Workshop for simpoint

汇报人：薛春海

1

Simpoint介绍

为什么要做Simpoint?

- DSA时代，快速设计出一款特定领域的SoC，完成复杂的软件栈开发和硬件验证，并快速推向市场，**敏捷开发**是各类AI芯片公司追求的目标，也是目前大家遇到的最大挑战
- Gem5是一款时钟精确仿真的SOC模拟器，精确的周期模拟需要机器**花费数周数月时间，时间成本过高**

为什么要做Simpoint?

- Simpoint在**指令数方面做了精简**，能做到**几十到几百倍的加速**
- Simpoint加速仿真能取得一个**较高的仿真精度，通常误差在5%以内**
- 减少了全面分析整个程序执行所需的硬件资源和时间成本，**提高开发效率**

香山敏捷开发框架

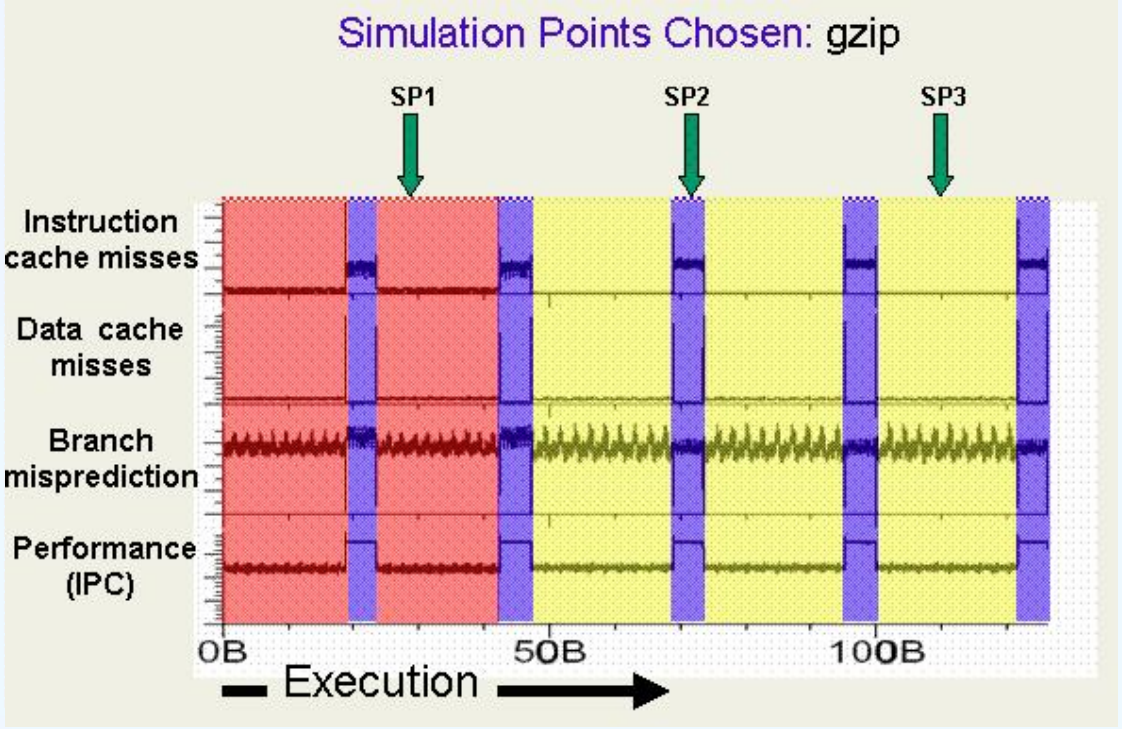
由中国科学院计算所牵头发起 **“香山” 处理器**项目，探索并建立了一套高性能处理器的**敏捷开发流程，其中就用到了**simpoint。

- 敏捷开发框架
  - 使用说明：https://xiangshan-doc.readthedocs.io/zh_CN/latest/tools/xsenv/
  - simpoint工具介绍：https://xiangshan-doc.readthedocs.io/zh-cn/latest/tools/simpoint/
  - Troubleshooting：https://xiangshan-doc.readthedocs.io/zh_CN/latest/tools/troubleshoot/

加速原理

不管多复杂的一个程序,它在绝大多数时间做的执行的操作就那么几个,如果我们能**统计出那几个主要的操作以及它们所占的比例,是不是只对那几个操作做仿真再加权**就可以了呢?

SimPoint 使用**静态分析对运行时所执行的指令流分段**，然后把各段**聚合**，SimPoint认为一个聚类里的片段是相似的，并确定聚类中的**热点代码段（即SimPoint）及聚类所占的权重**。



仅模拟三个挑选的模拟点，SimPoint 能够显著缩短仿真时间

运行机制

simpoint首先把程序运行时**所执行的指令流切段**，片段长度一般50M~200M，称为Interval。然后对所有的片段**提取BBV**(Basic Block Vector) **表征程序片段的相似度向量(即生成的simpoint.bb.gz文件)**，再**对这些向量做k-means聚类**，把相似的段归到一类，便可以区分出k类的程序片段，然后计算出不同类型片段所占的比例，称为**权重**(weight)。

在每个聚类中选取一个指令段做代表，找出各个**聚类中心的程序片段作为checkpoint**，使用GEM5仿真的时候**只执行选取出的指令段运行**，然后根据每个聚类的大小对执行的指令段**加权重，再加到一起就是原程序的跑分结果**。

基于检查点的采样 (典型： SimPoint)

- 选择性、带权重采样
- 每个采样点较大 (50M~200M)

2

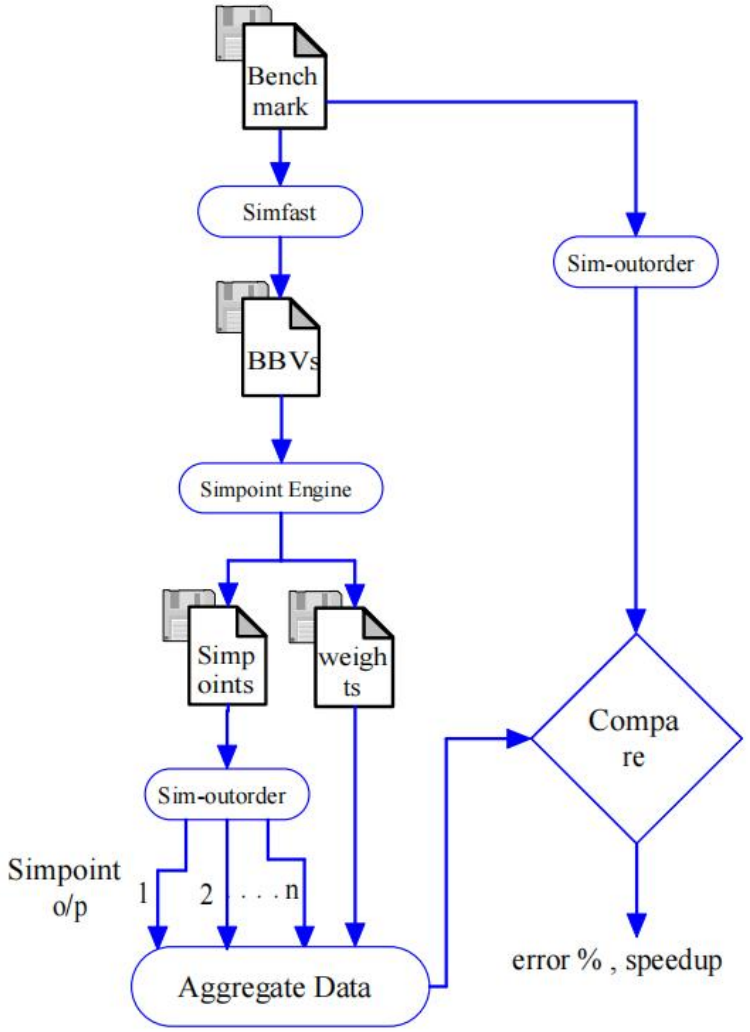
simpoint仿真结果

统计数据

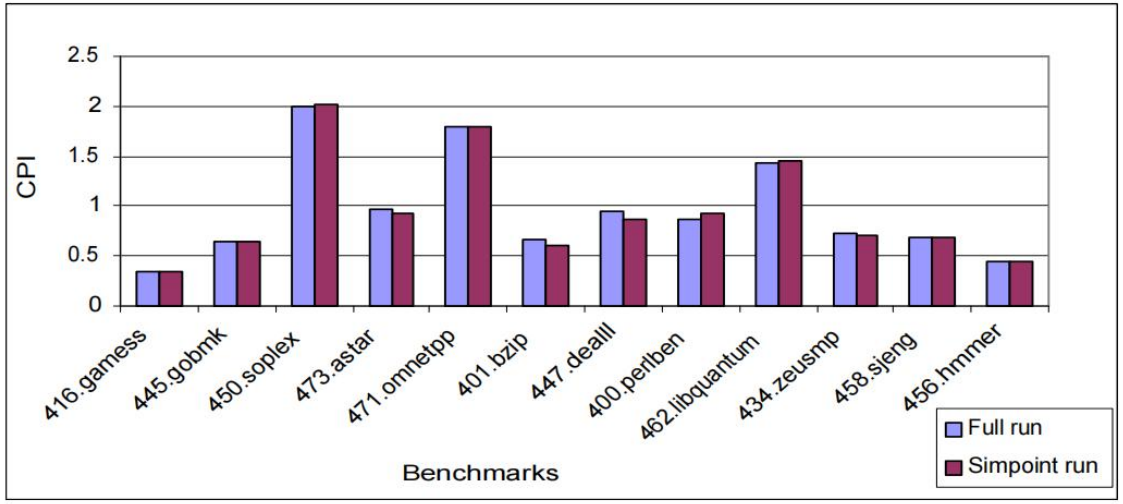
- 对用simpoints加速得出的数据和全程仿真的数据进行比较
- 得出误差百分比error%和加速比sppedup
- 得出缓存缺失率和CPI等

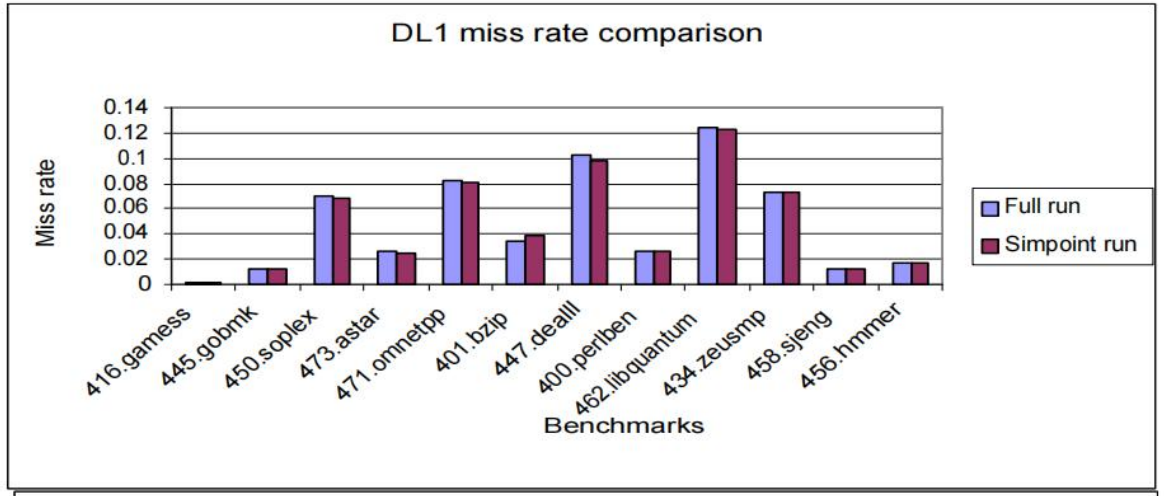
$$MissRatio = \frac{\sum_{i=0}^{n}(misses_i * weight_i)}{\sum_{i=0}^{n}(lookups_i * weight_i)}$$

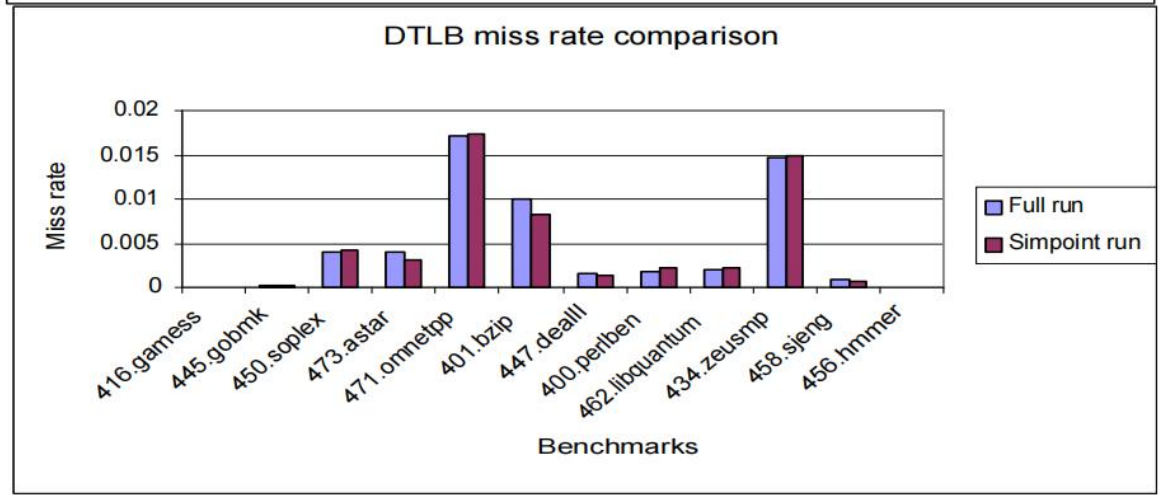
$$CPI = \sum_{i=0}^{n}(CPI_i * weight_i)$$



仿真精度







- simpoint运行的最终聚合度量是使用前一节中提到的公式计算的，利用生成的checkpoint进行重载并加权计算。
- 对仿真结果进行了比较，得到了仿真点的精度，通常误差精度在5%以内。

生成模拟点所需时间

SPECINT Benchmark	No. of Sim-points	Total No. of Instructions (in Billions)	Simulation time	SPECFP Benchmark	No. of Sim-points	Total No. of Instructions (in Billions)	Simulation time
401.bzip	21	213.9	15 hrs	410.bwaves	14	2317.9	7 days
445.gobmk	19	430.7	22 hrs	435.gromacs	20	2387	5 days 17 hrs
456.hmmer	9	2593.1	7 days	437.leslie3d	24	2609.3	7 days 3 hrs
458.sjeng	12	3187.7	9 days	444.namd	21	2223.8	6 days
462.libquantum	20	1989	5 days 13 hrs	447.dealII	3	3.7	13 minutes
464.h264ref	14	5663	8 days 6 hrs	450.soplex	21	486.4	1 days 10 hrs
471.omnetpp	11	729.9	2 days16 hrs	482.sphinx3	21	4004.8	9 days 22 hrs
473.astar	9	966.5	3 days 17 hrs	459.GemsF	25	223.6	17 hrs
400.perlbench	14	184.5	2.6 hrs	434.zeusmp	26	1992	1 day
				433.milc	21	1222.4	15 hrs
				436.cactusA	9	4560.1	14 days 7 hrs

不同benchmark进行采样并生成weights和checkpoints所花时间

simpoints加速比

- 对几个benchmark进行全程仿真和重载reload
- 并对仿真时间进行了比较，得到了仿真点的加速比

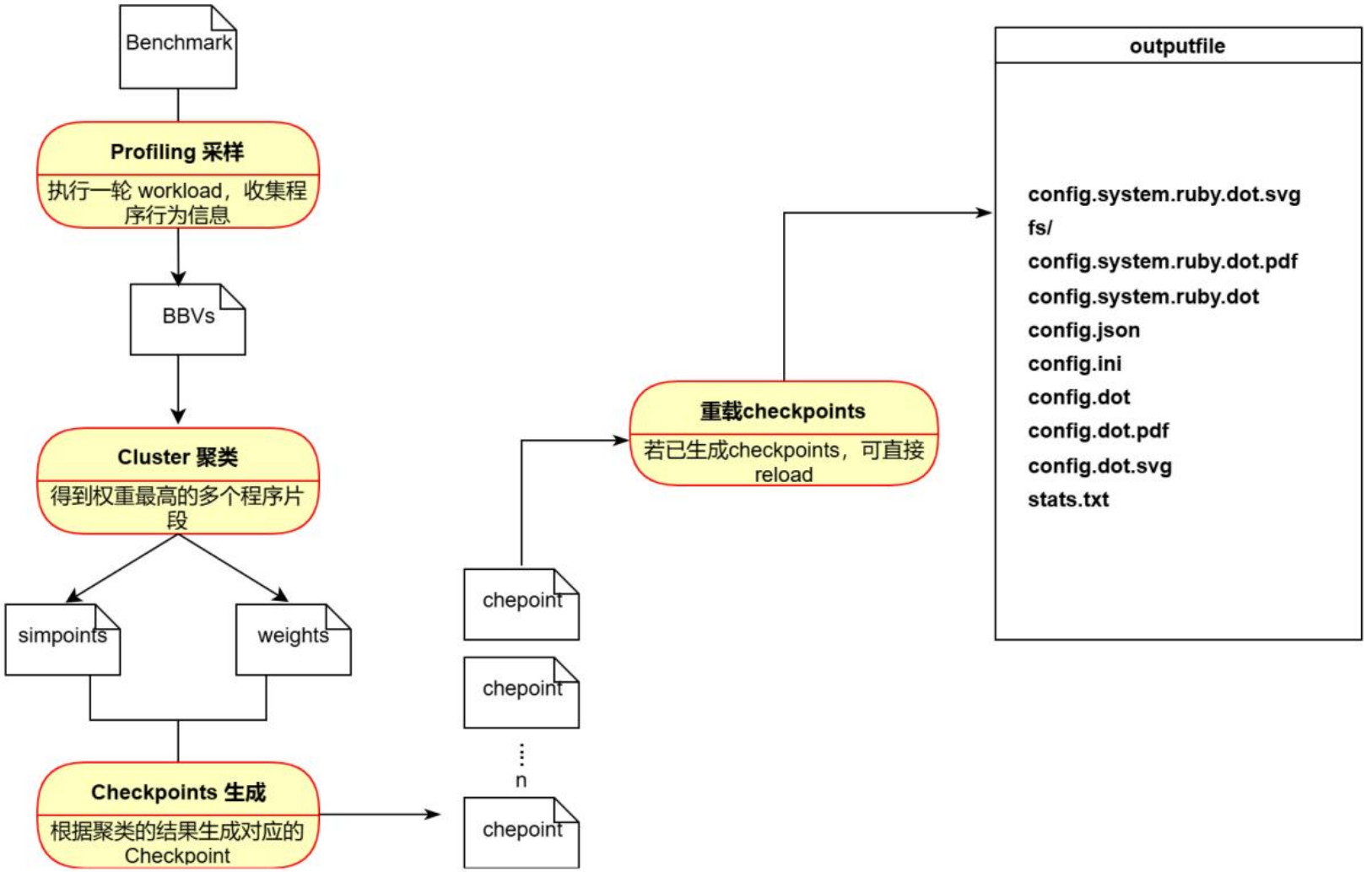
Benchmark	Simpoint run	Full run	Speedup
445.gobmk	5.53 hrs	34 days	148
450.soplex	3.62 hrs	77 days	502
473.astar	11 hrs	78 days	171
471.omnetpp	10.21 hrs	91 days	213
401.bzip	5 hrs	11 days	53.3
447.dealII	2.6 mins	1.9 hrs	42.6
400.perlbench	3.7 hrs	4 days 13 hrs	29.6
462.libquantum	19 hrs	42 days	53

不同benchmark进行重载relaod与全程仿真所花时间对比

3

总体架构

Simpoint整体架构



环境配置

环境配置：

- Ubentu 20.04
- python3.9.7(与python版本关系不大，3.6以上就可以)
- Simpoint3.2(官网的直接编译行不通，很多报错)
- gem5 v20.1.0.4 (如使用gem5 v20版本，最好用ubentu20.04)
- benchmark：graph500-2.1.4 或 dhrystone
- 交叉编译工具 aarch64-linux-gnu-gcc

4

Profiling 采样

Profiling 采样

gem5 运行文件夹中**生成一个块向量文件（simpoint.bb.gz）**

相当于正常运行一遍程序（但因为是NonCachingSimpleCPU，不需要像O3一样需要处理cache，比**直接用gem5仿真会快很多**）

 # 注：gem5 只有 atomic cpu 支持生成BBV
否则会报错 `fatal: SimPoint/BPProbe should be done with an atomic cpu`

实际命令：

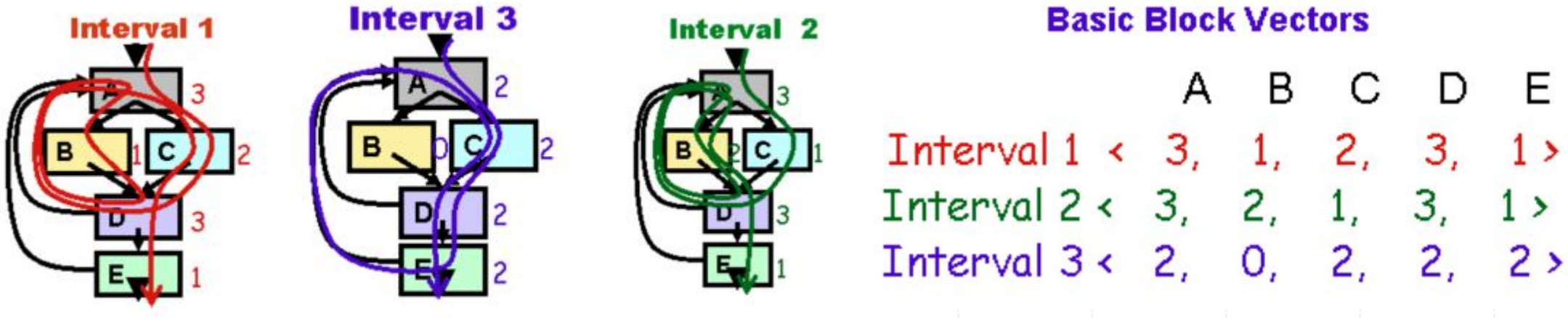
```
build/ARM/gem5.opt configs/example/se.py \

--cpu-type=NonCachingSimpleCPU \

-c /home/chun/graph/graph500-2.1.4/seq-csr/seq-csr \

--simpoint-profile --simpoint-interval 10000000 \

-o ' -s 13 -e 12' \
```

Profiling 采样

程序的切段和统计是在运行时完成的，所以要得到结果需要**至少运行一次要跑的程序**，并实时的进行统计，收集程序行为信息。

统计这段时间内，每个basic block执行的次数，再将每个basic block出现的次数和block中包含的指令条数相乘就得到这个basic block的对应的值，所有这些值组成一个向量。

假设程序运行时包含3个100Million条指令组成的指令流，就会有3个向量，统称为**basic block vector**(即生成的向量文件simpoint.bb.gz)。



Basic block vector（BBV）

simpoint.bb.gz文件格式：T[:BB_id:count]

每个T对应一个interval，BB_id为basic block的编号；
count为对应的basic block的值，值的计算方式为在此interval中对应的block出现的次数和block中指令的个数的乘积。

```
simpoint.bb.gz
stats.txt
root@ubpl:/home/chun/gem5_opt/m5out#
```

```
T:194:2 :218:7 :219:4 :220:4 :221:2 :229:15 :230:5 :231:2 :232:4 :233:2 :239:2 :240:18 :241:6 :242:7 :243:5 :244:2 :280:16 :29
12:8 :513:2 :514:7 :637:11 :638:14 :643:3 :644:5 :645:11 :658:18 :659:10 :660:53 :661:117 :662:51 :663:1457 :664:2 :665:15 :66
76:18
T:194:2 :218:7 :219:4 :220:4 :221:2 :229:15 :230:5 :231:2 :232:4 :233:2 :239:2 :240:18 :241:6 :242:7 :243:5 :244:2 :280:16 :29
12:8 :513:2 :514:7 :637:11 :638:14 :643:3 :644:5 :645:11 :658:18 :659:10 :660:53 :661:117 :662:51 :663:1457 :664:2 :665:15 :66
77:12
T:666:120 :667:90 :668:40 :669:960 :670:40 :671:40 :672:40 :673:6 :674:9 :678:8 :679:1 :680:4 :681:1 :682:630 :683:42 :684:209
164 :694:861 :695:164 :696:164 :697:123 :698:205 :699:243 :700:164 :701:246 :702:164 :703:246 :704:40 :705:168
T:682:720 :683:48 :684:241 :685:2892 :686:1687 :687:336 :688:96 :689:144 :690:82 :691:123 :692:192 :693:196 :694:1008 :695:192
:28 :705:192
T:682:720 :683:48 :684:240 :685:2880 :686:1680 :687:336 :688:96 :689:144 :690:68 :691:102 :692:192 :693:192 :694:1029 :695:196
:56 :705:108
T:682:735 :683:49 :684:241 :685:2892 :686:1680 :687:336 :688:96 :689:144 :690:76 :691:114 :692:192 :693:192 :694:1008 :695:192
:40 :705:132
```

5

聚类

Cluster 聚类

参数：

-maxK　N　表示K的取值从1到N，然后用BIC选择最合适的k值。

-saveSimpoints　将最终选择的模拟点信息放到文件

-saveSimpointWeights　将选择的每个模拟点的权重放到文件

-loadFVFile 载入Frequence vector文件，就是前面生成的BBV文件

实际命令：

```
./simpoint -maxK 30 -numInitSeeds 1 -loadFVFile

/home/chun/gem5_opt/m5out/simpoint.bb.gz -inputVectorsGzipped

-saveSimpoints /home/chun/SimPoint.3.2/output/graph500.simpoints

-saveSimpointWeights /home/chun/SimPoint.3.2/output/graph500.weights
```

聚类

BBV中每条记录对应一个interval中每个basic block的统计值，如果有两条记录中每个block的值完全相同。

那**跑这两个interval的指令流和跑其中一个interval的指令流两次**，在统计概念上来说**效果是一样**的，聚类也是依赖于此。

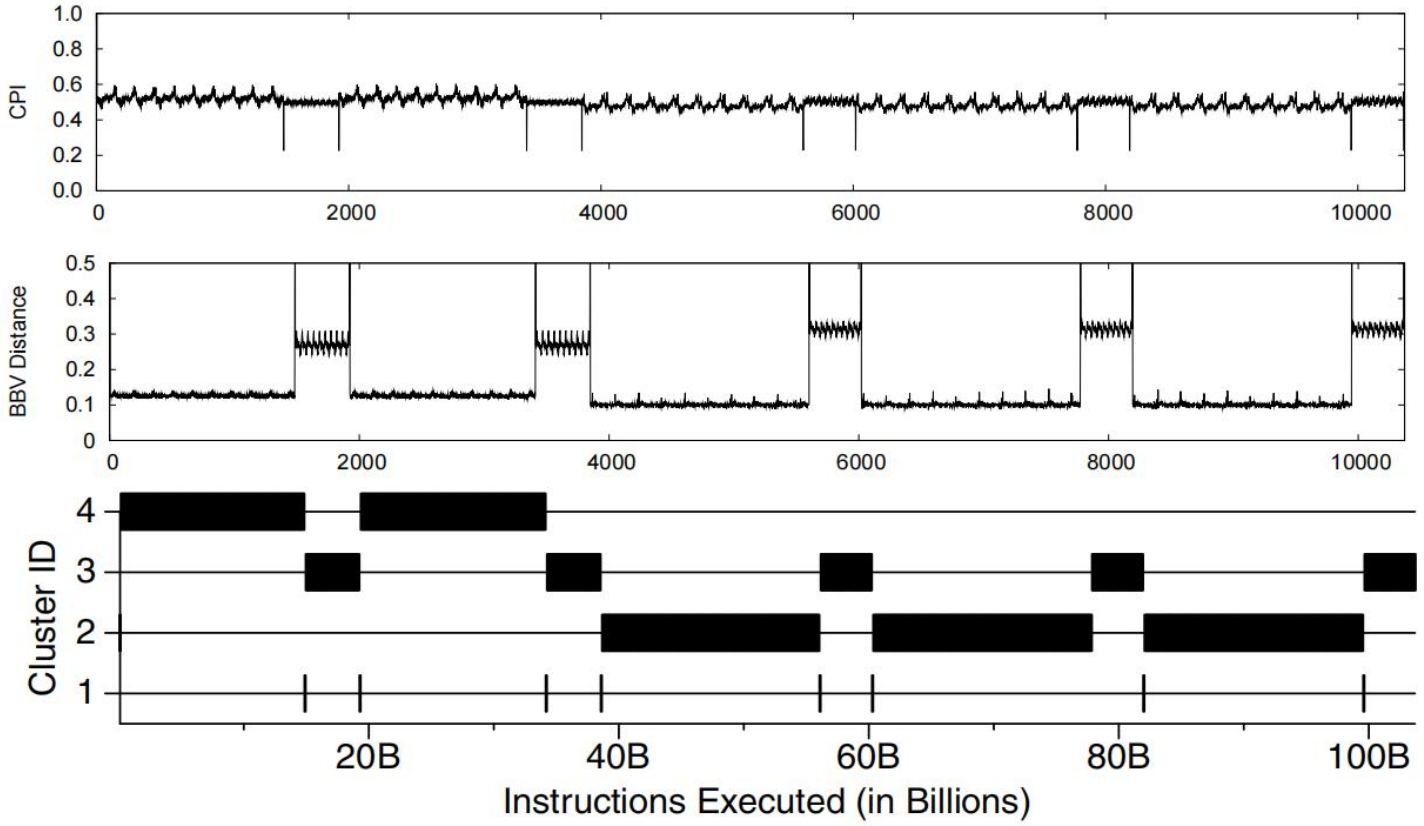
但要做到两个interval完全一样的可能性比较低，做个退让，**把“相似”的interval也归为一类**。

论文选择使用欧几里得距离或曼哈顿距离来度量interval的相似度。

$$EuclideanDist(a,b) = \sqrt{\sum_{i=1}^{D}(a_i - b_i)^2}$$

$$ManhattanDist(a,b) = \sum_{i=1}^{D}|a_i - b_i|$$

Cluster 聚类



上图为gzip-graphic程序执行期间的**哪些interval被划分为不同的聚类**，这些聚类由k-means聚类算法确定，**完整运行被划分为 4 个Cluster**

k-means聚类算法

哪些interval应该聚为一类呢？Simpoint选择经典的k-means算法

k-means算法还有个缺点就是需要提前确定k的值，**k表示最后需要聚合成几个类**

simpoint中用的方法是按某种规则选择多个k值，分别计算得到聚类的结果后使用**BIC算法进行打分，打分最高的k值作为最终的选择**并作为**最优的聚类数量**

Simpoint中的k-means聚类算法实现：**输入为simpoint.bb.gz文件**，会输出两个文件：**simpoints文件和weights文件**(这里simpoint工具只会生成各段指令的起始位置和权重)

确定最优聚类数量

K-Means 聚类算法的多次运行结果：

Run number 6 of at most 7, k = 20：最多 7 次运行中的第 6 次，本次运行的聚类数量 k 为 13

进行了 1 次 K-Means 迭代
BIC 分数为 6143.48

失真（Distortion）为 0.979843 ，每个聚类的失真
方差（Variance）为 0.0056967 ，每个聚类的方差

类似地，后面的 Run number 7（k = 12）也给出了相应的运行信息，包括初始化种子、迭代次数、BIC 分数、失真、方差等

最后提到，对于 BIC 阈值，**最佳的聚类是运行 6（k = 13），k值打分最高**

```
Run number 6 of at most 7, k = 13
---------------------------------------------------------------
  ---------------------------------------------------------------
  Initialization seed trial #1 of 1; initialization seed = 493575231
  ---------------------------------------------------------------
  Initialized k-means centers using random sampling: 13 centers
  Number of k-means iterations performed: 10
  BIC score: 6143.48
  Distortion: 0.979843
  Distortions/cluster: 0.395747 0.0961579 0.100246 8.1852e-34 0.104838 1.20371e-35 0.170939 7.1059e-07 9.91
  Variance: 0.00569676
  Variances/cluster: 0.0659578 0.00246559 0.00911326 0 0.00953077 0 0.0284898 3.55295e-08 9.014e-08 0.00787
  The best initialization seed trial was #1

---------------------------------------------------------------
Run number 7 of at most 7, k = 12
---------------------------------------------------------------
  ---------------------------------------------------------------
  Initialization seed trial #1 of 1; initialization seed = 493575232
  ---------------------------------------------------------------
  Initialized k-means centers using random sampling: 12 centers
  Number of k-means iterations performed: 3
  BIC score: 5461.86
  Distortion: 1.67769
  Distortions/cluster: 0.00100842 0.027229 0.0128228 0.0168647 0.00475209 0.00459791 1.79352e-33 0.00095185
  Variance: 0.00969761
  Variances/cluster: 0.000201684 0.000825122 0.000400714 0.00281078 0.000950417 0.000919583 0 0.000190372 0
  The best initialization seed trial was #1

---------------------------------------------------------------
---------------------------------------------------------------
Post-processing runs
---------------------------------------------------------------

  For the BIC threshold, the best clustering was run 6 (k = 13)
  Post-processing run 6 (k = 13)
    Saving simpoints of all non-empty clusters to file '/home/chun/SimPoint.3.2/output/graph500.simpoints'
    Saving weights of all non-empty clusters to file '/home/chun/SimPoint.3.2/output/graph500.weights'
chun@ubpl:~/SimPoint.3.2/bin$
```

simpoints和weights文件

k值=13，最终聚合成13个类

`61 0` 是指第 0 个 cluster 的聚类中心是第 61 次 sample,

即第 61*interval 条指令处，起始指令为 61*interval ，权重为 0.0378378

```
chun@ubpl:~/SimPoint.3.2/output$ cat graph500.simpoints
61 0
159 1
171 2
0 3
170 4
54 5
175 6
12 7
8 8
155 9
55 10
50 11
65 12
chun@ubpl:~/SimPoint.3.2/output$ cat graph500.weights
0.0378378 0
0.216216 1
0.0648649 2
0.00540541 3
0.0648649 4
0.00540541 5
0.0378378 6
0.113514 7
0.0648649 8
0.0648649 9
0.00540541 10
0.108108 11
0.210811 12
```

6

生成 Checkpoints

生成checkpoints

获取 Checkpoints，使用以下命令：

```
% build/ARM/gem5.opt <base options>
configs/example/fs.py
--take-simpoint-checkpoint=<simpoint file path>,
<weight file path>,
<interval length>,
<warmup length>
<rest of fs.py options>
```

<simpoint 文件>和<weight 文件>
由前面的聚类生成

 # 注： SimPoint 分析生成checkpoints应使用单个 AtomicSimpleCPU 配置运行
不支持多核模拟

实际命令：

```
root@ubpl:/home/chun/gem5_opt# build/ARM/gem5.opt configs/example/se.py \
> --take-simpoint-checkpoint=\
> /home/chun/SimPoint.3.2/output/graph500.simpoints,\
> /home/chun/SimPoint.3.2/output/graph500.weights,\
> 10000000,1000000 \
> --cpu-type=AtomicSimpleCPU \
> --mem-type DDR3_2133_8x8 --mem-size 16GB \
> --l2-hwp-type StridePrefetcher \
> --caches --l2cache --l1i_size 64kB \
> --l1d_size 32kB --l2_size 1MB \
> --l1i_assoc 8 --l1d_assoc 8 \
> --l2_assoc 16 --cacheline_size 128 \
> -c /home/chun/graph/graph500-2.1.4/seq-csr/seq-csr \
> -o ' -s 13 -e 12' \
>
gem5 Simulator System.  http://gem5.org
gem5 is copyrighted software; use the --copyright option for details.

gem5 version 20.1.0.4
```

生成checkpoints

拿到simpoints和weights文件后就可以知道各段interval的起始位置和权重了

可以使用GEM5快速forward到对应的interval，然后切换到GEM5的detailCPU模式开始运行

为了在后续仿真时跳过forward过程，加速仿真流程，GEM5提供了checkpoint功能

第一次运行时，forward到interval的起始点后使用checkpoint功能把快照保存起来，以后运行时直接从快照恢复速度快很多

```
simpoint analysis file: /home/chun/SimPoint.3.2/output/graph500.simpoints
simpoint weight file: /home/chun/SimPoint.3.2/output/graph500.weights
interval length: 10000000
warmup length: 1000000
0 0.00540541 0 0
8 0.0648649 79000000 1000000
12 0.113514 119000000 1000000
50 0.108108 499000000 1000000
54 0.00540541 539000000 1000000
55 0.00540541 549000000 1000000
61 0.0378378 609000000 1000000
65 0.210811 649000000 1000000
155 0.0648649 1549000000 1000000
159 0.216216 1589000000 1000000
170 0.0648649 1699000000 1000000
171 0.0648649 1709000000 1000000
175 0.0378378 1749000000 1000000
Total # of simpoints: 13
Global frequency set at 1000000000000 ticks per second
warn: DRAM device capacity (8192 Mbytes) does not match the address range ass
ned (16384 Mbytes)
0: system.remote_gdb: listening for remote gdb on port 7001
info: Entering event queue @ 0.  Starting simulation...
Writing checkpoint
```

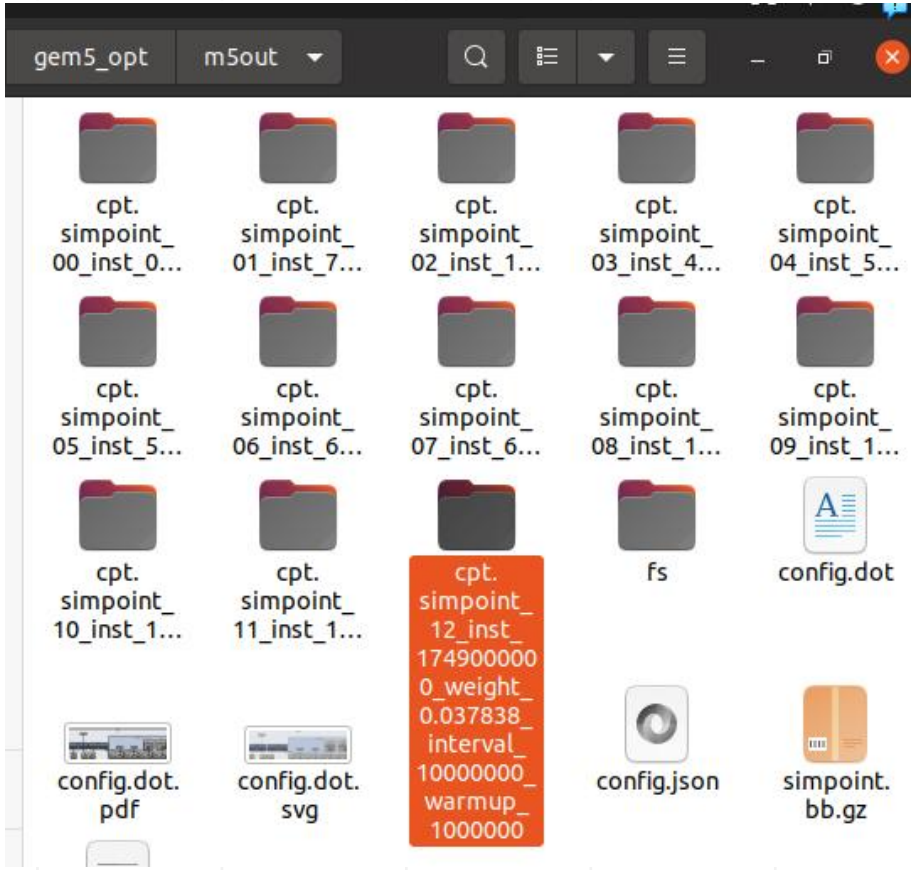
Checkpoints

checkpoint类似于上下文切换时保存的上下文，也类似于微架构中遇到分支指令时，保存的checkpoint

可以理解成是在**一个时刻CPU运行的快照，会保存架构寄存器和缓存memory的状态**。

输入为前面生成的 simpoints 和 weights 文件，输出生成了13个checkpoint文件

文件夹里面存储了给定时刻硬件的全部信息，比如cache配置，cpu类型等



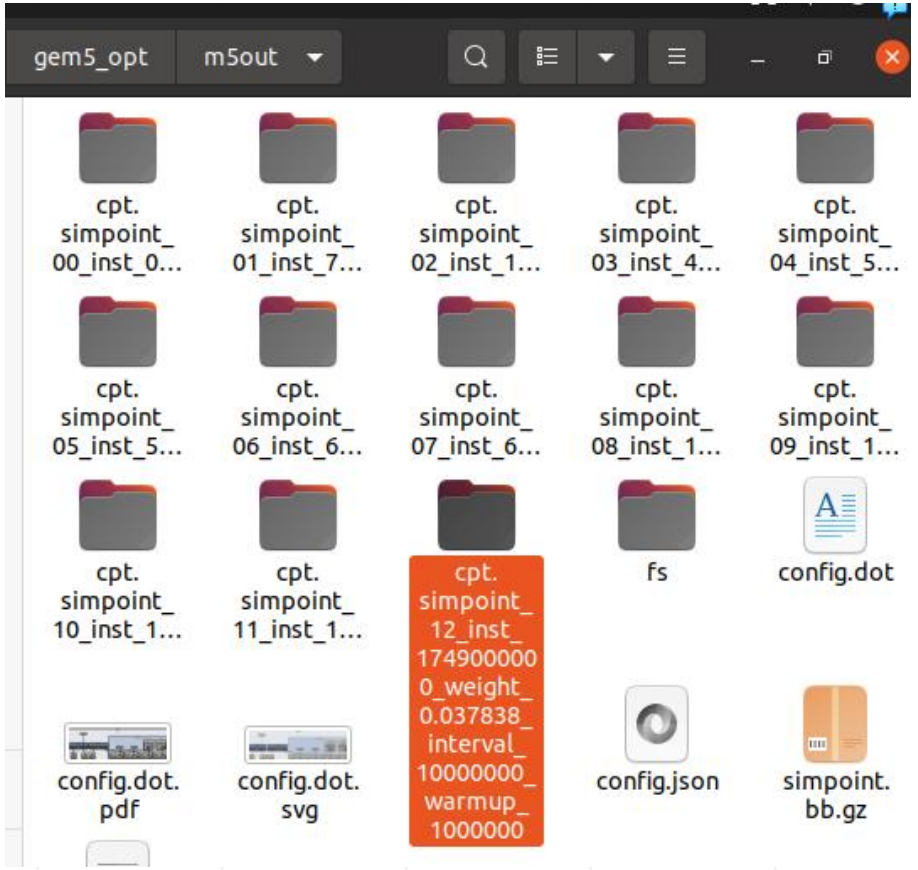
Checkpoints

文件目录如下：

```
cpt.simpoint_12_inst_1709000000_weight_0.064865_
int
erval_10000000_warmup_1000000
|    ├──── m5.cpt
|    └──── system.physmem.store0.pmem
```

m5.cpt即为当前**cpu运行状态的快照**，比如寄存器值，页表映射关系之类的
system.physmem.store0.pmem则是当时的内存的状态。

因此有了cpt和pmem，gem5就可以从任意一个checkpoint进行restore重载



7

reload

重载checkpoint

参数：

<base options> configs/example/fs.py
--restore-simpoint-checkpoint -r <N>
--checkpoint-dir <simpoint checkpoint path>
<rest of fs.py options>

<N> 是 （SimPoint 索引 + 1）。例如，“-r 1”将从 SimPoint #0 恢复。

实际命令：

```
root@ubpl:/home/chun/gem5_opt# build/ARM/gem5.opt configs/example/se.py \
> --cpu-type O3_ARM_v7a_3 \
> --cpu-clock 2.5GHz \
> --num-cpu 1 \
> --mem-type DDR3_2133_8x8 --mem-size 16GB \
> --l2-hwp-type StridePrefetcher \
> --caches --l2cache --l1i_size 64kB \
> --l1d_size 32kB --l2_size 1MB \
> --l1i_assoc 8 --l1d_assoc 8 \
> --l2_assoc 16 --cacheline_size 128 \
> -c /home/chun/graph/graph500-2.1.4/seq-csr/seq-csr \
> --restore-simpoint-checkpoint -r 13 \
> --checkpoint-dir /home/chun/gem5_opt/m5out
gem5 Simulator System.  http://gem5.org
gem5 is copyrighted software; use the --copyright option for details.

gem5 version 20.1.0.4
gem5 compiled Jul 18 2024 17:53:02
gem5 started Jul 27 2024 13:18:21
gem5 executing on ubpl, pid 160737
command line: build/ARM/gem5.opt configs/example/se.py --cpu-type O3_ARM_v7a_3 --cpu-clock 2.5GHz --nu
 64kB --l1d_size 32kB --l2_size 1MB --l1i_assoc 8 --l1d_assoc 8 --l2_assoc 16 --cacheline_size 128 -c
un/gem5_opt/m5out
```

reload重载

每次再运行程序的时候只需要重载 checkpoint，可以并行跑checkpoints，运行速度很快。

```
Resuming from /home/chun/gem5_opt/m5out/cpt.simpoint_12_inst_1749000000_weight_0.037838_interval_10000000
Resuming from SimPoint #12, start_inst:1749000000, weight:0.037838, interval:10000000, warmup:1000000
Global frequency set at 1000000000000 ticks per second
warn: DRAM device capacity (8192 Mbytes) does not match the address range assigned (16384 Mbytes)
0: system.remote_gdb: listening for remote gdb on port 7000
warn: Checkpoints for file descriptors currently do not work.
Switch at curTick count:10000
info: Entering event queue @ 907262769500.  Starting simulation...
Switched CPUS @ tick 907262779500
switching cpus
warn: PowerState: Already in the requested power state, request ignored
warn: User mode does not have SPSR
warn: User mode does not have SPSR
info: Entering event queue @ 907262779500.  Starting simulation...
Warmed up! Dumping and resetting stats!
info: Entering event queue @ 907614902000.  Starting simulation...
Done running SimPoint!
root@ubpl:/home/chun/gem5_opt#
```

合并仿真结果

右图为重载的输出

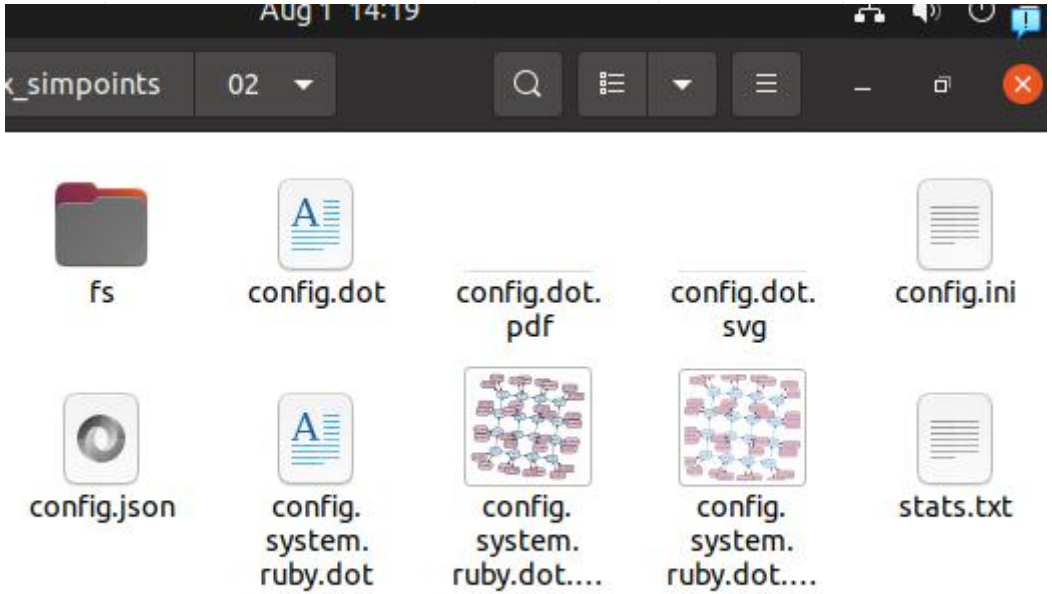
使用 SimPoint 的最后一步是**合并加权仿真点，以得出程序执行的总体性能估计**。

每个权重代表属于其阶段的总执行的比例，总体性能估计值是一组模拟点估计值的加权平均值。

例如，如果有 3 个模拟点，它们的权重为 [0.22， 0.33，0.45]，它们的 CPI 为 （CPI1， CPI2， CPI3）
则这些点的加权平均值为：

CPI = 0.22 * CPI1 + 0.33 * CPI2 + 0.45 * CPI3

加权平均计算出的 CPI 是程序完整执行的估计值，此计算方法可用于任何指标*



Simpoint结合Gem5部署

前置环境：

- 构建Gem5 v20.1.0.4
- 编译Simpoint 3.2
- 将benchmark编译成对应架构的可执行文件

1、使用Gem5分析benchmark生成BBVs

2、使用Simpoint 3.2 分析BBVs生成simpoints和weights

3、使用Gem5分析benchmark、simpoints、weights，生成checkpoints

4、使用Gem5重载checkpoints

8

Questions

benchmark的统计结果输出？

- 为什么没有正常仿真benchmark的统计结果输出(红框部分)？

  Simpoint是**离线分段分析程序**，并没有完整的从头到尾仿真执行benchmark

  故SimPoint也有缺陷：必须做离线分析，并且一个程序做一次。

```
Resuming from SimPoint #3, start_inst:10900, weight:0.093752, interval:1000, warmup:100
Global frequency set at 1000000000000 ticks per second
src/mem/dram_interface.cc:690: warn: DRAM device capacity (8192 Mbytes) does not match the address
src/base/statistics.hh:279: warn: One of the stats is a legacy stat. Legacy stat is a stat that doe
system.remote_gdb: Listening for connections on port 7000
src/sim/process.cc:396: warn: Checkpoints for pipes, device drivers and sockets do not work.
Switch at curTick count:10000
src/sim/simulate.cc:194: info: Entering event queue @ 6252000.  Starting simulation...
Switched CPUS @ tick 6262000
switching cpus
src/sim/power_state.cc:105: warn: PowerState: Already in the requested power state, request ignored
src/arch/arm/isa.hh:212: warn: User mode does not have SPSR
src/arch/arm/isa.hh:212: warn: User mode does not have SPSR
src/sim/simulate.cc:194: info: Entering event queue @ 6262000.  Starting simulation...
Warmed up! Dumping and resetting stats!
src/sim/simulate.cc:194: info: Entering event queue @ 6642000.  Starting simulation...
Done running SimPoint!
root@ubpl:/home/chun/gem5#
```

```
SCALE: 6
nvtx: 64
edgefactor: 5
terasize: 5.11999999999999969e-09
A: 5.69999999999999951e-01
B: 1.90000000000000002e-01
C: 1.90000000000000002e-01
D: 5.00000000000000444e-02
generation_time: 5.04374000000000033e-04
construction_time: 1.21261000000000003e-04
nbfs: 55
min_time: 6.31500000000000052e-06
firstquartile_time: 6.32200000000000063e-06
median_time: 6.32300000000000028e-06
thirdquartile_time: 6.32300000000000028e-06
max_time: 6.41400000000000076e-06
mean_time: 6.32520000000000002e-06
stddev_time: 1.25379424149260202e-08
min_nedge: 3.20000000000000000e+02
firstquartile_nedge: 3.20000000000000000e+02
median_nedge: 3.20000000000000000e+02
thirdquartile_nedge: 3.20000000000000000e+02
max_nedge: 3.20000000000000000e+02
mean_nedge: 3.20000000000000000e+02
stddev_nedge: 0.00000000000000000e+00
min_TEPS: 4.98908637355784178e+07
firstquartile_TEPS: 5.06088881859876588e+07
median_TEPS: 5.06088881859876588e+07
thirdquartile_TEPS: 5.06168933881682977e+07
max_TEPS: 5.06730007917656302e+07
harmonic_mean_TEPS: 5.05912856510466039e+07
harmonic_stddev_TEPS: 1.36468005630961106e+04
Exiting @ tick 2829510000 because exiting with last active thread contex
root@ubpl:/home/chun/gem5_opt# build/ARM/gem5.opt configs/example/se.py
```

可以使用别人的simpoints吗？

- 可以在社交网站上请求别人发SimPoint选好的点给他，使用别人的simpoints吗？

  这肯定是不行的，因为两边的Binary很可能不一样，受到程序版本、编译器版本、优化选项、库的版本的影响。A这边的第N个片段和B的第N个片段很可能不是同一个片段。

```
Total # of simpoints: 20
Global frequency set at 1000000000000 ticks per second
warn: DRAM device capacity (8192 Mbytes) does not match the address range assigned (16384 Mbytes)
mmap: Cannot allocate memory
fatal: Could not mmap 17179869184 bytes for range [0:0x400000000]!
Memory Usage: 127612 KBytes
```

THANKS