关于fftw库的相关文件复制到fft程序根目录中，在Makefile中加入Include\_m = -I./ -I../../my\_template\_arr/ **-I fftw3/include/ -L fftw3/lib**，之后就可以进入gem5中使用make all进行编译了。运行测试程序的命令如下：

./multi\_thread ../dataset/part-00000 ./output/out.s.mtx 3

./single\_thread ../dataset/part-00000 ./output/out.m.mtx

其中“multi\_thread”为多线程，“3”为线程数，“../dataset/part-00000”为测试数据集。

### 获取运行结果

当执行完需要执行的小矮人程序后，可以在m5out/stats.txt文件中就可以获取到运行程序时的全部运行状态。

## 在gem5 se模式下运行小矮人程序

在gem5 se模式下运行需要的步骤为：1）在当前操作系统中安装交叉编译工具，2）安装成功交叉编译工具后使用交叉编译工具重新编译小矮人程序，3）在gem5 se模式下运行小矮人程序，4）小矮人程序运行完成后进入到相应文件夹下获取运行结果。

### 安装交叉编译工具

因为我们需要在arm环境中运行小矮人程序，所以需要对小矮人程序进行重新编译，首先需要安装交叉编译工具。

首先下载arm-linux-gcc-4.4.3.tar安装包，下载地址为：“http://arm9download.cncncn.com/mini2440/linux/arm-linux-gcc-4.4.3-20100728.tar.gz”，并解压到相应的目录中。解压完成之后需要配置环境变量，将交叉编译工具的安装目录配置到环境变量中。

export PATH＝$PATH:/usr/local/arm/4.4.3/bin

其中“/usr/local/arm/4.4.3”为交叉编译工具的安装目录。

在导入环境变量之后执行命令：arm-linux-gcc –v当看到出现版本信息并且没有报错时，说明交叉编译工具安装成功。

### 交叉编译小矮人程序

以编译sort程序为例。进入到sort/src目录中，编辑MakeFile文件，将其中的gcc命令替换为arm-linux-gcc命令，同时因为在se模式下只能运行静态编译的程序，所以还需要将编译参数中的-c后加入-static参数实现静态编译。在编译过程中如果提示缺少某个库文件可以直接去搜索下载放入到相应位置重新编译即可。

### 运行小矮人程序

进入到gem5的根目录中，执行命令：./build/ARM/gem5.opt configs/example/se.py -c sort/src/omp/sort/./multi\_thread --options="../dataset/test.data.01 ./output/test.data.01"来运行程序。

其中“sort/src/omp/sort/./multi\_thread”为需要执行的程序的路径以及程序名称。因为执行负载测试的程序需要输入参数，可以使用options=""命令输入参数，例如指定输入路径可以使用options="/wikidata"命令进行指定。

运行其他小矮人程序只需要将程序的路径替换成其他小矮人程序的路径，将options参数替换成在fs模式下运行负载测试的参数就可以了。

sort运行命令：./build/ARM/gem5.opt configs/example/se.py –c /mnt/sdi/gwl/gem5/dwarf\_exp/dwarf/sort-openmp/src/omp/sort-openmp/run/single\_thread --option="/mnt/sdi/gwl/gem5/dwarf\_exp/dwarf/sort-openmp/src/omp/sort-openmp/dataset/test.data.01 /mnt/sdi/gwl/gem5/dwarf\_exp/dwarf/sort-openmp/src/omp/sort-openmp/dataset/run/output/out"

md5运行命令：/mnt/sdi/gwl/gem5/dwarf\_exp/dwarf/md5-openmp/src/omp/md5-openmp/run/single\_thread --option="/mnt/sdi/gwl/gem5/dwarf\_exp/dwarf/md5-openmp/src/omp/md5-openmp/dataset /mnt/sdi/gwl/gem5/dwarf\_exp/dwarf/md5-openmp/src/omp/md5-openmp/dataset/run/output/out"

wordcount运行命令：/mnt/sdi/gwl/gem5/dwarf\_exp/dwarf/wordcount-openmp/src/omp/ wordcount-openmp/run/single\_thread --option="/mnt/sdi/gwl/gem5/dwarf\_exp/dwarf/wordcount-openmp/src/omp/wordcount-openmp/dataset /mnt/sdi/gwl/gem5/dwarf\_exp/dwarf/wordcount-openmp/src/omp/wordcount-openmp/dataset/run/output/out"

union运行命令：/mnt/sdi/gwl/gem5/dwarf\_exp/dwarf/union-openmp/src/omp/union-openmp/run/single\_thread --option="/mnt/sdi/gwl/gem5/dwarf\_exp/dwarf/union-openmp/src/omp/union-openmp/dataset/test.data.01 /mnt/sdi/gwl/gem5/dwarf\_exp/dwarf/union-openmp/src/omp/union-openmp/dataset/test.data.02 /mnt/sdi/gwl/gem5/dwarf\_exp/dwarf/union-openmp/src/omp/union-openmp/dataset/run/output/out"

bfs运行命令：/mnt/sdi/gwl/gem5/dwarf\_exp/dwarf/bfs-openmp/src/omp/bfs-openmp/run/single\_thread --option="/mnt/sdi/gwl/gem5/dwarf\_exp/dwarf/bfs-openmp/src/omp/bfs-openmp/dataset/g.10\_2640.dump /mnt/sdi/gwl/gem5/dwarf\_exp/dwarf/bfs-openmp/src/omp/bfs-openmp/dataset/run/output/out"

matrixMult执行命令：./build/ARM/gem5.opt configs/example/se.py -c /mnt/sdi/gwl/gem5/dwarf\_exp/dwarf/matrixMult/src/omp/mtx\_mul/run/single\_thread --options="/mnt/sdi/gwl/gem5/dwarf\_exp/dwarf/matrixMult/src/omp/mtx\_mul/dataset/matrix /mnt/sdi/gwl/gem5/dwarf\_exp/dwarf/matrixMult/src/omp/mtx\_mul/dataset/matrix /mnt/sdi/gwl/gem5/dwarf\_exp/dwarf/matrixMult/src/omp/mtx\_mul/run/output/out.m.mtx"

fft执行命令：./build/ARM/gem5.opt configs/example/se.py -c /mnt/sdi/gwl/gem5/dwarf\_exp/dwarf/matrixMult/src/omp/mtx\_mul/run/single\_thread --options="/mnt/sdi/gwl/gem5/dwarf\_exp/dwarf/matrixMult/src/omp/mtx\_mul/dataset/matrix /mnt/sdi/gwl/gem5/dwarf\_exp/dwarf/matrixMult/src/omp/mtx\_mul/dataset/matrix /mnt/sdi/gwl/gem5/dwarf\_exp/dwarf/matrixMult/src/omp/mtx\_mul/run/output/out.m.mtx "

在提供的七个负载中，除了sort se模式运行报错之外，其他六个负载均可在gem5 se模式下成功运行。

### 获取运行结果

当执行完需要执行的小矮人程序后，可以在m5out/stats.txt文件中就可以获取到运行程序时的全部运行状态。