“PFU读源码”Pre稿

参考：玄铁C910微架构学习（11）——缓存系统的数据预取技术 - 知乎 (zhihu.com)

PFU Top

玄铁C910没有预取指令，仅支持硬件预取。玄铁C910支持**多模式（multi-mode）**和**多数据流（multi-stream）**的数据预取机制，通过对数据流进行模式匹配，有效地预取数据回填L1或L2缓存。主要有两种预取模式，一种是**全局预取模式**，即为正在执行的指令服务，适用于简单但连续的数据流，支持任意步长，并且预取的最大深度为64个缓存行（一行64Byte）。另一种模式是**多流预取模式**，即为最有需求的特定几条指令服务，适用于复杂的场景，该模式下最多支持8个具有不同步长的数据流，并预取的最大深度为32个缓存行（一行64Byte）。

（展示顶层结构框图）

我们知道，预取操作主要分为三个步骤，第一步计算步幅（指PMB、SDB、GSDB），也就是从指令中找到正确的访存模式。第二步预取控制（仍然是SDB、GSDB），它决定了预取策略并且评估置信度，预取的策略负责设置预取的深度，并动态调整预取的开始和停止。最后一步请求预取（指PFB、GPFB）。接下来我来简单介绍各个模块的功能和算法设计。

GSDB

GSDB——全局步幅检测缓冲区，为全局预取模式服务，用于计算预取步幅和评估步幅置信度。GSDB的核心是一台有限状态机。（展示SDB状态机）

- `IDLE`：默认初态；
- `GET_STRIDE`：收集3次访存地址，计算步幅；计算完成后跳转到CHECK_STRIDE检查步幅，并清空记录；
- `CHECK_STRIDE`：检查步幅，随后跳转到MONITOR_STRIDE监视或GET_STRIDE重新收集信息计算步幅，跳转前清空记录；
- `MONITOR_STRIDE`：监视步幅变化并调整置信度，置信度低时跳转到GET_STRIDE重新收集信息计算步幅，跳转前清空记录。

步幅计算：在`GET_STRIDE`状态下，记录连续的3次`load`指令的访存地址，前向差分，若差值相等即输出为步幅。（很草率是吧？哪怕来点二阶差分呢……）由于缓存以4KB分一页，64B为一行，需额外检查是否跨页、跨行。

其中与**记录**相关的部分是`Maintain newest iid`，通过iid比较子模块和表项命中情况以维持最新的DA阶段的`load`指令的iid。ld指令依次填入entry0，entry1，entry2。在填写entry2时，如果比entry1旧，entry1写入entry2，ld指令写入entry1。由于ROB处于退休窗口中的三条指令的commit条件是不同的，因此entry0是静态的。

步幅检查：字面意思。

步幅监视：通过不断地检查步幅来动态调整**置信度confidence**（取值0~3，初值2）。每成功检查步幅一次，则将置信度+1；若检查发现步幅与之前的步幅不一致，则置信度-1。置信度为0，此后按照原步幅预取可能性很低，因此重新计算步幅。

GPFB

GPFB——全局预取缓冲区为全局预取模式服务，从GSDB获得步幅后，GPFB会计算实际预取地址，并向BIU发出预取请求。首次计算预取地址，使用TSM。（展示TSM算法状态机）后续计算地址、发送预取请求，使用L1SM和L2SM。（展示L1SM算法状态机）因为使用了两类三个状态机来控制模块运行，所以我认为GPFB和PFB堪称PFU中最复杂最抽象的子模块。

动用两类三台状态机，其实都是为了回答"去哪里预取"的问题。当步幅准备好后，一旦发生缓存缺失事件，TSM就会开始第一次预取地址的计算——**访存地址+步幅+步幅高位**。这个公式很怪，究其原因是为了避免短步幅（步幅＜深度）导致浪费申请预取的次数。

后续的预取地址计算由L1SM、L2SM完成，如果不出意外，状态机会稳定在`REQ_PF`状态，此时的预取地址就是按照每次增加步幅高位的简单规则来更新的了。如果预取地址跨了页，会进入两个额外的状态`REQ_MMU`、`WAIT_PPN`来请求和等待地址映射，不改变预取地址的更新规则。其他意外状况，会将状态机困在`DEAD`，等待状态机重启。

疑问：谁来决定预取到L1DCache还是L2Cache?

PMB

PMB——预取缺失缓冲区，为多流预取模式服务。多流预取模式和全局预取模式最明显的区别，应该就是多流预取模式特有的PMB环节。PMB共有8个项，主要用于存储发生了缓存缺失的load/store指令的pc，并提醒SDB关注其中近期多次执行的项。

（展示流程图）一条PMB表项的生命周期是这样子的：发生缓存缺失时，找到PMB空位，将指令的pc、类型（load或store）记录下来。然后，根据指令类型选择load流水线或者store流水线中的有效指令的pc，与该entry中的pc进行比较，从而判断之前发生缓存缺失的指令是否出现多次执行的情况，当出现两次命中pc时，则尝试将该指令的pc送到sdb中进一步计算步幅。

由于一共只有8个项，因此为了防止一些只执行了一次的指令占用表项，每一个表项内部使用一个计数器，当表项有效时每周期+1，当计满时，则认为该项可以被替换，有新的项进来时，可以将其替换。

SDB

SDB——步幅检测缓冲区，为多流预取模式服务。SDB的步幅算法与GSDB的步幅算法类似，那么SDB和GSDB的区别是什么呢?

首先，SDB有2项，即可同时支持两条可能需要预取的指令进行步幅的计算和检查，起到两个GSDB的效果。

另外，全局预取模式是为所有DA阶段有效的`load`指令服务的，因此这些指令都会进入GSDB计算步幅；但是多流预取模式是针对不同指令分别分析的，因此需要用pc判断某条指令是否已被收录，只有收录的才能更新相应表项的数据。正因如此，为避免久久不见执行的僵尸指令占用SDB，每项配备了一个计数器，以便及时逐出僵尸表项，收录新表项。

疑问：为什么设计成8-2-8的沙漏形?

PFB

PFB——预取缓冲区，为多流预取模式服务。与GPFB类似，PFB也用于计算预取地址并发送预取请求，那么区别在哪里呢?

首先，PFB有8项，即可以同时支持8个不同步幅的数据流。每个数据流都是针对特定一条指令的，根据pc区分指令。为避免久久不成功预取的僵尸指令占用PFB，每项配备了一个计数器，以便及时逐出僵尸表项，收录新表项。

其次，L1DCache只支持1、2、4、8四种预取深度，L2Cache只支持4、8、16、32四种预取深度；而且L1DCache只支持`load`预取，不进行`store`预取，L2Cache则两者都支持。

另外，SDB置信度取值0~7，初值6。容错率大大增加。

其他有用的信息

发送请求的优先级判断：如果多流预取模式或全局预取模式同时发出请求，则优先处理多流预取模式的请求；如果多流模式下，多个pfb中的entry同时请求，则按entry的排序进行作为优先级排序，没有得到响应的请求则继续等待。优先发送L1Dcache的预取请求，仅当line fill buffer满了的时候（此时无法将预取的数据写回L1Dcache），会选择发送L2Cache的预取请求。

两种模式的对比与总结：**全局预取模式：**主要用于对于连续的等步幅的地址访问，对于密集矩阵和数组访问的情况下十分有效，当六次访问（三次用于步幅计算、三次用于步幅检查）的访问地址的步幅相等时，就会进行预取，预取的深度可通过**MHINT**配置，L1缓存最大支持16个缓存行的预取深度，L2缓存最大支持64个缓存行的预取深度。由于预取不准确会导致污染缓存，因此通过监视步幅，改变置信度，通过评估置信度动态调整预取的开始和停止，减少对缓存的污染。

多流预取模式：跨步幅访问的长序列称为流，多流预取表示同时支持多个步幅的预取，对于交错跨步幅的序列十分有效，比如矩阵向量乘法，多个矩阵交替访问，而每个矩阵访问的步幅可能不同，这时全局预取模式可能就无法发挥作用，而多流预取就可以应对这种情况。因此，多流预取模式是针对同一个指令（同一pc）多次执行，而每一次的访问地址按等步幅变化的情况下进行预取，当一条访存指令发生了缓存缺失且执行了三次，就会进行步幅的计算和检查（共需要再遇到该指令执行六次），步幅检查通过后就会进行预取操作，玄铁C910可同时支持8个不同的步幅数据流，L1缓存支持的最大预取深度为8个缓存行，L2缓存最多支持32个缓存行的预取。