Data Cache Prefetching with Perceptron Learning

<https://doi.org/10.48550/arXiv.1712.00905>

感知机（Perceptron Learning Algorithm）是一个二分类器，输入为特征空间，输出表示所属类别。

感知机模型可以表示为：

f(x)=sign(w⋅x+b)

其中， 输入x是n维的特征向量，输出y是1或者-1，w,b为感知机的模型参数。感知机模型时一种线性分类器，属于判别模型。

感知机模型的几何解释：线性方程w⋅x+b =0对应于特征空间Rn中的一个超平面S，其中w是该平面的法向量，b是超平面的截距。这个超平面将该空间分为两个部分，位于不同部分的实例属于不同的类别，位于相同部分的实例属于相同的类别。（可以简单地用二维平面来理解）

感知机的训练过程（即确定参数w，b的过程）主要使用了批梯度下降和随机梯度下降。此处不展开讲述，感兴趣的同学可以去自行了解。

常见的数据预取方法（如通过步幅计算进行预测）在面对不规则的数据访问时容易出错。可能出现清除有用的缓存并带来无用的数据，导致高速缓存空间的消耗和性能的下降。

内存污染通常发生在缓存的容量有限且数据访问模式不规律的情况下。一些常见的情况包括缓存的容量不足以容纳所有需要的数据，或者数据的访问模式难以预测，导致某些数据被频繁替换。内存污染会降低缓存的命中率，影响系统的性能。因此，缓存算法的设计通常旨在最小化内存污染的发生，以提高缓存的效率。

文章中提出了一种新颖的基于感知器学习的数据缓存预取方案。其关键思想是采用两级预取机制。首先利用先前基于表的预取机制，例如步幅预取或马尔可夫预取，进行初步决策，然后利用神经网络感知器检测和跟踪程序内存访问模式，帮助拒绝那些不必要的预取决策，从而降低内存污染和平均内存访问延迟，提高整体性能。

a) 步幅预取：步幅预取是一种广泛使用的预取器。传统的步幅预取器使用表格来保存与步幅相关的信息，并根据当前缓存未命中地址找出步幅模式。一旦根据表中记录的内存访问行为确认了步幅，预取器被触发后，关于地址A+s、A+2s、...、A+ds的预取请求将立即发出 - 其中s是检测到的步幅，d是预取度，是与实现相关的预取前瞻距离。在更激进的预取实现中，将使用更高的值来用于d。在高性能处理器中，步幅预取是最广泛使用的一种。

b) 马尔可夫预取：马尔可夫模型可以作为全局相关预取机制的基础。它的工作基于这样一个假设：未命中地址流可以用马尔可夫图近似，即一个概率状态机。这个图记录了从一个未命中地址转移到另一个未命中地址的转换的概率。图中的节点代表每个缓存未命中，弧表示从一个节点到另一个节点的每个转换概率，连接的。假设执行模式是重复的，则马尔可夫图用于预测下一个可能的缓存未命中地址。基于马尔可夫图，预取算法可以使用节点和弧中的信息以多种方式来预测未来的地址。

c) 全局历史缓冲区：全局历史缓冲区是一个FIFO结构，保存最近的缓存未命中地址，可以用于实现许多不同的预取方法，包括步幅预取和马尔可夫预取。此外，全局历史缓冲区提供了完整的缓存未命中历史

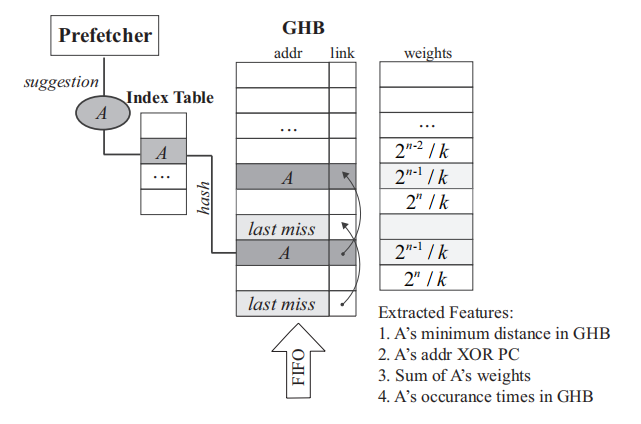
C910采用了步幅预取

在文章中，作者提出了一个两级预取器，第一级是之前基于表格的预取器与全局历史缓冲区（GHB）的结合，提供关于应该预取哪个缓存行的初步建议，同时将相关信息传递给下一级。在第二级中，感知器将利用这些定量信息来确定是否将预取请求发送到缓存读取队列。

感知器的输入向量如图所示，这些特征包括：

1. 预取距离（Prefetch distance）：预取距离指的是在数据被处理器需求访问之前，数据块已经被提前从内存预取到缓存中的距离。如果一个数据块的预取距离很高，意味着该数据块已经在缓存中待了很长时间，但却还没有被处理器使用。这种情况被称为缓存污染。
2. 转移概率（Transition probability）：当发生缓存丢失时，新条目将被推送到GHB中。靠近上次缺失的缓存行的条目在未来内存访问中有很高的可能性被使用。靠得越近，可能性越大。使用马尔可夫图中使用的转移概率来量化这个特征。最后一次缺失的缓存行出现了k次。然后，在GHB时间序列中，每个条目都被赋予权重2n-m/k，其中m表示该条目与最后一次缺失的缓存行之间的距离。如果预取器建议将A预取到缓存中，那么与A相关的所有权重之和将作为转移概率的表示。
3. 块地址位异或程序计数器（Block address bits XOR program counter）：此指标显示了数据块的时间和空间距离。
4. 出现频率（Occurrence frequency）：如果一个块经常出现在GHB中，这意味着处理器多次需要该块，但它被错误地从缓存中替换。因此，值越高，块应该被预取的可能性越高。
5. 固定输入（Fixed input）：感知器需要一个阈值来做出判断。如果输出超过某个阈值，那么就确定预取该块，反之则不进行预取。然而，找到适用于所有情况的固定阈值是不可能的。因此，为了获得最佳性能，应该为不同的工作负载提供不同的阈值。文章的解决方案是提供一个额外的输入设置为1，并相应地分配一个新的权重。这个新的权重可以作为阈值，同时适用于各种工作负载。

计算量化特征和权重乘积的总和作为 y\_out。如果 y\_out 超过 0，则决定预取该数据块。相反，如果 y\_out 不超过 0，则决定不预取该数据块。

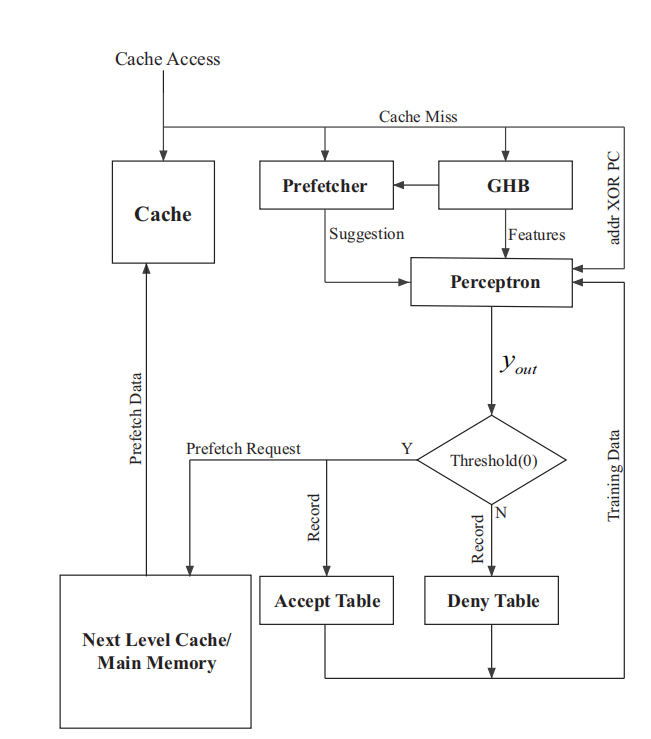


预取机制数据路径如图所示。一旦发生缓存未命中，GHB 将推送一个新的条目。同时，预取器将被触发，并提供预取地址的建议。在 GHB 中进行搜索，将相关信息从 GHB 中提取，并作为输入传递给感知器。

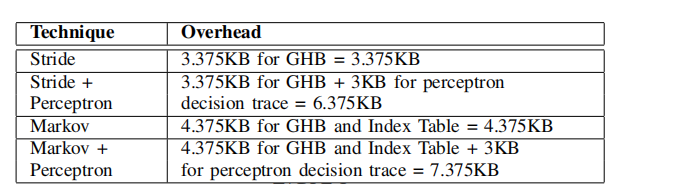
经过量化后，感知器计算每个输入和权重的总和，得到 yout。

如果 yout 超过 0，将采纳此建议，并通过缓存读取队列将预取请求发送到下一级缓存或主存储器。同时，这一接受记录也将被推送到接受表中。否则，建议将被拒绝，并记录在拒绝表中。接受表和拒绝表中的记录都将作为感知器的训练数据，以便对可能的内存访问模式变化进行调整。

对于接受表中的条目，如果在256个缓存引用内没有被访问，则被视为错误预测。对于拒绝表中的条目，如果在32个缓存未命中（32个预取触发）内没有被访问，则被视为正确的拒绝。这些信息将作为感知器在线训练的反馈。



空间开销：



可以看到，感知机的空间开销还是比较小的。

实验结果：

1. 感知器学习分别带来了 0.05% 和 0.29% 的 IPC 增加，这表明新方案实现了轻微的性能提升。
2. 平均而言，与原始预取器（步幅预取、马尔科夫预取）相比，新方案分别将预取正确率的算术平均值提高了 5.68% 和 1.82%。
3. 平均而言，与原始预取器（步幅预取、马尔科夫预取）相比，新方案分别将预取错误率的算术平均值降低了 11.17% 和 14.5%。
4. 平均80.14% 的来自步幅预取器的不必要的预取建议和平均49.71% 的来自马尔可夫预取器的不必要的预取建议被感知器在第二级中检测到并拒绝。减小了内存压力。
5. 值得注意的是，在使用感知器学习后缓存命中率的轻微浮动，平均分别下降了 0.03% 和 0.15%。即感知器不可避免地会拒绝一些必要的预取请求，这进一步导致了缓存未命中，作为降低缓存污染和平均内存访问延迟的代价。

这篇论文提出了一种新的两级预取方案，第一级采用基于表的预取工作，用于提供建议并提供必要的相关信息，第二级采用感知器学习工作，用于做出最终决策。与使用固定模式不同，感知器学习结合了局部和全局、时间和空间历史，可以动态检测和跟踪程序的内存访问模式。

实验结果显示：感知器拒绝了大量不必要的内存请求，从而改善了缓存污染并减少了内存流量，同时对指令每周期执行数（IPC）和缓存命中率产生了较小的影响。