REPORT OF LAB1

本报告介绍了我设计的 Modal Prefetcher 和复现的 C/DC Prefetcher.

MODAL PREFETCHER

Modal Prefetcher 是我设计的一个 prefetcher. 它工作的方式非常的简单, 但在特定的 trace 下效果不错. 源文件在 `nline` 文件夹下.

工作原理

在每一轮取址中, 我们做以下操作:

- 对地址进行差分, 即计算出相邻两次取址之间的差 `delta`.
- 将 `delta` 放在一个长度为 `BUFFER_LEN` 的 buffer 里. 这个 buffer 是一个 FIFO 的队列.
- 决定选址的方式是找到 buffer 里出现次数最多的 `max_freq_delta`. 如果这个 `max_freq_delta` 在4KB之内, 而且出现次数大于阈值 `LEAST_TIMES`, 我们就选择按这个步长进行 prefetching. 阈值的设置是为了避免出现次数过少的 `delta` 浪费L2和LLC的空间和队列.

我借鉴了 example 里的设计, 采用了如下方式决定 prefetch 到 L2 还是 LLC: 用 `get_l2_mshr_occupancy(0)` 得到目前 L2 被填充的字节数, 假如这个值小于8, 将新的数据 prefetch 到 L2, 否则将新的数据 prefetch 到 LLC.

实验参数

实验中我们选取了如下参数: `BUFFER_LEN=2048;LEAST_TIMES=20.` 结果在最后的表格中.

C/DC PREFETCHER

AC/DC Prefetcher 是Adaptive Data Cache Prefetcher的简称. 名字大概是在致敬伟大的 AC/DC 乐队. 在本实验中, 没有采用 Adaptive 的方式来调节 prefetch degree, 而实现了其余的算法, 称之为 C/DC Prefetcher.

另外, 在代码中我进行了简化, 去除了文章里使用的 Index Table. 但是 prefetching 的方式是一致的.

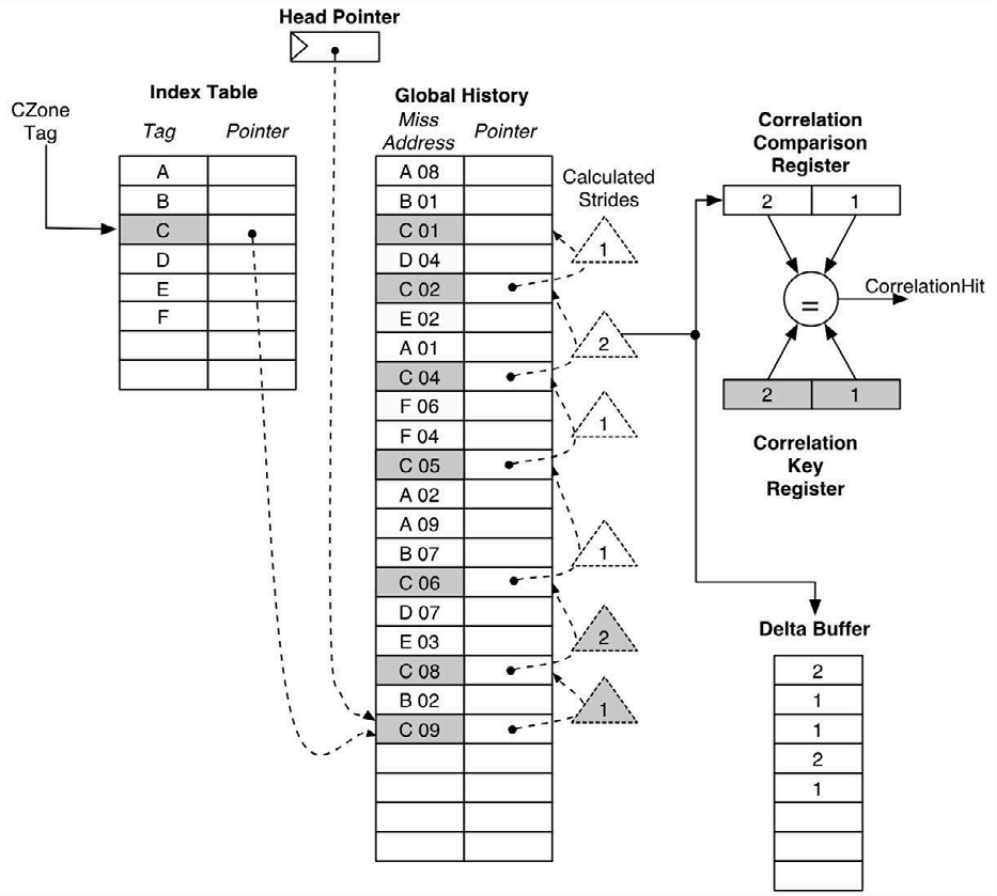
源文件在 `cdc` 文件夹下.

工作原理

在 C/DC Prefetcher 中, 地址被分成了不同的 Czones. 每个 Czone 的地址除了最后 12 个字节都相同. 这样的划分保证同一个 Czone 里的地址间距一定小于 4KB, 互相可以取得到. 当一个 Czone 中的地址 miss 了, prefetcher 会回溯同一个 Czone 之前 miss 的地址, 寻找相同的取址模式.

C/DC Prefetcher 有如下组件, 如下图所示.

- Global history buffer: 用来最近储存 L2 miss 的地址, 即是 FIFO 的.
- Correlation key register: 用来储存该 Czone 地址的 `delta` . 只储存最近两个 `delta` .
- Correlation comparision register: 用来储存 Czone 之前的 `delta` , 和 correlation key register 进行比较.
- Delta buffer: 在送入寄存器的同时, 地址的 `delta` 也写入 delta buffer. 用于之后生成要 prefetch 的地址. 是 LIFO 的栈结构.



每当一个地址发生 L2 miss, prefetch做如下操作:

- 将这个地址写入 global history buffer.
- 从新到旧地计算 global history buffer 中同一个 Czone 地址之间的差分, 并依次先送入 correlation key register, 再 FIFO 的方式送入 correlation comparison register. 送入寄存器时, 同时写入 delta buffer.
- 在送入中 correlation comparison register 的过程中, 如果发现两个寄存器中的值是全同的, 说明发现了相同的模式, 引发取址. 取址的方式是依次出栈, 累加在 base address 上. 取址的数目由 `PREFETCH_DEGREE` 来定义. 为了避免重复, 栈顶部的两个元素不参与.
- 假如 `PREFETCH_DEGREE` 大于 delta buffer 中 `delta` 的数量, 则循环利用栈中的值.
- 还有一种特殊情况: 假如发现 correlation key register 里两个值是相等的, 则认为是等步长的模式.

Reference: *"Nesbit, K.J., A.S. Dhodapkar, and J.E. Smith. "AC/DC: An Adaptive Data Cache Prefetcher." In Proceedings. 13th International Conference on Parallel Architecture and Compilation Techniques, 2004. PACT 2004., 135–45. Antibes Juan-les-Pins, France: IEEE, 2004." https://doi.org/10.1109/PACT.2004.1342548.*

实验参数

实验中, 我们将 prefetch degree 设置为 6, 相对而言是一个 less agressive 的参数选择. Global history buffer 的大小设置为 `GHB_LENGTH=256`.

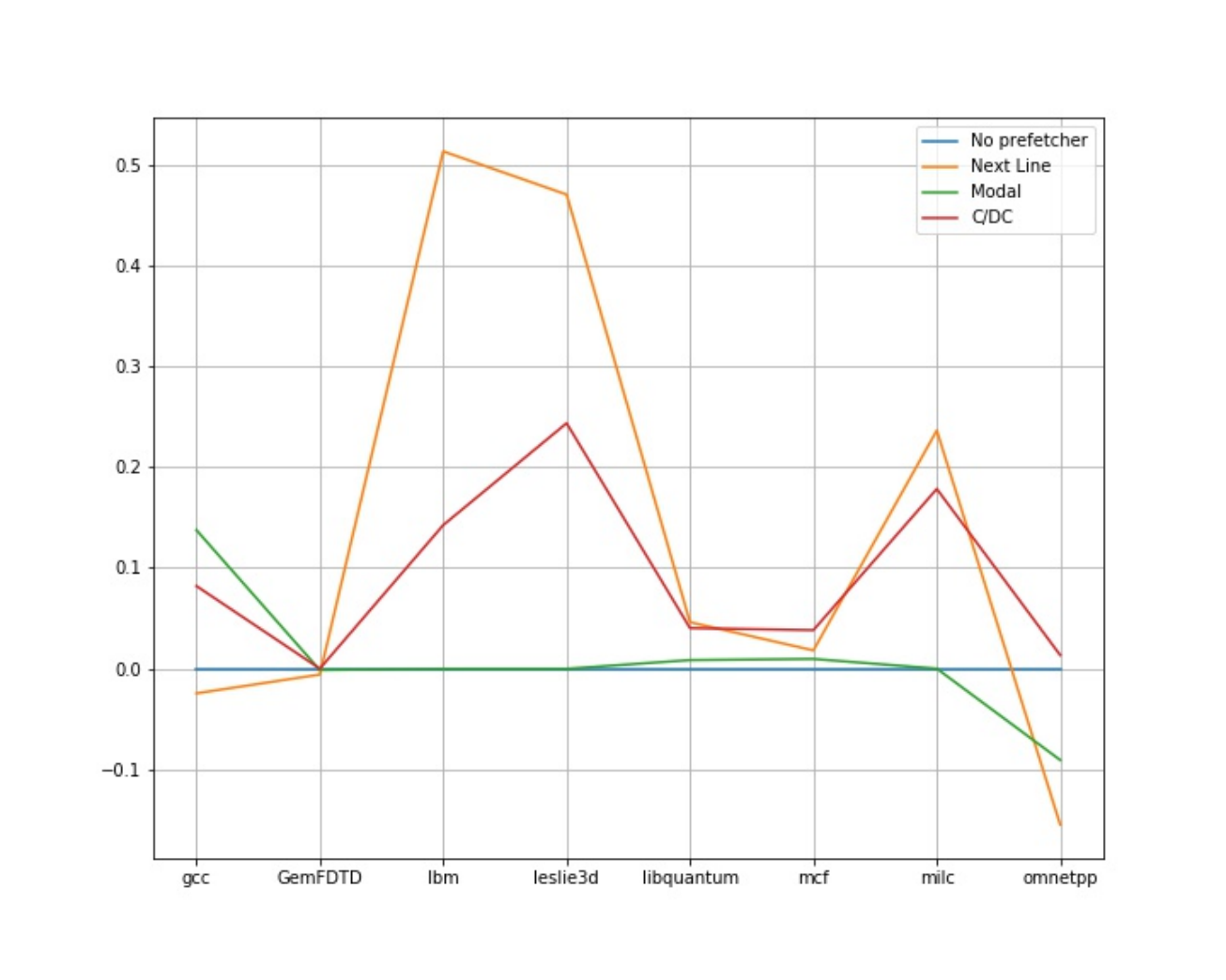
实验结果

两种 prefetcher 的 IPC 结果呈现在下表. 实验使用了 dpcsim2 给出的所有样例 traces.

作为参照, 也给出了不采用 prefetcher 以及采用直接取下一行的 Next Line Prefetcher 的结果.

Prefetcher	gcc	GemFDTD	lbm	leslie3d	libquantum	mcf	milc	omnetpp
No prefetcher	0.291329	3.444995	1.057604	0.985227	3.148157	0.344332	0.980482	2.191883
Next Line Prefetcher	0.284244	3.425332	1.600095	1.448414	3.293444	0.350610	1.212129	1.853220
Modal	0.331398	3.440693	1.057604	0.985227	3.175392	0.347700	0.980655	1.993612
C/DC	0.315239	3.445538	1.208256	1.225124	3.275189	0.357534	1.155308	2.221513

将几种 prefetcher 相对无 prefetcher 的加速比展示在下图.



简单的评论:

- Modal Prefetcher 对 gcc 的 trace 有非常好的表现, 其他的 trace 表现得不好. 推测原因可能在于 gcc trace 中地址间跨度分布比较集中, 适合用 Modal Prefetcher 来处理. 对于其他的 traces, 可能采用 adaptive 的方式确定参数的话, 可以有更好的效果.
- C/DC Prefetcher 整体表现很稳定, 至少都能取得不差于无 prefetch 的表现.
- 比较有趣的是, 在特定的 traces 下 Next Line Prefetcher 有很好的表现. 包括 lbm, leslie3d, milc. 说明这些 traces 中含有很多相邻的取址操作.
- 总的来说, prefetchers 的表现和 traces 的关系是很大的.