实验课程: 计算机体系结构

实验名称: 论文复现

专业名称: 计算机科学与技术

学生姓名: 吴臻

学生学号: 21307371

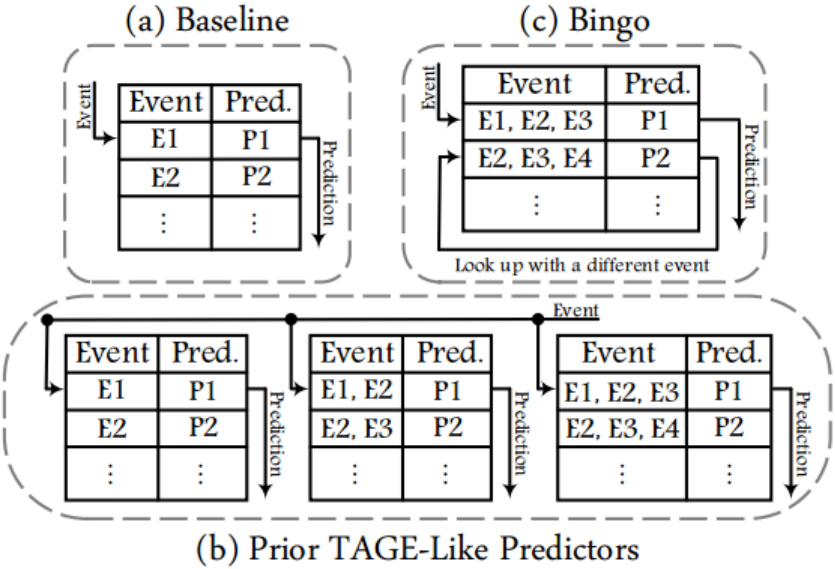
论文简介

论文标题：**Bingo Spatial Data Prefetcher**

> **随着现代大数据应用程序的兴起，庞大的数据集超出了有限容量的缓存，并存储在内存中。因此，执行这些工作负载的处理器经常遇到缓存未命中，导致核心停滞，造成显著的性能损失。为了减少缓存未命中的次数，空间数据预取器通过基于内存页面之间访问模式的相似性来预取未来的内存引用，这类预取器被广泛应用。本文提出了一种新的空间数据预取器——Bingo预取器，旨在提高准确性和预测机会。**

背景

- 现代大数据应用程序具有庞大的数据集，远远超出了缓存的容量，因此处理器经常遇到缓存未命中，导致性能下降。空间数据预取器通过预取未来的内存引用来减少缓存未命中的次数，从而提高性能。
- **空间数据预取技术**：这是一种**利用内存页面之间的访问模式相似性**来预测和获取未来内存引用的技术，可以减少缓存未命中的次数，提高大数据应用的性能。
- **现有技术**：**TAGE-like predictor**是一种基于事件的分支预测器，它使用多个元数据表（metadata table）记录不同类型的事件，如PC+addr（程序计数器+地址）和PC+offset（程序计数器+偏移量），以提高预测精度。



问题

- **核心问题**：大数据应用中的分支预测器（branch predictor）需要高精度和低存储开销，但现有的技术难以同时满足这两个要求。
- **现有空间数据预取器的局限性**：现有的空间数据预取器存在**准确性和预测机会不足**的问题，这些预取器只使用一个事件（如触发访问的指令和地址）来选择预取模式，导致预取精度低或预取机会少，需要一种新的方法来改进空间数据预取的效果。
- **技术缺陷**：TAGE-like predictor的缺点是**每个元数据表都需要单独的索引和标记**，导致存储开销很大，而且元数据表之间的冲突和重复也会降低预测效率。

设计理念

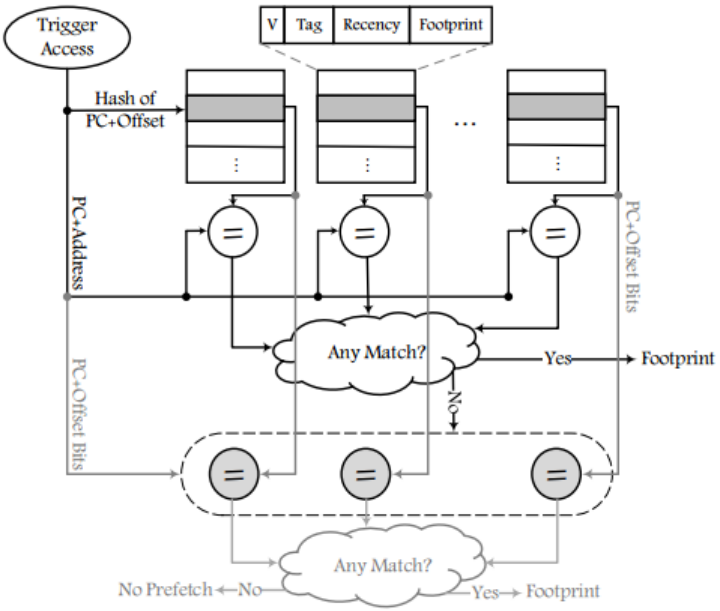


Figure 5. The details of the history table lookup in Bingo prefetcher. Gray parts indicate the case where lookup with long event fails to find a match. Each large rectangle indicates a physical way of the history table.

1. **基本思想**：主要基于观察，即**短事件嵌套在长事件中。**通过持有长事件，可以通过忽略长事件的部分来确定短事件。例如，对于Bingo，'PC+Offset'的信息包含在'PC+Address'中，因此通过知道'PC+Address'，也就能知道'PC+Offset'。

2. **历史表合并**：**只有一个历史表，只存储长事件的历史**，但可以用长事件和短事件进行查找。对于Bingo，历史表存储了在每个'PC+Address'事件之后观察到的足迹，但可以用触发访问的'PC+Address'和'PC+Offset'进行查找，这能提供高准确性并且还不失去预取机会。

3. **索引和标记**：表的索引仅使**用最短事件的哈希，但标记使用最长事件。**要存储历史元数据中的信息，需要将其与最长事件相关联，然后存储在历史表中。最短事件对应的位用于索引历史表以找到应存储元数据的集合；然而，**所有最长事件的位用于标记条目。**例如，使用Bingo，每当要将新的足迹存储在元数据组织中时，它与相应的'PC+Address'相关联。为了找到历史表中新条目的位置，仅使用'PC+Offset'的哈希来索引表。确定位置后，将条目存储在历史表中，但使用'PC+Address'的所有位来标记条目。

4. **预测**：每当需要进行预测时，**首先使用最长事件查找历史表**；如果找到匹配项，则将其用于进行预测。否则，应使用下一个最长事件查找表。由于长事件和短事件映射到相同的集合，因此

无需检查新的集合；相反，将测试同一集合的条目以查找与较短事件的匹配项。

5. **多事件关联**：在提出的设计中，每当使用较短事件查找表时，可能会找到多个匹配项。例如，对于Bingo，可能没有任何条目与触发访问的'PC+Address'匹配，同时多个条目与访问的'PC+Offset'匹配。这种情况对于Bingo，是个挑战，因为它需要根据可能不同的多个足迹信息发出预取请求。可以在这种情况下**使用各种启发式方法**，例如基于最近信息选择最新的足迹，或者根据所有匹配条目足迹中指示的块发出预取请求。论文经过一定的尝试，在经验上发现，考虑发出预取请求时所有匹配足迹信息提供了最佳性能：如果在至少20%的匹配条目足迹中存在缓存块，则预取该块。

实现方法

> **本研究采用了ChampSim模拟基础设施，模拟了基于Intel最新Xeon处理器的系统。通过对比实验结果，分析了Bingo预取器相对于现有技术的性能优势。同时，结合了现有的空间数据预取技术，提出了Bingo预取器的实现方法**

1. **系统配置和模型概述：**

Table I
EVALUATION PARAMETERS.

Parameter	Value
Chip	14 nm, 4 GHz, 4 cores
Cores	4-wide OoO, 256-entry ROB, 64-entry LSQ
Fetch Unit	Perceptron [76], 16-entry pre-dispatch queue
L1-D/I	Split I/D, 64 KB, 8-way, 8-entry MSHR
L2 Cache	8 MB, 16-way, 4 banks, 15-cycle hit latency
Main Memory	60 ns zero-load latency, 37.5 GB/s peak bandwidth

- 使用**ChampSim8**进行模拟，配置基于Intel最新的**Xeon处理器**，包括四个乱序执行核心和8MB共享的最后一级缓存(LLC)。
- 使用两个内存通道访问DRAM，提供最大37.5 GB/s的带宽。
- 操作系统使用4KB页面，虚拟到物理地址映射通过随机首次触摸翻译机制完成。
- 缓存块大小为64字节。

2. **工作负载：**

- 选择了**大数据服务器和科学应用程序**进行评估，以及来自一组内存密集的SPEC程序的五个四核代表性**混合工作负载**。
- 使用**SimFlex方法**模拟服务器工作负载，并对每个服务器应用程序创建五个检查点，其中包括热化缓存、分支预测器和预测表。
- 对SPEC基准进行模拟**至少100M指令**，使用前20M作为热身阶段，剩余80M进行测量。

3. **预取器配置：**

- 比较提案方法与最先进的空间数据预取器。
- 进行**灵敏度分析**，找到在工作负载套件中提供合理缺失覆盖所需的存储容量。
- 针对每个预取器，从原始工作中提取配置，并通过观察所有工作负载的平均缺失覆盖来调整存储容量。

4. **预取器具体配置：**

- **BOP** (Best Offset Prefetcher): 基于最近的请求表，通过测试单个偏移量来确定是否能够预测当前访问。配置包括256个最近请求表项。
- **SPP** (Signature Path Prefetcher): 计算每个偏移量模式的签名，根据签名为每个偏移预测的概率进行自适应调整。配置包括256个签名表项、512个模式表项和1024个预取过滤器项。
- **LDP** (Variable Length Delta Prefetcher): 利用多个历史差分来预测给定页面上的访问流。配置包括16个差分历史缓冲区、64个偏移预测表和三个64个差分预测表。
- **AMPM** (Access Map Pattern Matching): 标记最近访问的缓存块，检测步幅访问模式，并预测将来将被访问的块。将内存访问映射表扩展到LLC的整个容量。
- **SMS** (Spatial Memory Streaming): 引入足迹元数据到'PC+Offset'的触发访问。配置包括16K项16路关联的历史表。
- **Bingo**: 将空间模式与多个事件相关联，并以容量优化的方式将所有模式存储在单一的统一历史表中。使用16路组关联结构的历史表，并根据在第VI-A节的灵敏度分析中进行的设置容量。

5. **模拟和性能评估：**

- 在LLC的背景下研究所有数据预取器，利用大容量的**多兆字节LLC**（相对于一级缓存）为页面在该级别停留更长的时间提供机会。
- 考虑每个核心都有自己的预取器，独立于其他核心（即核心之间没有元数据共享）。
- 通过模拟和测量在不同工作负载下的性能，以评估提出方法相对于其他预取器的效果。

论文结果

1. **存储需求：**

- Bingo的效果直接取决于**历史表的大小**，历史表越大，缺失覆盖率越高。
- 确定将16K项分配给Bingo的历史表，总存储需求为119KB，仅占LLC容量的6%。

2. **缺失覆盖率与过度预测：**

- Bingo在所有工作负载下提供**最高的缺失覆盖率**，通过将足迹元数据与多个事件关联，最大程度地提取空间相关的数据访问模式，平均覆盖超过63%的缓存缺失。
- **过度预测**与其他预取器相当，但Bingo表现卓越。

3. **系统性能：**

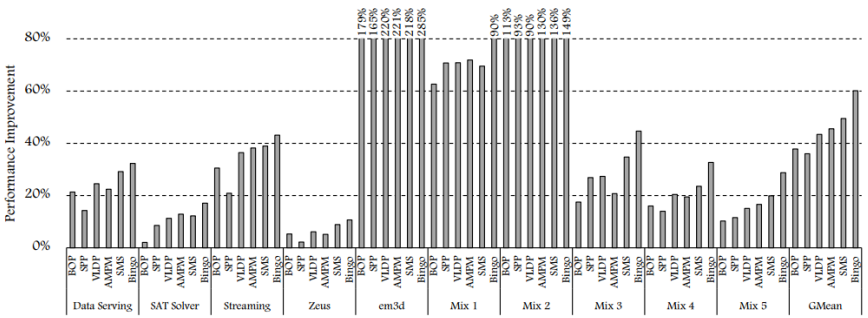


Figure 8. Performance comparison of prefetching techniques, normalized to a baseline system with no prefetcher.

- 从上面的图片可以看到，Bingo在所有工作负载下**一致优于其他预取器**，性能提升范围从11%到285%。
- 在Zeus中，由于内存访问在时间上更相关而非空间上，Bingo的性能提升较小。

4. **性能密度：**

- 使用性能密度（吞吐量每单位面积）评估预取器的效能。
- Bingo相较于其他预取器，将基线系统的性能密度提高了59%，显示其在**有效利用芯片硅区域**方面的优势。

5. **ISO度比较：**

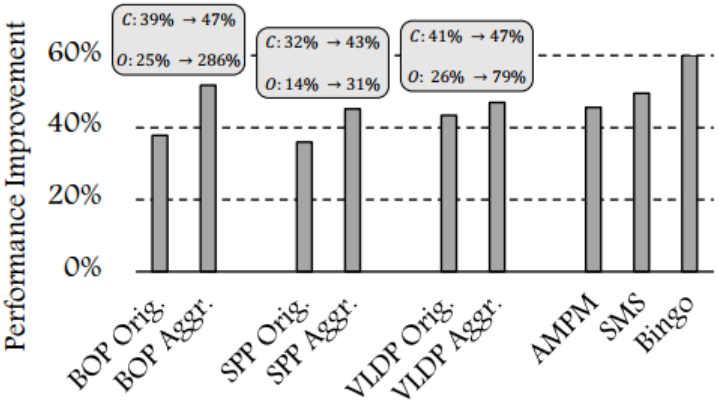


Figure 10. ISO-degree comparison of prefetching methods. 'Orig' indicates the original and so-far–evaluated version of an SHH-based prefetcher; however, 'Aggr' represents the aggressive and high-degree version. Callouts indicate how the coverage and overprediction of prefetchers vary from the original version to the aggressive version: 'C' and 'O' stand for 'Coverage' and 'Overprediction,' respectively.

- PPH（基于页面的历史）方法，如Bingo，具有更好的**及时性**，通过一次性为每个页面收集足迹并发出所有预期使用的块的预取请求，实现了卓越的及时性。
- 在ISO度比较中，Bingo仍然明显优于其他方法，显示其在及时性和性能提升方面的强大表现。

论文分析

总结：

> **这篇论文讲的是大数据应用里内存不足的问题，论文提出怎么用空间数据预取技术来解决这个问题。空间数据预取技术就是根据数据访问的规律，提前把可能用到的数据放到内存里面，这样就不会出现缓存缺失或者导致程序变慢的情况。论文发现，现在的空间数据预取技术还有很多不足，因为它只能根据一个事件来判断数据的规律，而不是多个事件。论文提出了一个新的方法，能够把多个事件的数据规律都考虑进去，这样就能提高空间数据预取的效果和准确性。论文还提出了一个通用的方法，能够去掉一些重复的或者没用的数据，让空间数据预取技术更高效。论文的实验结果显示，新的空间数据预取技术比现在的技术要好很多。**

创新点：

> **论文提出了一种创新的Bingo空间数据预取器,它结合了TAGE-like预测器和空间数据预取器的技术。Bingo预取器的特点是，它能够根据不同的事件（如PC+Address和PC+Offset）来判断每个页面的数据规律，然后把这些规律存到一个表里面，叫做统一元数据表（UMT）。UMT能够根据情况自动调整事件的重要性和替换方式，这样就能达到TAGE-like预测器的高准确度，而且还能节省很多存储空间。Bingo预取器比现在的空间数据预取技术好，它能够更准确地预测数据，也能提供更多的预测机会，使得计算机运行得更快。**

不足：

> **尽管Bingo预取器在性能上有所改进，但仍存在一些不足之处，例如在特定工作负载下的表现不佳，以及对于某些特定应用场景的适用性有限。**

论文复现

github地址：https://github.com/wzzzzzzzzzzzzz/Computer-Architecture

环境：ubuntu 20.04

cmake 3.26.3

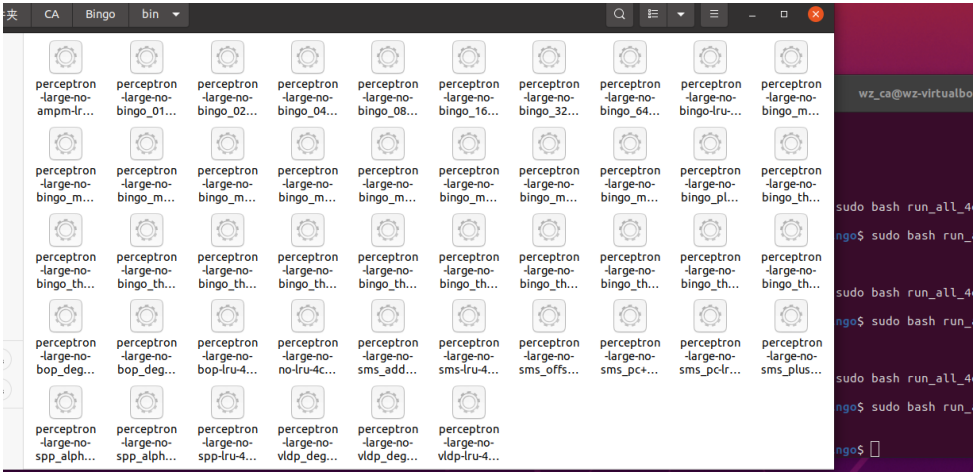
```
-- Performing Test run_inlines_hidden_test
-- Performing Test run_inlines_hidden_test - Success
-- Configuring done (26.8s)
-- Generating done (0.5s)
-- Build files have been written to: /home/wz_ca/Bingo/cmake-3.26.3
--------------------------------------------
CMake has bootstrapped.  Now run make.
wz_ca@wz-virtualbox:~/Bingo/cmake-3.26.3$ make
[  0%] Building C object Source/kwsys/CMakeFiles/cmsys.dir/ProcessUNIX.c.o
[  0%] Building C object Source/kwsys/CMakeFiles/cmsys.dir/Base64.c.o
[  0%] Building C object Source/kwsys/CMakeFiles/cmsys.dir/EncodingC.c.o
[  0%] Building C object Source/kwsys/CMakeFiles/cmsys.dir/MD5.c.o
```

```
wz_ca@wz-virtualbox:~/Bingo/cmake-3.26.3$ cmake --version
cmake version 3.26.3

CMake suite maintained and supported by Kitware (kitware.com/cmake).
wz_ca@wz-virtualbox:~/Bingo/cmake-3.26.3$
```

实验步骤

1. **执行脚本文件，设置四核处理器**：`bash build_all_4core.sh`

```
ChampSim is successfully built
Branch Predictor: perceptron-large
L1D Prefetcher: no
LLC Prefetcher: vldp
LLC Replacement: lru
Cores: 4
Binary: bin/perceptron-large-no-vldp-lru-4core

wz_ca@wz-virtualbox:~/CA/Bingo$
```



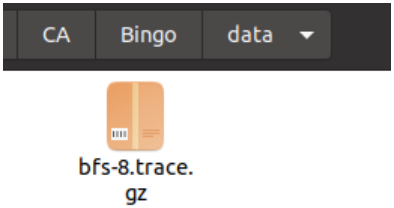
2. **修改文件夹权限，方便后续读写文件，否则会出现第二张图片的情况**

```
wz_ca@wz-virtualbox:~/CA/Bingo$ cd ..
wz_ca@wz-virtualbox:~/CA$ chmod -R 777 Bingo/
wz_ca@wz-virtualbox:~/CA$ cd Bingo/
```

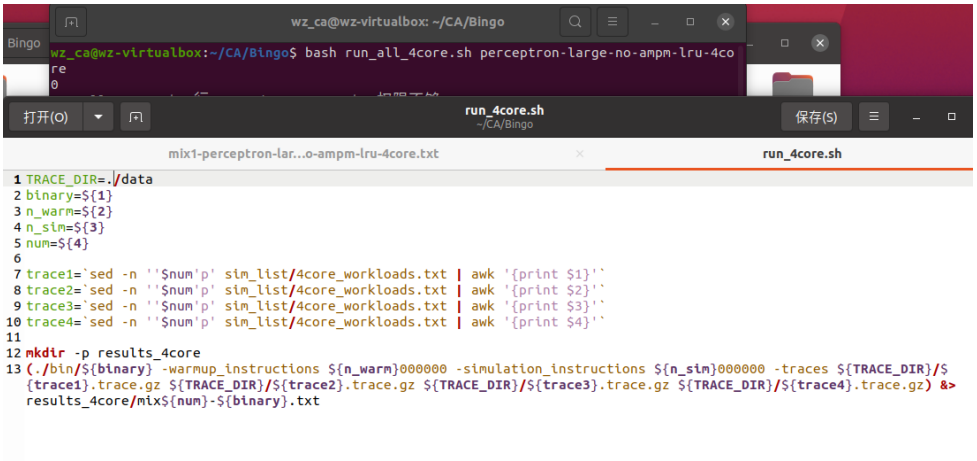
```
wz_ca@wz-virtualbox:~/CA/Bingo$ sudo bash run_all_4core.sh perceptron-large-no-ampm-lr
u-4core
[sudo] wz_ca 的密码：
0
run_all_4core.sh: 行 17: ./run_4core.sh: 权限不够
```

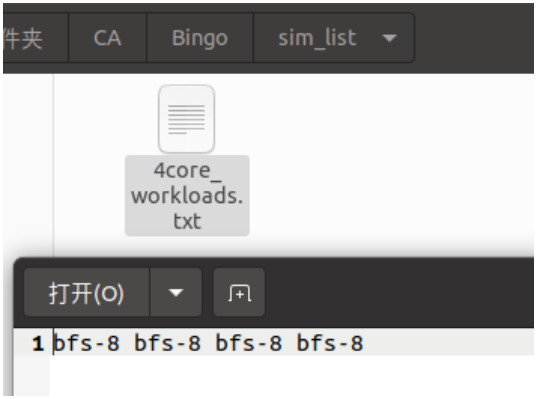
3. **放置数据集至（data文件夹下）**

> **数据集下载地址：**
>
> https://utexas.app.box.com/s/2k54kp8zvrqdfaa8cdhfquvcxwh7yn85/folder/132804598561

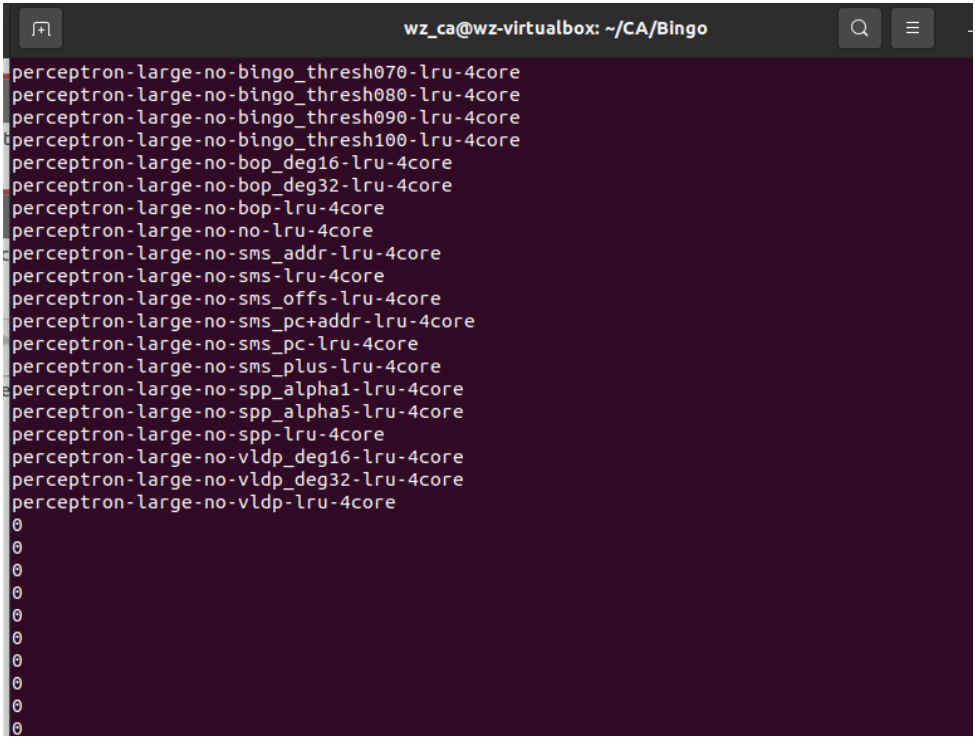


4. **添加数据路径**

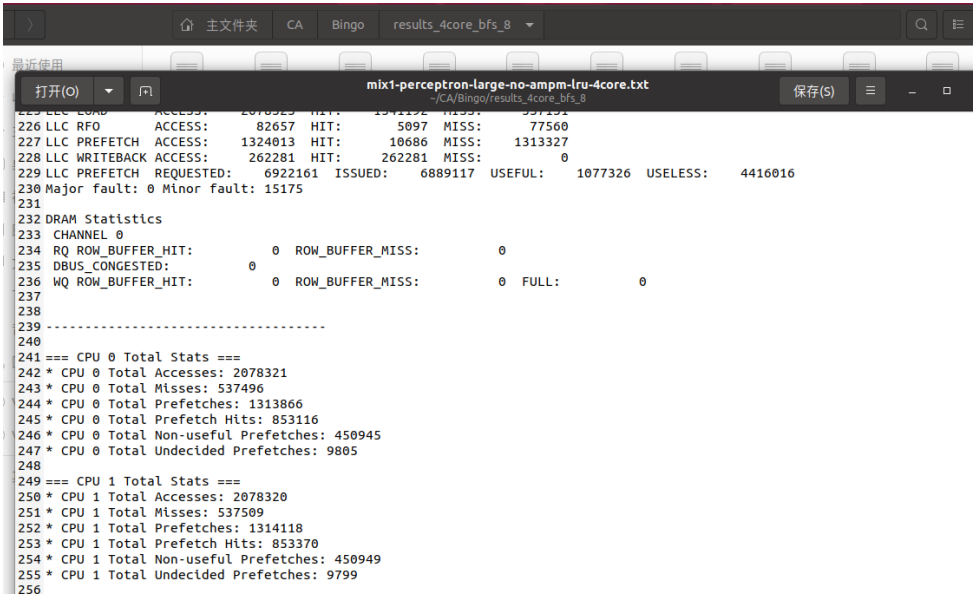




5. **运行** `bash hpca19_run_all_4core_v2.sh`


wz_ca@wz-virtualbox: ~/CA/Bingo
perceptron-large-no-bingo_thresh070-lru-4core
perceptron-large-no-bingo_thresh080-lru-4core
perceptron-large-no-bingo_thresh090-lru-4core
perceptron-large-no-bingo_thresh100-lru-4core
perceptron-large-no-bop_deg16-lru-4core
perceptron-large-no-bop_deg32-lru-4core
perceptron-large-no-bop-lru-4core
perceptron-large-no-no-lru-4core
perceptron-large-no-sms_addr-lru-4core
perceptron-large-no-sms-lru-4core
perceptron-large-no-sms_offs-lru-4core
perceptron-large-no-sms_pc+addr-lru-4core
perceptron-large-no-sms_pc-lru-4core
perceptron-large-no-sms_plus-lru-4core
perceptron-large-no-spp_alpha1-lru-4core
perceptron-large-no-spp_alpha5-lru-4core
perceptron-large-no-spp-lru-4core
perceptron-large-no-vldp_deg16-lru-4core
perceptron-large-no-vldp_deg32-lru-4core
perceptron-large-no-vldp-lru-4core
0
0
0
0
0
0
0
0
0
0

6. **得到结果**



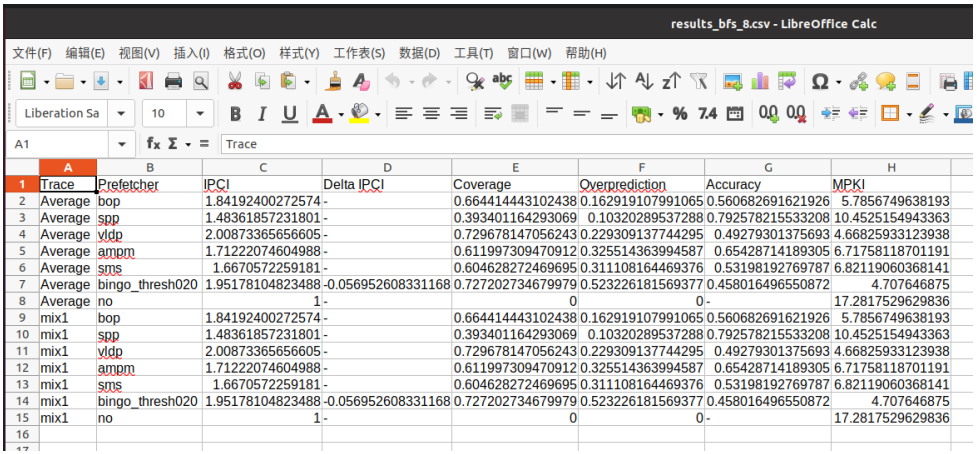
7. 比较指标

> 基于前面得到的results_4core文件夹
>
> ```
> python3 champsim_results.py
> ```

champsim_results.py按如下修改:

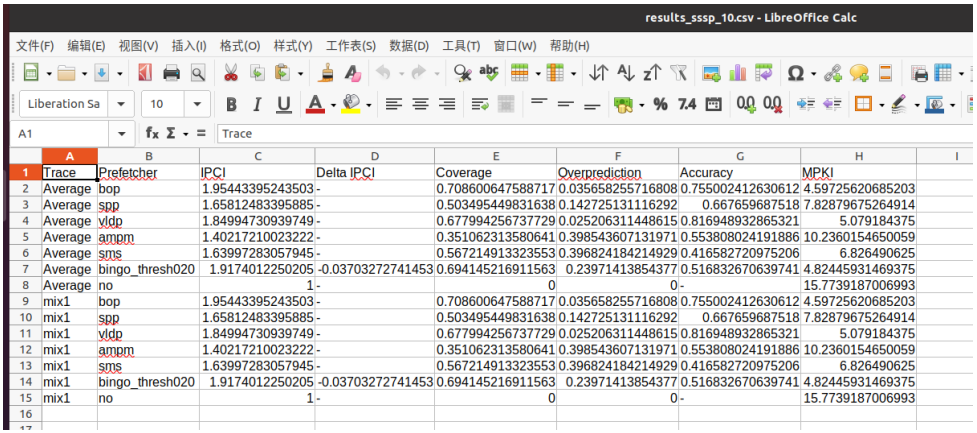
```
wz_ca@wz-virtualbox:~/CA/Bingo$ python3 champsim_results.py
wz_ca@wz-virtualbox:~/CA/Bingo$
```

bfs-8



	A	B	C	D	E	F	G	H
1	Trace	Prefetcher	IPCI	Delta IPCI	Coverage	Overprediction	Accuracy	MPKI
2	Average	bop	1.84192400272574	-	0.664414443102438	0.162919107991065	0.560682691621926	5.7856749638193
3	Average	spp	1.48361857231801	-	0.393401164293069	0.103202895372888	0.792578215533208	10.4525154943363
4	Average	vldp	2.00873365656605	-	0.729678147056243	0.229309137744295	0.49279301375693	4.66825933123938
5	Average	ampm	1.71222074604988	-	0.611997309470912	0.325514363994587	0.65428714189305	6.71758118701191
6	Average	sms	1.6670572259181	-	0.604628272469695	0.311108164469376	0.53198192769787	6.82119060368141
7	Average	bingo_thresh020	1.95178104823488	-0.056952608331168	0.727202734679979	0.523226181569377	0.458016496550872	4.707646875
8	Average	no	1	-	0	0	-	17.2817529629836
9	mix1	bop	1.84192400272574	-	0.664414443102438	0.162919107991065	0.560682691621926	5.7856749638193
10	mix1	spp	1.48361857231801	-	0.393401164293069	0.103202895372888	0.792578215533208	10.4525154943363
11	mix1	vldp	2.00873365656605	-	0.729678147056243	0.229309137744295	0.49279301375693	4.66825933123938
12	mix1	ampm	1.71222074604988	-	0.611997309470912	0.325514363994587	0.65428714189305	6.71758118701191
13	mix1	sms	1.6670572259181	-	0.604628272469695	0.311108164469376	0.53198192769787	6.82119060368141
14	mix1	bingo_thresh020	1.95178104823488	-0.056952608331168	0.727202734679979	0.523226181569377	0.458016496550872	4.707646875
15	mix1	no	1	-	0	0	-	17.2817529629836
16								

sssp-10



	A	B	C	D	E	F	G	H
1	Trace	Prefetcher	IPCI	Delta IPCI	Coverage	Overprediction	Accuracy	MPKI
2	Average	bop	1.95443395243503	-	0.708600647588717	0.035658255716808	0.755002412630612	4.59725620685203
3	Average	spp	1.65812483395885	-	0.503495449831638	0.142725131116292	0.667659687518	7.82879675264914
4	Average	vldp	1.84994730939749	-	0.677994256737729	0.025206311448615	0.816948932865321	5.079184375
5	Average	ampm	1.40217210023222	-	0.351062313580641	0.398543607131971	0.553808024191886	10.2360154650059
6	Average	sms	1.63997283057945	-	0.567214913323553	0.396824184214929	0.416582720975206	6.826490625
7	Average	bingo_thresh020	1.9174012250205	-0.03703272741453	0.694145216911563	0.23971413854377	0.516832670639741	4.82445931469375
8	Average	no	1	-	0	0	-	15.7739187006993
9	mix1	bop	1.95443395243503	-	0.708600647588717	0.035658255716808	0.755002412630612	4.59725620685203
10	mix1	spp	1.65812483395885	-	0.503495449831638	0.142725131116292	0.667659687518	7.82879675264914
11	mix1	vldp	1.84994730939749	-	0.677994256737729	0.025206311448615	0.816948932865321	5.079184375
12	mix1	ampm	1.40217210023222	-	0.351062313580641	0.398543607131971	0.553808024191886	10.2360154650059
13	mix1	sms	1.63997283057945	-	0.567214913323553	0.396824184214929	0.416582720975206	6.826490625
14	mix1	bingo_thresh020	1.9174012250205	-0.03703272741453	0.694145216911563	0.23971413854377	0.516832670639741	4.82445931469375
15	mix1	no	1	-	0	0	-	15.7739187006993
16								

> 分析:本次实验一共跑了两个数据集,分别是bfs-8和sssp-10
> 1.从IPCI（Instructions Per Cycle）这一列来看,Bingo的性能较强,每周期执行的命令数几乎最高
> 2.从Coverage这一列来看,我们不难看出Bingo在所有工作负载下提供几乎最高的缺失覆盖率,这与论文结果的特点一致(因为Bingo通过将足迹元数据与多个事件关联，最大程度地提取空间相关的数据访问模式)
> 3.从Overprediction来看,Bingo在bfs-8数据集表现较差,在sssp-10上表现中等,并没有同论文说的"表现卓越"一样
> 4.从Accuracy来看,Bingo的表现较差,这一点是比较大的不足,并且论文结果也没有对准确度进行分析,可能是它的准确度并没有得到较好的提升
> 5.从MPKI（Misses Per Kilo Instructions）--每千条指令的缺失次数来看,Bingo的表现优异,具有最低的MPKI,较低的MPKI值表示更好的缓存性能.

实验总结

> 本次论文复现的难度还是相当大的,首先是论文的选取,我选择的论文都是开源的,一开始选择了其他几篇论文,这些论文对电脑的配置要求都比较高,不是普通笔记本电脑能达到的,经常复现到一半,就出现虚拟机磁盘空间太小或者cuda配置等等问题,在磕磕碰碰下,最后选择了这篇Bingo论文,这篇论文的复现难度不算特别大,给的开源代码也十分齐全,我所需要做的,就是下载数据集,构建好环境,设置好多核处理器,修改数据存放路径,设置训练数据来源,运行bash文件即可.虽然从现在回看,听起来感觉不是很难,不过当时还是花费了一番功夫的,因为这篇论文的github并没有复现的教程,好在最后还是复现成功,也算是本科生涯第一次成功复现体系结构的论文了，相信这次实验经历对我之后的学习生涯（科研生涯）有着非同一般的影响!