宾果空间数据预取器

穆罕默德·巴赫夏利普尔·梅赫兰·沙克里纳娃·佩曼·洛特菲-卡姆兰·哈米德·萨尔巴兹-阿扎德
谢里夫理工大学计算机工程系
基础科学研究所计算机科学学院

摘要—应用程序广泛使用具有规则和固定布局的数据对象，这导致在内存区域上重复出现访问模式。

利用这一现象预取未来的内存引用，并隐藏DRAM访问的长延迟。尽管最先进的空间数据预取器在减少数据缺失数量方面很有效，但我们观察到仍有很大的改进空间。为了选择预取的访问模式，现有的空间预取器将观察到的访问模式与具有高重现概率的短事件或具有低重现概率的长事件相关联。因此，预取器要么提供低精度，要么失去重要的预测机会。

我们发现，将观察到的空间模式与单个事件相关联会显著限制空间数据预取器的效率。在这篇论文中，

我们证明

将观察到的空间模式与短和长相关联
在不丢失预测的情况下获得高精度的事件

和长事件被用
机遇。
的最佳访问模式。我们为宾果游戏提出了一种存储高效的设计

组大数据应用程序的详细评估，我们表明宾果比没有数据预取器的基线系统性能提高了60%，比性能最好的先前空间数据预取器提高了11%。

关键词-大数据应用，内存系统，数据预取，空间相关性。

I. 介绍

对于许多大数据应用来说，长延迟片外存储器访问是众所周知的性能瓶颈。由于处理器和片外存储器的速度不匹配，每次访问动态随机存取存储器时，处理器很容易停滞数百个周期，从而失去显著的性能潜力。今天的高度推测性和深度流水线乱序处理器充其量只能容忍主数据高速缓存未命中，并在片外存储器访问时招致相当大的性能损失[1]，[2]，[3]，[4]，[5]，[6]，[7]。

传统上，处理器设计人员增加了片内高速缓存的容量，以提高命中率并减少片外访问次数。然而，这种方法不太适用于当今的处理器，因为它会导致缓存命中延迟增加[3]，[8]，[9]，[10]。此外，使用硅面积来增加内核数量比扩大缓存更有益[3]，[8]，[9]。最后是

2378-203 x/19/$ 31.00 2019
IEEE DOI 10.1109/hpca .
2019.00053

应用程序数据集的持续增长(例如，图形处理和机器学习)已经导致了数百千兆字节甚至几万亿字节的数据集；比活动芯片上可能的最大缓存大几个数量级。

系统架构师使用了各种工具来弥补处理器和内存之间的性能差距。数据预取是这些工具中的一种，它在减少缓存未命中中的延迟方面表现出了巨大的潜力[11]，[12]，[13]。预取是预测未来内存访问的行

cessor 明确要求它们，以隐藏片外访问的长延迟。如今，几乎每个高性能处理器都使用数据预取(例如，英强融核[14]、AMD 皓龙[15]和 UltraSPARC III

目标是规则和/或不规则的内存访问模式。

空间数据预取器通过依赖空间地址相关性来预测未来的内存访问:
多页内存中的访问模式 1。也就是说，如果一个程序已经访问了页面 A 的 X、Y、Z 位置，那么它将来很可能会接触到相同或相似页面的 X、Y、Z 位置。访问模式展示了空间相关性，因为应用程序使用具有规则和固定布局的数据对象，并且当数据结构被遍历时，访问会重新出现[17]，[18]。

每当应用程序请求页面时，空间数据预取器(例如[18]，[19]，[20]，[21])会观察对页面的所有访问，并记录一个足迹，指示应用程序使用了页面的哪些块。然后，他们将足迹分配给一个事件，并将事件、足迹对存储在历史表中，以便在将来事件再次发生时使用记录的足迹。事件通常从触发器访问中提取，即第一次访问页面 2。在同一事件再次发生时，空间预取器使用记录的足迹来预取当前请求的页面的未来存储器引用。

数据预取器，例如时间数据预取器[22]，[23]，[24]

1 这里，页面是内存中一大块连续的缓存块，保存着几千字节的数据。这种页面不一定与操作系统页面或动态随机存取存储器页面相同，无论是在性质上还是在大小上。

2 例如，请求的指令的程序计数器
页面第一次可以是一个分配了足迹的事件。

Authorized licensed use limited to: CHONGQING UNIVERSITY. Downloaded on August 31,2020 at 12:41:54 UTC from IEEE Xplore. Restrictions apply.

元数据存储空间减少。此外，空间预取器可以通过将从相似页面中学习到的访问模式推广到新的未观察到的页面来预取性能关键的强制未命中(即，未看到的高速缓存未命中)，从而显著提高系统性能。最后，正如最近的工作所显示的[29]，空间预取器不仅通过减少片外访问的次数来提高性能，而且通过减少耗能的动态随机存取存储器行激活的次数来提高存储系统的能效。

传统上，未命中覆盖率，即预取器消除的缓存未命中的比例，是预取器设计中的主要考虑因素。因此，预取器消除缓存未命中的能力有所提高，而存储效率和预取准确性等其他因素被边缘化。尽管如此，随着多核处理器的广泛使用，存储需求和预取精度等其他因素变得越来越重要。硬件优化器(如预取器)的存储开销应该最小；否则，消除优化器并将其硅片专用于进一步增加内核数量可能是有益的[30]。预取精度也变得至关重要，因为高内核数已将设计推入内存带宽墙，主要是因为引脚数可扩展性差[31]，[32]，[33]，[34]，[35]，[36]。因此，预取器应该高度精确，以有效地使用 DRAM 模块[37]，[38]，[39]，[40]，[41]的有限带宽。在这两者中，预取精度比存储效率更重要，因为设计通常会首先触及带宽墙[8]，[9]，[42]，[43]。

受先进的分支预测器 TAGE [44]的启发，最近的许多工作使用多个 cas- caded 历史表 3 提高了基于预测器的硬件优化器的效率。在这种策略中，不是依靠单个历史表来预测未来的事件(图 1-(a))，而是使用几个历史表来进行预测(图 1-(b))，每个历史表都有特定的信息。这些表格记录了长事件和短事件的历史。长事件是指几个具体事件的巧合。例如，
“使用指令 I5 访问页面 P2 的第三个高速缓存块”可以被认为是一个长事件(三个事件的巧合)。另一方面，短事件指的是
一些特定事件的巧合。例如，“指令 I5 的执行”可以被认为是一个短事件(仅仅是一个事件)。

类似于 TAGE 的预测器中的多个级联历史表中的每一个都以特定的长度存储事件的历史，并提供对存储的事件之后将发生什么的预测。
期望具有高

TAGE 分支预测器本身的灵感来自于以前关于数据压缩的工作[45]，该工作研究可预测性极限。

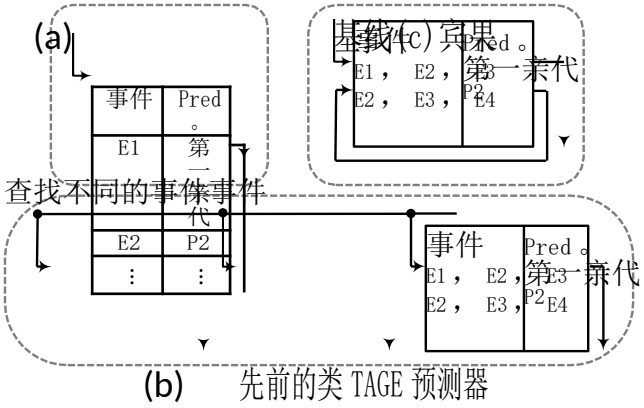


图 1。基于预测器的硬件优化方案的比较。

长时间事件重复发生的可能性很低。因此，如果一个预测者仅仅依靠长期事件的历史，它很少能做出预测(但是当做出预测时，它是高度准确的)。恰恰相反，

和基于长期事件的预测一样准确。为了两全其美，类似 TAGE 的预测器记录了长事件和短事件的历史。每当有

一个接一个；它们从最长的历史表开始

无法进行预测，他们会切换到下一个最长的历史表并重复这个过程。这一过程使预测者能够尽可能准确地预测，同时不会失去预测的机会。

许多先前的工作通过使用类似于 TAGE 的策略来提高各种基于预测器的硬件优化器的效率，展示了巨大的潜力。类 TAGE 策略用于分支预测[46]，[47]，[48]，数据预取[28]，[49]，[50]，数据值预测[51]，[52]，[53]，存储器依赖性预测[54]，[55]，高速缓存命中/未命中预测[56]，近似计算中的质量预测[57]，基于预测的 DRAM 调度[58]，以及指令类型预测[59]。

在本文中，我们[illegible]利用这一思想，并提出[illegible]，然而[illegible]有效机制[illegible]。宾果[illegible]，[18]，[19]，[20]，[21])一样，将每页的足迹存储为元数据，但与它们不同的是，

一个事件。每当预取时间到来时(即触发访问发生时)，宾果会找到与发生时间最长的事件相关联的足迹。因此，宾果在不损失预取机会的情况下发出准确的预取请求，主动向处理器提供请求的数据。

宾果的天真实现需要奉献

Authorized licensed use limited to: CHONGQING UNIVERSITY. Downloaded on August 31,2020 at 12:41:54 UTC from IEEE Xplore. Restrictions apply.

多个历史表，以保存元数据。在这样的实现中，每当需要存储足迹时，就将其插入所有元数据表中，并分配给每个表中具有不同长度的事件。这种方法已经被以前的类似TAGE的预测器所采用，它增加了大量的面积开销，如图1-(b)所示。我们观察到，在空间数据预取的环境中，存储在级联的类TAGE表中的元数据的很大一部分是冗余的。为了有效地消除冗余，我们提出了一个优雅的解决方案，将所有历史表的元数据合并成一个

张
2020-09-01 00:18:09

空间预取：连页内访问的顺序都不用记录（能减小存储需要）

（图1-(c)）。通过在单个历史表中组织元数据，我们利用所提议的实施方式，统一表格。

历史表被查询多次，每次都有一个不同的事件来查找与最长事件

在本文中

- 我们表明，事件发出预取请求是最先进的空间数据预取器效率低下的主要原因。
- 我们提出了一个类TAGE的预测器来准确和最大限度地提取空间相关的数据访问模式。
- 我们建议一个方案来巩固多个历史
- 综上所述，我们提出了一个名为宾果的空间数据预取器，并针对各种大数据应用对其进行了细致的评估。我们表明，我们的建议平均将系统性能提高了60%，比没有预取器的基线提高了285%。同时，它的性能比性能最好的先前空间数据预取器平均高11%，最高可达67%。

II. 背景

现代大数据应用程序拥有庞大的数据集，使容量有限的缓存相形见绌，并且驻留在内存中。因此，执行这些工作负载的处理器经常会遇到使内核停滞的缓存未命中，从而导致显著的性能损失[3]，[4]，[28]，[34]，[60]，[61]。空间的
数据预取器[18]、[19]、[20]、[21]、[50]、[62]、[63]、[64]，
[65]，[66]通过基于存储器页面之间的访问模式的相似性预取未来的存储器引用来减少高速缓存未命中的数量。

空间数据预取一直被认为是有效的，因为它提供了独特的内在特性。首先，与其他类型的数据预取器相比，空间预取器，例如时间数据预取器[22]，[23]，[24]，[25]，[26]，[27]，[28]，需要数量级更少的存储来保存元数据信息。与时间预取器不同，

（即块地址离页面开始的距离）

或者增量(即落入一页中的两次连续访问的距离)，并且不需要存储完整地址。因此，他们需要更少的存储空间来存储元数据。此外，在空间预取中，

因此，不需要记录页内访问顺序，这进一步降低了存

存页面通常小于或等于一个动态随机存取存储器行(例如，2 - 2kb，而不是2 - 8kb)，所有与触发访问一起发送的预取请求都会受到行缓冲区命中的影响，因此，所有这些请求都会被快速提取并缓存在最后一级缓存(LLC)中，从而降低了提取顺序的影响[29]。

空间数据预取器的另一个同样显著的优势是它们消

单身

缓存未命中的能力。强制缓存未命中是重要类程序性能下降的主要原因，例如，扫描主导的工作负载，其中扫描大量数据会产生大量缓存无法捕获的不可见内存访问[67]。通过将过去页面中观察到的模式应用于新的未观察到的页面，空间预取器可以减轻强制缓存未命中，从而显著提高系统性能。

最后，精确的空间数据预取器不仅提高了性能，还提高了内存子系统的能量利用率。空间数据预取器有机会通过精确预测预期使用的缓存块，并在单个动态随机存取存储器行激活中提取所有有用的缓存块，来提高动态随机存取存储器行缓冲区命中率。通过这样做，它们可以防止多次高能耗的动态随机存取存储器行激活，否则如果没有空间预取器存在，就会发生这种情况[29]。

我们将先前关于空间数据预取的工作分为两类:每页历史(PPH)和共享历史(SHH)方法。PPH指的是记录

将记录的历史与事件相关联，最后将历史存储在元数据

局所有访问的方法

共享元数据组织。

SHH类空间预取器将存储效率作为首要设计考虑。这些预取器的范围从简单的跨步预取器[68]，[69]，[70]到更高级的预取器
预取的复杂试探法[50]，[62]，[63]，[64]。通常，这些预取器维护一个单一的元数据结构来记录所有页面观察到的模式。换句话说，它们不会为每一页存储一个模式；相反，他们将随处可见的访问模式融合到一个统一的组织中。例如，基于增量的SHH预取器(例如[50])可能观察到三个连续的访问，比如在特定的页面P1中的A1、A2和A3，并且

Authorized licensed use limited to: CHONGQING UNIVERSITY. Downloaded on August 31,2020 at 12:41:54 UTC from IEEE Xplore. Restrictions apply.

生成两个连续的增量 d1 和 d2 (d1 = A2 A1 和 d2 = A3 A2)。在这种情况下，它不是记录“在 page P1 中观察到 A1、A2 和 A3”，而是在全局元数据历史中记录“d2 跟随 d1”

SHH 策略显著降低了存储需求-

预取器消除缓存未命中的能力

。另一个重要挑战

嘘的方法是 的数量

请求预取器立即发出。在 PPH 方法中，正如我们将很快讨论的，每当预取器被触发时，它会立即获取页面中所有预期要使用的块，这由页面覆盖区决定。然而，在 SHH 方法中，没有这样丰富的信息，因此，预取器不知道它应该发出多少预取，以便及时接收块。基于 SHH 的方法 SPP [62]提出了自适应抑制预取程度的技术:只有当预测的估计精度高于某个阈值时，才会发出预取。虽然这种启发式方法在控制预取程度方面可能很有用，但它们会导致

越来越依赖节流的准确性另一类空间预取是 PPH。一页

一次

被应用程序第一次请求(即触发

访问)，PPH 方法

只要应用程序正在使用该页面，就可以访问该页面。

使用时(即页面结束驻留)，这些方法就会关联

记录的历史通常是一个位向量，称为页面覆盖区，其中每个位对应于页面的一个块:覆盖区中的“1”表示

印相关联的事件通常是从触发器访问中提取的。例如，Kumar 和 Wilkerson [17]提出使用触发器访问的“PC+Address”作为事件:触发器指令的“PC”与触发器指令请求的“Address”相结合。作为另一种情况，索莫吉等人[18]评估了几个试探法作为事件，并根据经验发现“PC+Offset”比其他试探法表现得更好:触发指令的“PC”与“Offset”相结合，Offset 是所请求的高速缓存块离所请求的页面开始的距离。稍后，在相同事件再次发生时(例如，对于“个人计算机+偏移”的情况，具有相同“个人计算机”的指令请求在页面的“偏移”距离内的高速缓存块)，这些预取器应用存储的足迹来预测和预取当前请求的页面的未来存储器引用。

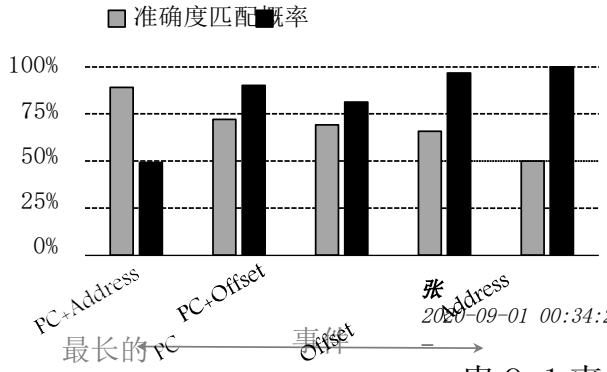
与这些方法相关的一个挑战是找到页面足迹应该分配到的最佳事件。每种启发式方法都有自己的优缺点。例如，在提到的两个事件中，“PC+地址”[17]是高度准确的，因为它保守地等待相同的指令被重新执行和相同的地址被触摸。虽然这种方法是准确的，但它无法覆盖强制缓存未命中，因为存储的

张
2020-09-01 00:23:43
-
全局下记录三角洲
[illegible]，一 某个页不是
模式就导致缓存未
要使用的足迹。另一方面，PC+Offset’[17]，具有攻击性，可以通过将一个页面的足迹与另一个页面来覆盖强制缺失，但是基于它所做的预测并不太准确。在本文中，我们表明，与将每个足迹与多个事件相关联并使用最佳匹配事件进行预取的机制相比，仅依赖其中一种启发式方法是次优的。

III. 动机

计划设计一个高性能的空间数据预取器，我们将宾果的设计空间缩小到基于 PPH 的方法。图 2 显示了各种试探法的准确性和匹配概率，作为页面覆盖区关联的事件，在所有应用程序中取平均值 4。准确性是所有预取的百分比

事件的一小部分



张
2020-09-01 00:34:28
-
一串 0,1 来记录页中
区是否被访问了

图 2。各种试探法作为与页面访问历史相关联的事件的准确性和匹配概率。

随着事件变长，预测的准确性增加，而匹配概率通常降低。在评估的试探法中，“PC+Address”是最长的事件(即，相同的指令和相同的地址应该同时发生)，它给出了最高的预测精度，但是对于该事件，由于事件重复发生的概率较低，预测的机会较少。因此，如果预测者仅仅依靠

4 完整的应用程序列表请参见第五节。

Authorized licensed use limited to: CHONGQING UNIVERSITY. Downloaded on August 31,2020 at 12:41:54 UTC from IEEE Xplore. Restrictions apply.

对这一事件，它的预测将是准确的，但它将无法经常做出预测。

另一方面，随着事件变短，预测的准确性降低，但预测机会通常增加。以“偏移”作为事件，这是被评估的事件中最短的事件(即，刚好一个块离页面开始的距离应该再次出现)，有很高的预测机会；但是预测不如更长时间的事件准确。因此，如果预取器仅使用此事件，它通常能够发出预取请求，但预取将

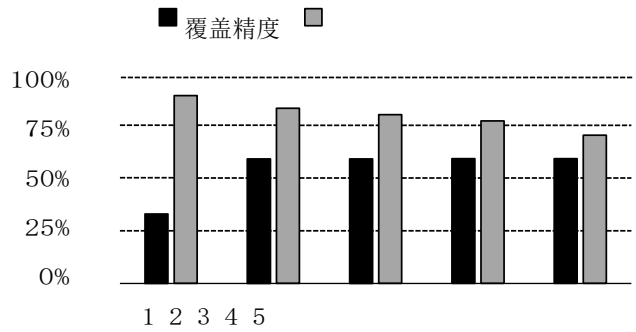


图 3。页面足迹关联的事件数量不同的类 TAGE 预取器的覆盖范围和准确性。当预取器使用一个事件时，该事件是 PC+地址。随着事件数量的增加，预取器可以使用更短的事件。当事件数变为 5 时，预取器可以使用所有事件。

不可接受的不准确。

这一观察激发了一种机制，其中不止一个事件被用来进行预测。一旦记录了页面足迹，它就与多个事件相关联，然后存储在历史表中。也就是说，页面足迹与，比如说，

‘PC+Offset’和‘PC’，然后存储在历史表中。每当预取时间到来时(即发生触发访问)，预取器查找事件最长的历史(即“PC+Address”):如果找到匹配，预取器根据匹配的足迹发出预取请求；否则，它会以递归方式查找具有次长事件(即“PC+Offset”)的历史记录。这样，预取器受益于高精度和高机会，克服了先前提出的空间预取器的限制。

为了演示使用多个事件的重要性，图 3 显示了当事件数量从 1 到 5 不等时，将页面覆盖区关联到多个事件的空间预取器的未命中覆盖率和准确性。当事件数为 1 时，预取器总是将页面足迹与最长的事件(即“电脑+地址”)相关联。随着事件数量的增加，预取器可以将页面足迹与较短的事件相关联。当事件数为 5 时，预取器能够将页面覆盖区与所有事件相关联，包括最短的事件(即“偏移”)。

如图 3 所示，增加事件数量使预取器能够覆盖更多的高速缓存未命中，同时提高预取的准确性。我们观察到最高的改善

张
2020-09-01 00:49:22
辅助存储器中的进入记录
完后，会将记录的模式发给
历史表格，然后自由的掉辅
助存储器中的入口.

两个事件(即“电脑+地址”和“电脑+偏移量”)，因为预取器的未命中覆盖率显著增加。然而，将事件数量增加到两个以上，并不会带来重大改善；因此，为了简单起见，我们

空间预取器，宾果。

IV. 宾果空间预取器

像以前的工作[18]一样，宾果使用一个小的辅助存储器来记录空间模式，同时处理器访问空间区域。在访问新页面(即触发访问)，宾果在其辅助存储器中为

并开始为它记录足迹。在页面驻留结束时(即，每当来自页面的块被无效或从高速缓存[18]中逐出时)，答对了

辅助存储器中的相应条目。

与以前的工作不同，宾果使用“电脑+地址”和“电脑+偏移”事件进行预取。宾果的简单实现需要两个不同的历史表，就像以前的类 TAGE 方法一样。一个表维护每个“个人电脑+地址”后观察到的足迹历史，而另一个表保存与“个人电脑+偏移量”相关联的足迹元数据在寻找预取模式时，逻辑上，首先，触发访问的“个人计算机+地址”用于搜索长历史表。如果找到匹配，则相应的占用空间被用于发出预取请求。否则，触发器访问的“PC+Offset”用于查找短历史表。如果匹配，匹配条目的足迹元数据将用于预取。如果没有找到匹配的条目，将不会发出预取。

带来巨大的存储开销。我们观察到，在空间数据预取的环境中，相当一部分

桌子是多余的。冗余是指两个元数据表(与长事件和短事件相关联的表)提供相同预测的情况。图 4 显示了用于空间预取的类似 TAGE 的历史表的减少。在这个实验中，每次空间预取器需要进行预测时，我们都会确定长事件和短事件是否提供相同的预测。如图所示，存在相当大的冗余，从 SAT Solver 中的 26%到MM32 中的 93%不等。

张
2020-09-01 00:53:52
设立两个历史表格，
给个人电脑+地址用于
测；个给电脑+偏移
短预测；该方法存储
大。

为了有效地消除元数据存储中的冗余，多个历史表，我们建议

每次都有不同的事件。图 5 详细说明了我们的实用

Authorized licensed use limited to: CHONGQING UNIVERSITY. Downloaded on August 31,2020 at 12:41:54 UTC from IEEE Xplore. Restrictions apply.

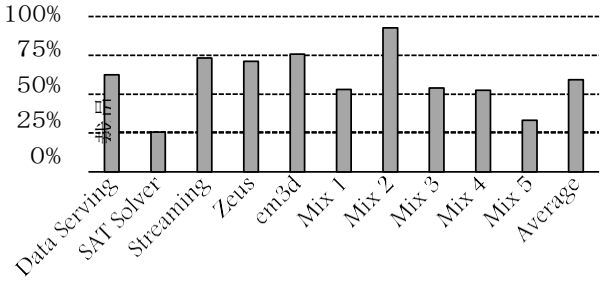


图 4。空间预取的类 TAGE 预测器的历史元数据中的冗余。冗余被定义为长事件和短事件提供相同预测的查找部分。

宾果游戏的设计，只使用一个历史表。主要思想是基于这样一个事实，即短事件是在长事件中进行的。也就是说，通过把长事件放在手边，我们可以找出短事件是什么，只需忽略长事件的一部分。对于宾果游戏，在“电脑+地址”中携带“电脑+偏移量”的信息

移量”是多少。为了利用这种现象，我们建议只有一个历史表

短期事件。对于宾果的情况，历史表存储在每个“个人电脑+地址”事件之后观察到的足迹，但是使用触发访问的“个人电脑+地址”和“个人电脑+偏移量”进行查找，以便提供高精度，同时不会失去预取机会。

为了实现这一点，我们发现表应该只

但是

每当一条信息要存储在历史元数据中时，它都与最长的事件相关联，然后存储在历史表中。为此，对应于最短事件的位被用于索引历史表，以找到应该存储元数据的集合；但是，最长事件的所有位都用于标记条目。更具体地说，在宾果游戏中，每当一个新的足迹将被存储在元数据组织中时，它就与相应的“个人电脑+地址”相关联为了在历史表中找到新条目的位置，仅使用“个人计算机+偏移量”的散列来索引表 6。通过知道集合，基线替换算法(例如，LRU)被用于选择受害者来打开用于存储新条目的房间。确定位置后，条目存储在历史表中，但“个人电脑+地址”的所有位都用于标记条目。每当需要预测时，历史表

5 这也适用于我们丢弃但没有用于宾果游戏的事件。当我们知道“电脑+地址”时，所有像“电脑”、“地址”和“偏移”这样的事件都是已知的此外，其他类似于 TAGE 的预测器也是如此，包括原始的 TAGE 分支预测器[44]，其中使用多个历史长度来索引元数据表。

6 重复地，对应于“PC+偏移量”的位被带入

’电脑+地址’

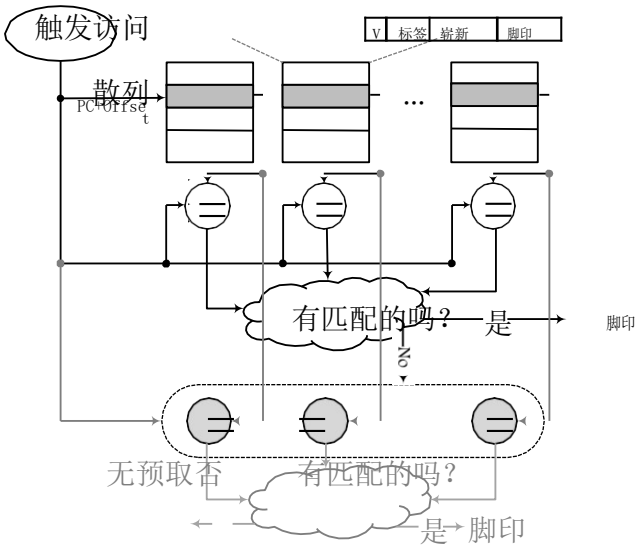


图 5。宾果预取器中历史表查找的详细信息。灰色部分表示使用长事件查找无法找到匹配的情况。每个大矩形表示历史表的一种物理方式。

首先是最长的事件；如果找到匹配，它将用于进行预测。否则，应该用倒数第二长的事件来查找表。因为长事件和短事件都映射到同一个集合，所以不需要检查新的集合；相反，测试相同集合的条目以找到与较短事件的匹配。

使用宾果，首先用触发访问的“电脑+地址”来查找表格。如果找到匹配项，相应的足迹元数据将用于发出预取请求。否则，应该使用触发器访问的“PC+Offset”来查找该表。我们了解

有对应的“PC+Offset”条目应该在同一个集合中。因此，我们测试同一个集合的条目来寻找匹配。然而，在这种情况下，并非条目中存储的标签的所有位都是匹配所必需的；只需要匹配“PC+偏移”位。这样，我们将每个足迹与一个以上的事件相关联(即“电脑+地址”和“电脑+偏移量”)，但将足迹元数据存储在表中，只与其中一个(较长的一个)相关联，以减少存储需求。这样做，冗余会自动消除，因为元数据足迹使用其“个人电脑+地址”标签存储一次。

在建议的设计中，无论何时查找具有较短事件的表，都有可能找到多个匹配。使用宾果，可能没有条目与触发访问的“电脑+地址”匹配，同时，有多个条目与访问的“电脑+偏移量”匹配。这种情况对宾果提出了挑战，因为它应该基于多

Authorized licensed use limited to: CHONGQING UNIVERSITY. Downloaded on August 31,2020 at 12:41:54 UTC from IEEE Xplore. Restrictions apply.

可能不同的 ple 足迹信息。在这种情况下，可以采用各种试探法:例如，基于最近信息 7 选择最近的足迹，或者对在所有匹配条目的足迹中指示的块发出预取请求。我们评估了许多这样的试探法，并根据经验发现，在发出预取请求时考虑所有匹配的占用空间信息可以获得最佳性能:如果缓存块存在于至少 20%匹配条目的占用空间中，则预取该缓存块。

V. 方法学

我们使用第二届数据预取锦标赛(DPC- 2) [72]中使用的模拟基础设施 champs i8[71]来精心模拟一个配置如表一所示的系统。我们基于英特尔最新的至强处理器[73]对系统进行建模。该芯片有四个 OoO 内核，带有一个 8 MB 共享末级高速缓存(LLC)。两个内存通道用于访问片外动态随机存取存储器，提供 37.5 GB/s 的最大带宽。操作系统使用 4 KB 页面，虚拟到物理地址的映射通过随机首触转换机制完成，支持长时间运行的模拟[74]。我们使用 CACTI 7.0 来估计缓存的延迟[75]。在整个内存层次结构中，缓存块大小为 64 字节。

表一评估参数。

参数	价值
芯片	14 纳米，4 千兆赫，4 核
核心	4 英寸宽 OoO，256 入口 ROB，64 入口 LSQ
读取部件	感知器[76]，16 项预调度队列
L1-发展/国际	分割输入/输出，64 KB，8 路，8 入口 MSHR
L2 高速缓存	8 MB、16 路、4 个存储体、15 周期命中延迟
主存储器	60 ns 零负载延迟，37.5 GB/s 峰值带宽

A. 工作量

表二总结了我们模拟工作负载的主要特征。根据文献[18]，[23]，[28]，[67]，我们选择了几个大数据服务器和科学应用进行评估。我们还考虑了来自一组内存密集型 SPEC 程序[77]的五个四核代表性混合工作负载，这些程序的执行性能对内存访问延迟高度敏感。

我们使用 SimFlex [78]方法来模拟服务器工作负载。对于每个服务器应用程序，我们创建了五个检查点，包括热缓存、分支预测器和预测表。每个检查点都是在一个时间间隔内绘制的

7 历史表中的条目(就像任何其他关联结构一样)存储一些替换位(例如，最近)，以便在集合已满且需要驱逐一个条目(例如，LRU)时帮助选择受害者。基于这些信息，我们可以在多个匹配中选择最近的条目。8 我们模拟器的源代码和评估过的数据预取器的实现可从以下网址获得
<https://github.com/bakhshalipour/Bingo >。

操作系统观察到的 10 秒模拟时间。然后我们从每个检查点运行 200 K 条指令，使用前 40 K 条指令来预热队列(例如 ROB)，其余的用于实际测量。基于 SimFlex [78]，我们的测量值以 95%的置信度和不到 4%的误差进行计算。对于 SPEC 基准测试，我们在每个内核上运行至少 100 M 指令的模拟，并使用前 20 M 作为预热，下 80 M 用于测量。

表二应用参数。

应用	描述	MPKI 有限责任公司
数据服务	卡珊德拉数据库，15GB 雅虎！基准	6.7
SAT 求解器	Cloud9 并行符号执行引擎	1.7
流动	达尔文流媒体服务器，7500 个客户端	3.9
宙斯	宙斯网络服务器 4.3 版，16 K 连接	5.2
em3d	40 万节点，2 级，跨度 5，15%远程	32.4
混合 1	lbm、omnetpp、soplex、sphinx3	15.7
混合 2	lbm，libquantum，sphinx3，zeusmp	12.5
混合 3	milc，omnetpp，perlbench，soplex	12.7
混合 4	astar、omnetpp、soplex、tonto	14.7
混合 5	GemsFDTD、gromacs、omnetpp、soplex	12.6

B. 预取器的配置

我们将我们的提议与最先进的空间数据预取器进行比较。对于每个预取器，我们都会执行敏感度分析，以便找到在我们的工作负载套件中提供合理的未命中覆盖率所需的存储。我们从最初工作中建议的配置开始，测量所有工作负载的平均未命中覆盖率。然后，我们增加元数据表的容量，以观察预取器对分配的存储有多敏感。如果
平均未命中覆盖率没有显著变化(> 5%)，我们将相同的存储分配给预取器，如中所述
最初的提议。否则，我们会增加预取器的存储，直到其平均未命中覆盖率达到稳定。接下来，我们将描述预取器的配置:

最佳偏移预取器:BOP [63]是最近的数据预取器，也是第二届数据预取锦标赛(DPC-2)的冠军[72]。防喷器基于以前的工作，即沙盒预取器[64]，并试图提高其及时性。每次进入时，防喷器测试单个偏移，以确定是否能够预测当前的进入。通过评估各种偏移量，它试图发现预计会产生及时预取的偏移量。我们用一个 256 条目的最近请求表来评估国际收支。

签名路径预取器:SPP [62]是另一个最近的数据预取器，它为每个增量偏移模式计算签名，并为每个签名估计增量预测的概率。通过利用这些概率，SPP 自适应地改变其预取

Authorized licensed use limited to: CHONGQING UNIVERSITY. Downloaded on August 31,2020 at 12:41:54 UTC from IEEE Xplore. Restrictions apply.

度，对 DRAM 模块的带宽压力较小。我们使用 256 项签名表、512 项模式表和 1024 项预取过滤器来评估 SPP。

可变长度增量预取器:VLDP [50]是最先进的空间数据预取器，它使用增量的多个历史来预测给定页面中的访问流。我们使用 16 项增量历史缓冲区、64 项偏移预测表和三个 64 项增量预测表来模拟 VLDP。

访问图模式匹配:AMPM [21]是另一个最先进的空间预取器，也是第一届数据预取锦标赛(DPC-1)的冠军[79]。AMPM 在一个名为

内存访问图。基于存储的信息，AMPM 检测出交错的访问模式，然后预测未来将被访问的块。我们扩大了内存访问映射表，以涵盖有限责任公司的全部容量。

空间内存流:短信是一个强大的空间数据预取器和我们的建议的基础。短信将足迹元数据与触发访问的“个人电脑+偏移量”相关联。短信已经远远超出了它的用途，在像时空这样的领域显示出巨大的潜力

时间预取[67]，确定商品动态随机存取存储器获取粒度[29]，以及管理管芯堆叠动态随机存取存储器高速缓存[36]，[80]，[81]。我们为短信配备了 16 K 条目 16 路关联历史表。

宾果:我们的建议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一个统一的历史表中。我们对历史表使用 16 路集合关联结构，并根据第六章-第一节中的敏感度分析设置其容量

我们在有限责任公司的背景下研究所有数据预取器，因为一个多兆字节有限责任公司的相当大的容量(与主缓存相比)为页面在这个级别的更长驻留铺平了道路。每当页面在缓存中停留很长时间时，就会释放更大的机会来访问该页面中的不同数据。这使得空间预取器能够完全观察每个页面的数据访问，并精确学习页面内访问模式[28]、[36]、[81]。我们认为每个内核都有自己的预取器，独立于其他内核(即内核之间没有元数据共享[30])。所有方法在有限责任公司访问时触发，并直接预取到有限责任公司(即，没有预取缓冲器[23]，[28])。

VI. 估价

A. *存储要求*

像任何其他数据预取器一样，宾果的有效性直接取决于进行预测的历史的大小。图 6 显示了当专用于其历史表的条目数量变化时，宾果的未命中覆盖率是如何变化的。随着历史表变得更大，未命中覆盖范围也增加，因为预取器能够将观察到的事件与遥远的过去历史进行比较，因此匹配的可能性增加。超过 16 K 的条目，覆盖范围停滞不前，有效地利用了可用的机会。因此，我们决定在宾果历史表中投入 16 K 个条目。使用 16 K 条目的历史表，预取器的总存储需求为 119 KB，仅占 LLC 容量的 6%。

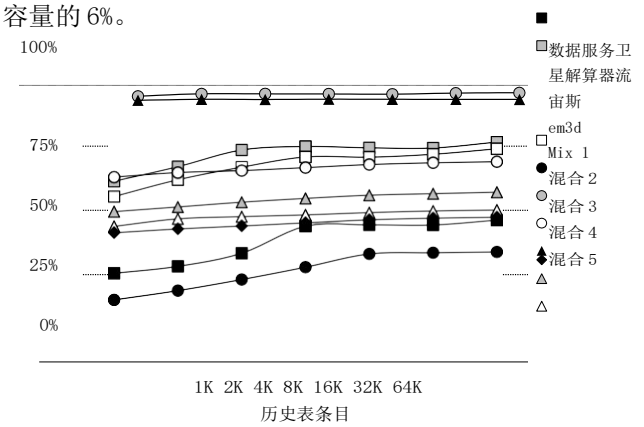


图 6。建议的数据预取器的未命中覆盖率是历史表中条目数的函数。

B. *错过承保范围和超额信贷*

为了评估所提出的预取器的有效性，图 7 显示了所有竞争预取技术的覆盖范围和过度预测。覆盖未命中是指预取器成功捕获的未命中。过度预测是不正确的预取，它被标准化为没有预取器的基线系统中数据未命中的数量。

如图所示，宾果在所有工作负载中提供了最高的未命中覆盖率。通过将足迹元数据与多个事件相关联，并匹配最长的事件，宾果最大限度地精确提取空间相关的数据访问模式，并显著减少缓存未命中的数量。平均而言，宾果覆盖了超过 63%的缓存未命中，比第二好的预取器高出 8%。宾果的超额预测与其他竞争的预取器不相上下。

证实了许多先前的工作(例如[3]，[23]，[28])，现代环境中的复杂访问模式

9 不要与准确性混淆，准确性是指所有预取中正确预取的比例。

Authorized licensed use limited to: CHONGQING UNIVERSITY. Downloaded on August 31,2020 at 12:41:54 UTC from IEEE Xplore. Restrictions apply.

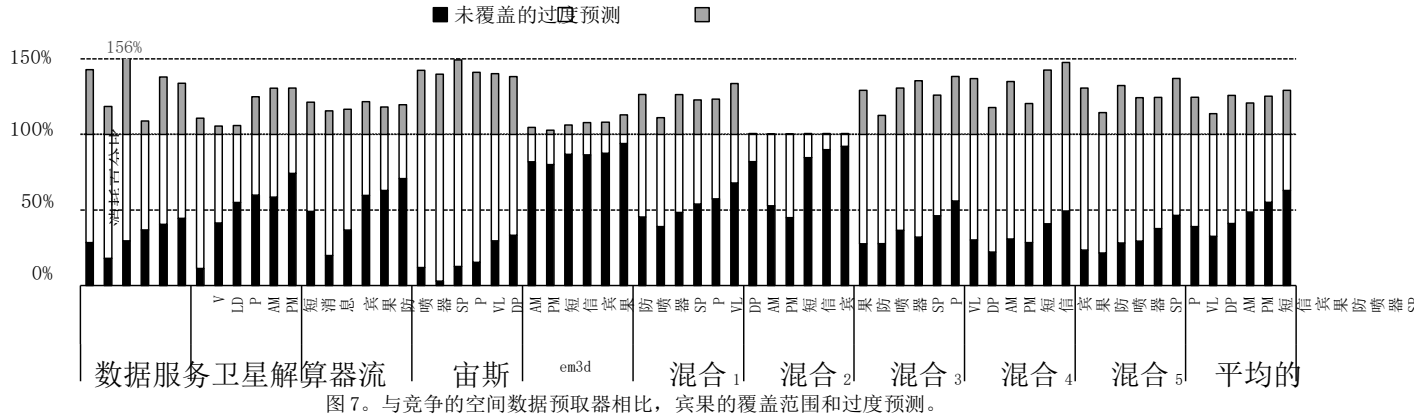


图7。与竞争的空间数据预取器相比，宾果的覆盖范围和过度预测。

服务器工作负载超出了简单的基于增量的方法所能捕获的范围。服务器工作负载重复地跨越各种数据结构，导致页面之间频繁切换(例如，在数据库缓冲池中[67])。当各种页面同时被遍历时，许多模式会在一个页面中表现出来，而不一定会在其他页面中表现出来；因此，基于SHH的空间预取器(例如BOP、SPP、VLDP)面临重大挑战，这使得它们无法提供高水平的未命中覆盖率。

像防喷器和防喷器中所提出的技术，当预取器提供高超调时，会降低预取器的速度。然而，通过这样做，它们也减少了未命中覆盖率，因为预取请求，包括正确的请求，是延迟发出的。VLDP使用三角洲的多个历史进行预测，因此提供了更高的未命中覆盖率。然而，它的多级预取效率很低。在预测页面中的下一次访问时，VLDP使用该预测作为历史表的输入来进行更多的预测。我们观察到这种策略对于服务器工作负载是不准确的，并且随着预取程度的增加会产生更多的过度预测10。通过维护每个页面的足迹元数据，AMPM和短信提供了比其他先前的预取器高一个数量级的覆盖率。然而，短信比AMPM更激进，导致更高的过度预测和更高的误报率。短信将足迹与触发访问的“个人电脑+偏移量”相关联，并在再次获得相同的“个人电脑+偏移量”时应用所学习的足迹。然而，正如我们所展示的，“个人电脑+偏移量”不足以提供高精度。

C. 系统性能

图8显示了宾果游戏的性能提升以及其他预取技术没有预取器。宾果在所有工作负载中始终优于竞争预取方法。宾果的性能提升幅度从宙斯的11%到em3d的285%。未命中覆盖率、及时性和预取的准确性是宾果卓越性能提升的主要因素。对于大多数工作负载，宾果提供了显著的性能提升。然而，在宙斯中，内存访问在时间上比空间上更相关[28]。甚至那些空间上可预测的进程也已经被乱序处理并行获取，导致空间预取器的性能显著提高。

由于未命中覆盖率低，与其他方法相比，BOP和SPP等技术提供的性能提升更低，尤其是在服务器工作负载方面。VLDP提供了更高的性能，主要是因为有更好的脱靶量。然而，它落后于基于PPH的方法，如AMPM和短信，这些方法保留页面足迹，并使用它为所有预期使用的缓存块发出更准确的预取(参见第二节)。虽然AMPM和短信提供了高水平的性能提升，但它们的性能仍然明显低于宾果。通过将足迹元数据与多个事件相关联并匹配最长的事件，宾果最大限度地、准确地提取空间相关的数据访问模式，并将其用于准确、及时的预取。

D. 性能密度

许多先前的工作[8]，[9]，[30]，[83]认为，硬件优化器的性能增益，像预取器一样，应该超过它们的面积开销。否则，使用硅不动产来进一步增加内核数量可能会更有利。先前的工作使用了一个称为性能密度的指标，定义为单位面积的吞吐量，来量化设计如何有效地使用硅片:只有当硬件预取器能够提高性能密度时，将硬件预取器集成到系统中才是有益的[8]，[30]。

10 在数据预取的背景下，VLDP的前期工作[28]，[62]也做出了这一观察。此外，Kolli等人[82]在服务器工作负载的指令预取环境中，对这种策略的不准确性进行了类似的观察。

Authorized licensed use limited to: CHONGQING UNIVERSITY. Downloaded on August 31,2020 at 12:41:54 UTC from IEEE Xplore. Restrictions apply.

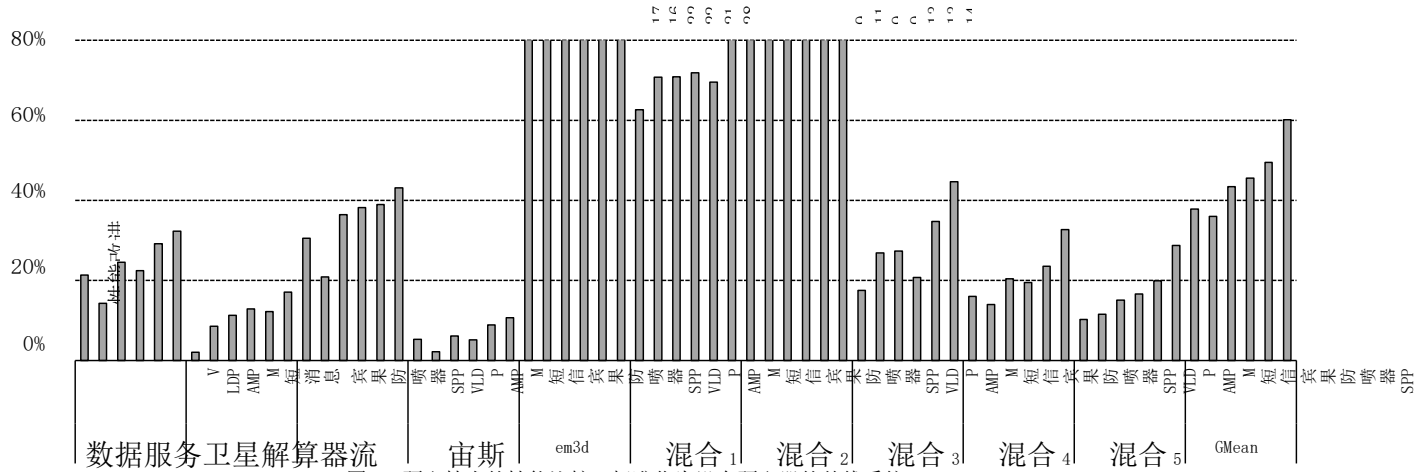


图8。预取技术的性能比较，标准化为没有预取器的基线系统。

由于被评估的预取器的面积要求各不相同，为了更好地了解它们如何有效地使用芯片硅面积，我们评估了它们的性能密度。图9总结了这项研究的结果，显示了与没有预取器的基线系统相比，所有评估设计的性能密度改进。对于这个实验，我们只考虑内核、缓存、互连和内存通道的面积，而忽略了输入/输出设备的面积。

宾果将基线系统的性能密度提高了59%，比性能第二好的方法高10%。与其他预取器相比，宾果的性能密度提升略低于其性能提升。主要原因是与其他方法的元数据表相比，宾果的历史表更大。然而，由于宾果的总面积开销不到有限责任公司面积的6%和整个芯片面积的一小部分，性能密度的下降是微不足道的(小于1%)。结果表明，宾果的性能提高远远超过其存储开销，使其成为一个实用的

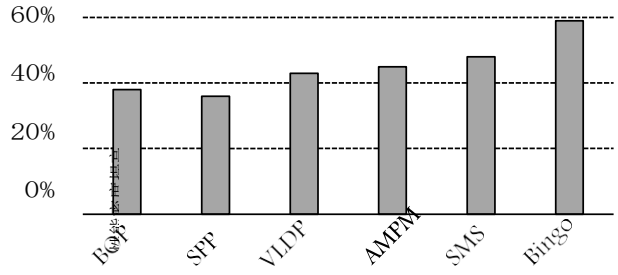


图9。与没有预取器的基线系统相比，被评估方法的性能密度提高。

高性能处理器的设计。

E. 等度比较

当然，宾果和其他基于PPH的方法相对于基于SHH的方法的性能提升很大一部分来自于它们更好的及时性。宾果，像其他基于PPH的方法一样，为每一页收集一个足迹，并一次为该页的所有预期要使用的块发出预取请求，从而获得卓越的及时性。然而，在基于SHH的方法中，没有关于页面足迹的信息；因此，这些方法不知道每当它们被触发时应该发出多少预取。先前的工作采用了各种机制，以便在基于SHH的方法的上下文中实现多级预取。例如，VLDP在预测页面中的下一次访问时，使用该预测作为历史表的输入来发布更多预取。通常，预取的程度和它的预测精度之间有一个折衷:预取器越激进，预期的错误预取就越多。为了应对多级预取的过度预测，使用各种试探法的先前工作尝试限制多级预取的数量。例如，VLDP允许预取度高达4。

在本节中，为了提供一个等度比较，我们解除了禁令，并允许基于SHH的预取器在每次激活时发出进一步的预取请求。我们将防喷器/VLDP的最大程度设置为32，并将SPP的置信阈值[62]设置为1%。图10显示了这个实验的结果。由于时效性更好，基于SHH的方法性能略有提升；然而，由于许多错误的预取，它们的过度预测同样会增加，从而导致严重的片外带宽污染。使用等度预取，宾果仍然明显优于所有其他方法。

Authorized licensed use limited to: CHONGQING UNIVERSITY. Downloaded on August 31,2020 at 12:41:54 UTC from IEEE Xplore. Restrictions apply.

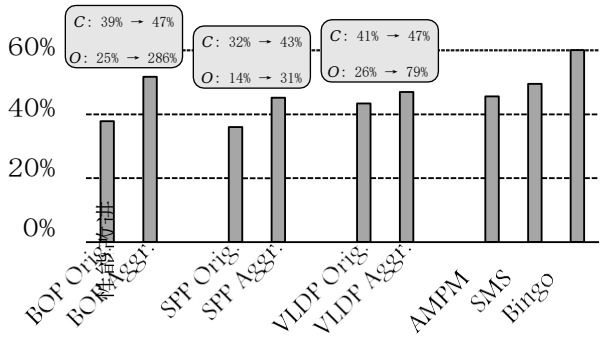


图 10。预取方法的等度比较。Orig ’表示基于 SHH 的预取器的原始版本和目前为止评估的版本；然而，‘Aggr’代表了侵略性和高度的版本。标注表示预取器的覆盖范围和超预测如何从原始版本变化到激进版本:“C”和“O”分别代表“覆盖范围”和“超预测”。

VII. 相关著作

最近的许多工作针对硬件数据预取器环境中的长延迟内存停顿。国际军事存在[84]
从形式为 A[B[i]]的间接模式中识别和预取不规则的内存访问，这在许多应用程序中非常丰富，如图形分析和稀疏线性
代数。TEMPO [85]增加了内存控制器来跟踪对页表条目的访问，以便预取翻译后访问。Domino [28]使用一个或两个最后数据未命中的组合来在历史中找到正确的临时地址流，并预取后续的数据未命中。B-Fetch [86]利用分支预测器在执行程序之前运行，从而沿着预期的未来路径预取加载指令。最近的一些研究[87]，[88]也评估了深度学习算法在内存访问预测中的应用。这些方法都不依赖于空间相关性，因此可以与我们的建议正交使用。

几种方法在数据预取的上下文之外使用空间模式预测器。SFP [17]使用基于空间足迹的预测器来识别高速缓存块的有用字，并将这些字存储在解耦的二级高速缓存中[89]。BuMP [29]通过依赖识别打开的 DRAM 行的使用密度的空间占用预测器来确定 DRAM 获取粒度。足迹高速缓存[36]，[80]，[81]提出将管芯堆叠的动态随机存取存储器作为具有多块提取粒度的基于页面的高速缓存来管理。内存占用缓存方法利用空间内存占用预测器来获取和缓存每一页中预期要使用的缓存块，从而降低动态随机存取存储器模块的带宽压力。我们的建议可以被结合到这样的方案中，以实现更高的预测效率。

VIII. 结论

长延迟片外未命中通常会使处理器停滞以等待数据到达，这是大数据应用程序性能下降的主要原因。空间数据预取是一种减少缓存未命中次数或其有害影响的技术。空间数据预取器利用了这样一个事实，即数据访问在几千字节的存储区域上是空间相关的，并且这种相关性是可预测的。在这项工作中，我们展示了最先进的空间数据预取器没有充分利用现有的机会，因为只将足迹元数据与单个事件相关联。为了提高空间预取的覆盖率和准确性，我们提出了一种将足迹元数据与多个事件相关联的实用方法。此外，我们建议了一种通用机制来消除基于历史的预测元数据表中的冗余。我们表明，所提出的空间预取器明显优于竞争的空间预取器。

承认

我们感谢匿名评论者的宝贵意见。我们感谢 PARSA-EPFL 公司的马克·萨瑟兰为我们提供了在我们的框架中模拟服务器工作负载所需的工具。我们感谢 IPM 高性能计算中心的成员维护和管理用于进行实验的集群。

参考

[1] T.“一阶超标量处理器模型”，载于《计算机体系结构国际研讨会论文集》(ISCA)，第 338-349 页，2004 年。

[2] R.Hameed，W. Qadeer，M. Wachs，O. Azizi，A. Solomatnikov，B. C. Lee，
南理查森、科兹拉基斯和霍洛维茨，“理解通用芯片低效率的根源”，载于《计算机体系结构国际研讨会论文集》(ISCA)，第 37-47 页，2010 年。

[3] 米（meter 的缩写））Ferdman，A. Adileh，O. Kocberber，S. Volos，M. Alisafaee，D. Jevdjic，
C.Kaynak，A. D. Popescu，A. Ailamaki 和 B. Falsafi，“清除云:现代硬件上新兴横向扩展工作负载的研究”，载于《编程语言和操作系统架构支持国际会议(ASPLOS)会议录》，第 37–48 页，2012 年。

[4] 米（meter 的缩写））Ferdman，A. Adileh，O. Kocberber，S. Volos，M. Alisafaee，D. Jevdjic，
C.Kaynak，A. D. Popescu，A. Ailamaki 和 B. Falsafi，“量化新兴横向扩展应用与现代处理器之间的不匹配”，《计算机系统上的 ACM 事务》(TOCS)，第 30 卷，第 15:1–15:24 页，2012 年 11 月。

[5] A.Mirhosseini，A. Sriraman 和 T. F. Wenisch，“面对黑仔微秒提高服务器效率”，高性能计算机体系结构国际研讨会论文集(HPCA)，2019 年。

[6] 米（meter 的缩写））哈希米，e．易卜拉希米，o．穆特鲁和 Y. N．派特，“用增强的内存控制器加速相关的高速缓存未命中”，载于《计算机体系结构国际研讨会论文集》(ISCA)，第 444-455 页，2016 年。

[7] 米（meter 的缩写））哈希米、穆特鲁和派特，“连续提前运行:内存密集型工作负载的透明硬件加速”，载于《微体系结构国际研讨会论文集》，第 61:1–61:12 页，2016 年。

Authorized licensed use limited to: CHONGQING UNIVERSITY. Downloaded on August 31,2020 at 12:41:54 UTC from IEEE Xplore. Restrictions apply.

[8] 页（page 的缩写）洛特菲-卡姆兰，b .格罗特，m .费尔德曼，s .沃洛斯，o .考伯，j .皮科雷尔，
A.Adileh，D. Jevdjic，S. Idgunji，E. Ozer 和 B. Falsafi，“横向扩展处理器”，载于《计算机体系结构国际研讨会论文集》(ISCA)，第 500-511 页，2012 年。

[9] 页（page 的缩写）埃斯迈里-多克特、巴赫夏利普尔、霍达班德洛、洛特菲-卡姆兰和
H.Sarbazi-Azad，“横向扩展处理器与能效”，arXiv 预印本 arXiv:1808.04864，2018。

[10] 名词（noun 的缩写）“反应型 NUCA:分布式高速缓存中接近最优的块放置和复制”，载于《计算机体系结构国际研讨会论文集》(ISCA)，第 184-195 页，2009 年。

[11] D.Guttman，M. T. Kandemir，M. Arunachalamy 和 V. Calina，“基于英特尔至强融核的数据预取的性能和能量评估”，载于《系统和软件性能分析国际研讨会论文集》(ISPASS)，第 288 - 297 页，2015 年。

[12] 动词 （verb 的缩写）姬广亮 • 内兹，r .乔约萨，F. J .卡佐拉，A. Buyuktosunoglu，P. Bose 和 F. P. O'Connell，“让数据预取更智能:POWER7 上的自适应预取”，载于《并行体系结构和编译技术国际会议论文集》(PACT)，第 137 - 146 页，2012 年。

[13] H.康和王俊林:“硬件预取还是不预取？:虚拟化环境研究和核心绑定方法，“编程语言和操作系统架构支持国际会议(ASPLOS)记录”，第 357 - 368 页，2013 年。

[14] A.索达尼、R. Gramunt、J. Corbal、h . s . Kim、K. Vinod、S. Chinthamani、
南 Hutsell，R. Agarwal 和 y-c . Liu，“骑士登陆:第二代英特尔至强融核产品”，IEEE Micro，第 36 卷，第 2 期，第 34 - 46 页，2016 年。

[15] 页（page 的缩写）康威和休斯，“AMD 皓龙北桥架构”，IEEE 微，第 27 卷，第 2 期，第 10-21 页，2007 年。

[16] T.Horel 和 G. Lauterbach，“UltraSPARC-III:设计第三代 64 位性能”，IEEE Micro，第 19 卷，第 3 期，第 73 - 85 页，1999 年。

[17] 南 Kumar 和 C. Wilkerson，“利用空间足迹开发数据缓存中的空间局部性”，载于《计算机体系结构国际研讨会论文集》(ISCA)，第 357-368 页，1998 年。

[18] 南索莫吉，韦尼施，艾拉马基，法尔萨菲和莫索沃斯，“空间记忆流”，载于《计算机体系结构国际研讨会论文集》(ISCA)，第 252-263 页，2006 年。

[19] J.F. Cantin，M. H. Lipasti 和 J. E. Smith，“隐形预取”，载于《编程语言和操作系统架构支持国际会议论文集》(ASPLOS)，第 274 - 282 页，2006 年。

[20] C.陈，杨世海，法萨菲，莫索沃斯，“精确和复杂有效的空间模式预测”，载于《高性能计算机体系结构国际研讨会论文集(HPCA)》，2004 年，第 276-287 页。

[21] Y.Ishii，M. Inaba，K. Hiraki，“数据缓存预取的访问图模式匹配”，载于《国际超级计算会议论文集》，第 499 - 500 页，2009 年。

[22] Y.索利欣、李和托里拉斯，“使用用户级内存线程进行相关预取”，载于《计算机体系结构国际研讨会论文集》(ISCA)，第 171-182 页，2002 年。

[23] T.索莫吉，哈达威拉斯，金，艾拉马基和法萨菲，“共享内存的时间流”，《计算机体系结构国际研讨会论文集》(ISCA)，第 222-233 页，2005 年。

[24] 米（meter 的缩写））Ferdman 和 B. Falsafi，“最后接触相关数据流”，在系统和软件性能分析国际研讨会(ISPASS)的会议记录中，第 105-115 页，2007。

[25] Y.周，“面向商业应用的低成本基于时代的相关预取”，载于《微体系结构国际研讨会论文集》，第 301-313 页，2007 年。

[26] T.费曼、法萨菲和莫索沃斯，“用于时间存储流的实用片外元数据”，载于《高性能计算机体系结构国际研讨会论文集》(HPCA)，第 79-90 页，2009 年。

[27] A.Jain 和 C. Lin，“线性化不规则内存访问以改进相关预取”，载于《微体系结构国际研讨会论文集》，第 247-259 页，2013 年。

[28] 米（meter 的缩写））Bakhshalipour，p .洛特菲-卡姆兰和 H. Sarbazi-Azad，“多米诺时态数据预取器”，载于《高性能计算机体系结构国际研讨会论文集》(HPCA)，第 131-142 页，2018 年。

[29] 南 Volos，J. Picorel，B. Falsafi 和 B. Grot，“BuMP:大容量内存访问预测和流”，载于《微体系结构国际研讨会论文集》(MICRO)，第 545 - 557 页，2014 年。

[30] C.Kaynak，B. Grot 和 B. Falsafi，“SHIFT:瘦核服务器处理器的共享历史取指令”，载于《微体系结构国际研讨会论文集》，第 272 - 283 页，2013 年。

[31] B.M. Rogers，A. Krishna，G. B. Bell，K. Vu，X. Jiang 和 Y. Solihin，“缩放带宽墙:CMP 缩放的挑战和途径”，载于《计算机体系结构国际研讨会论文集》(ISCA)，第 371-382 页，2009 年。

[32] J.Huh，D. Burger 和 S. W. Keckler，“探索未来 CMPs 的设计空间”，载于《并行架构和编译技术国际会议论文集》(PACT)，第 199-210 页，2001 年。

[33] 米（meter 的缩写））Bakhshalipour，A. Faraji，S. A. Vakil Ghahani，F. Samandi，p .洛特菲-卡姆兰和 H. Sarbazi-Azad，“通过缓存内置换减少写回”，ACM 电子系统设计自动化交易(TODAES)，2019。

[34] 米（meter 的缩写））扎雷亚、洛特菲-卡姆兰和萨尔巴兹-阿扎德，“芯片堆叠的动态随机存取存储器:内存、高速缓存还是内存高速缓存？”，arXiv 预印本 arXiv:1809.08828，2018。

[35] H.A. Esfeden，F. Khorasani，H. Jeon，D. Wong 和 N. Abu-Ghazaleh，“CORF:合并 GPU 的操作数寄存器文件”，载于《编程语言和操作系统的架构支持国际会议论文集》(ASPLOS)，2019 年。

[36] D.服务器的芯片堆叠式动态随机存取存储器:命中率、延迟还是带宽？足迹缓存拥有一切”，载于《计算机体系结构国际研讨会论文集》(ISCA)，第 404-415 页，2013 年。

[37] E.易卜拉希米，李振杰，穆特鲁和派特，“多核系统的预取感知共享资源管理”，载于《计算机体系结构国际研讨会论文集》(ISCA)，第 141-152 页，2011 年。

[38] E.易卜拉希米，o .穆特鲁，C. J .李和 Y. N .派特，“多核系统中多个预取器的协调控制”，载于《微体系结构国际研讨会论文集》，第 316-326 页，2009 年。

[39] E.易卜拉希米，O. Mutlu 和 Y. N. Patt，“混合预取系统中链接数据结构的带宽高效预取技术”，载于《高性能计算机体系结构国际研讨会论文集》(HPCA)，第 7-17 页，2009 年。

[40] C.李，穆特鲁，纳拉西曼和派特，“预取感知动态随机存取存储器控制器”，载于微体系结构国际研讨会论文集，第 200-209 页，2008 年。

[41] 南 Srinath，O. Mutlu，H. Kim 和 Y. N. Patt，“反馈引导预取:提高硬件预取器的性能和带宽效率”，载于《高性能计算机体系结构国际研讨会论文集》(HPCA)，第 63-74 页，2007 年。

[42] 名词（noun 的缩写）Hardavellas，M. Ferdman，B. Falsafi 和 A. Ailamaki，“迈向服务器中的黑硅”，第 31 卷，第 4 期，第 6 - 15 页，2011 年。

[43] 米（meter 的缩写））巴赫夏利普尔，洛特菲-卡姆兰，马祖卢米，萨曼迪，纳德兰，
米（meter 的缩写））Modarressi 和 H. Sarbazi-Azad，“多核处理器的快速数据传递”，IEEE 计算机事务(TC)，第 67 卷，第 10 期，第 1416 - 1429 页，2018 年。

[44] A.(部分)标记的几何历史长度预测的案例
《指令级并行性杂志》(JILP)，2006 年。

[45] J.Cleary 和 I. Witten，“使用自适应编码和部分字符串匹配的数据压缩”，IEEE 通信事务(TCOM)，第 32 卷，第 4 期，第 396-402 页，1984 年。

[46] A.Seznec，“TAGE 分支预测器的新案例”，载于《微体系结构国际研讨会论文集》，第 117-127 页，2011 年。

[47] A.圣米格尔和阿尔伯里西奥，“最内部循环迭代计数器:分支历史的一个新维度”，载于国际微体系结构研讨会论文集，第 347-357 页，2015 年。

Authorized licensed use limited to: CHONGQING UNIVERSITY. Downloaded on August 31,2020 at 12:41:54 UTC from IEEE Xplore. Restrictions apply.

[48] 页（page 的缩写）Michaud，“一种替代的类似于 TAGE 的条件分支预测器”，ACM 架构和代码优化事务(TACO)，第 15 卷，第 30:1 – 30:23 页，2018 年 8 月。

[49] 米（meter 的缩写））Bakhshalipour，p .洛特菲-卡姆兰和 h .萨尔巴兹-阿扎德，“一种用于 L1 高速缓存的高效速度数据预取器”，《IEEE 计算机体系结构通讯》，第 16 卷，第 2 期，第 99 – 102 页，2017 年。

[50] 米（meter 的缩写））Shevgoor，S. Koladiya，R. Balasubramonian，C. Wilkerson，S. H. Pugsley，和
Z.Chishti，“高效预取复杂地址模式”，载于《微体系结构国际研讨会论文集》，第 141-152 页，2015 年。

[51] A.Perais 和 A. Seznec，“未来高端处理器的实用数据值推测”，载于《高性能计算机架构国际研讨会论文集》(HPCA)，第 428-439 页，2014 年。

[52] A.佩莱斯和塞兹奈克，“EOLE:为有效实现价值预测铺平道路”，载于《计算机体系结构国际研讨会论文集》(ISCA)，第 481-492 页，2014 年。

[53] A.Perais 和 A. Seznec，“BeBoP:用于超标量值预测的经济有效的预测器基础设施”，载于《高性能计算机体系结构国际研讨会论文集》(HPCA)，第 13-25 页，2015 年。

[54] A.Perais 和 A. Seznec，“成本有效的物理寄存器共享”，载于《高性能计算机体系结构国际研讨会论文集》(HPCA)，第 694-706 页，2016 年。

[55] A.Perais，F. A. Endo 和 A. Seznec，“寄存器共享促进平等预测”，载于《微体系结构国际研讨会论文集》，第 4:1 – 4:12 页，2016 年。

[56] J.Sim，G. H. Loh，H. Kim，M. O 'Connor 和 M. Thottethodi，“一种用于有效命中推测和自平衡调度的基本干净的 DRAM 高速缓存”，载于《微体系结构国际研讨会论文集》，第 247 – 257 页，2012 年。

[57] D.Mahajan，A. Yazdanbakhsh，J. Park，B. Thwaites 和 H. Esmaeilzadeh，“近似加速器基于预测的质量控制”，在跨系统堆栈近似计算研讨会(WACAS)上，2015。

[58] K.Kuroyanagi 和 A. Seznec，“通过估计请求权重和使用每线程交通灯的服务价值感知内存调度器”，内存调度锦标赛(MSC)，2012 年。

[59] 名词（noun 的缩写）预测指令的选择性预测和重放。博士论文，INRIA，2013 年。

[60] F.Khorasani，H. A. Esfeden，N. Abu-Ghazaleh 和 V. Sarkar，“具有虚拟持久处理器专门化的动态神经网络的寄存器内参数缓存”，载于国际微阵列技术研讨会论文集(MICRO)，2018 年。

[61] A.瓦基勒-加哈尼、马赫迪扎德-沙赫里、洛特菲-那敏、巴克夏利普尔、
页（page 的缩写）洛特菲-卡姆兰和 h .萨尔巴兹-阿扎德，“基于预期命中数的缓存替换策略”，《IEEE 计算机体系结构快报》，第 17 卷，第 1 期，第 64 – 67 页，2018 年。

[62] J.金，S. H .普利斯利，P. V .格拉茨，A. L. N .雷迪，c .威尔克森，和
Z.Chishti，“基于路径置信度的前瞻预取”，载于《微体系结构国际研讨会论文集》，第 60:1 – 60:12 页，2016 年。

[63] 页（page 的缩写）米肖，“最佳偏移硬件预取”，载于《高性能计算机体系结构国际研讨会论文集》(HPCA)，第 469-480 页，2016 年。

[64] 南 H. Pugsley，Z. Chishti，C. Wilkerson，p-f . Chuang，R. L. Scott，A. Jaleel，s-l . Lu，K. Chow，和 R. Balasubramonian，“沙盒预取:攻击性预取器的安全运行时评估”，《高性能计算机体系结构国际研讨会论文集(HPCA)》，第 626-637 页，2014 年。

[65] K.“交流/DC:自适应数据高速缓存预取器”，载于《并行体系结构和编译技术国际会议论文集》，第 135-145 页，2004 年。

[66] K.“使用全局历史缓冲区的数据高速缓存预取”，载于《高性能计算机体系结构国际研讨会论文集》(HPCA)，第 96-96 页，2004 年。

[67] 南索莫吉，韦尼施，艾拉马基和法萨非，“时空记忆流”，载于《计算机体系结构国际研讨会论文集》(ISCA)，第 69-80 页，2009 年。

[68] J.-L. Baer 和 t . f . Chen，“减少数据访问损失的有效片上预加载方案”，载于《国际超级计算会议论文集》，第 176-186 页，1991 年。

[69] 南“有效的基于流和基于执行的数据预取”，载于《国际超级计算会议论文集》，第 1-11 页，2004 年。

[70] 名词（noun 的缩写）P. Jouppi，“通过增加一个小的全关联高速缓存和预取缓冲器来提高直接映射高速缓存的性能”，载于《计算机体系结构国际研讨会论文集》(ISCA)，第 364-373 页，1990 年。

[71] “ChampSim。”https://github.com/ChampSim/，2017。

[72] 南帕格斯利、阿拉梅尔登、威尔克森和金，“第二届数据预取锦标赛(DPC-2)”，2015 年。

[73] “英特尔至强处理器 E3-1220 v6。”https://www.intel.com/content/www/us/en/产品/处理器/至强/e3-处理器/e3-1220-v6.html/，2017 年。

[74] 南 Franey 和 M. Lipasti，“标签表”，载于《高性能计算机体系结构国际研讨会论文集》(HPCA)，第 514-525 页，2015 年。

[75] ″ CACTI 7.0:一个模拟高速缓存/存储器、三维堆叠和片外输入输出的工具.″https://github.com/HewlettPackard/cacti/，2017。

[76] D.和林，“用感知器进行动态分支预测”，《高性能计算机体系结构国际研讨会论文集》，，第 197-206 页，2001 年。

[77] J.亨宁，“SPEC CPU2006 基准描述”，ACM SIGARCH 计算机体系结构新闻，2006。

[78] T.F. Wenisch，R. E. Wunderlich，M. Ferdman，A. Ailamaki，B. Falsafi，和
J.计算机系统模拟的统计抽样
IEEE Micro，第 26 卷，第 4 期，第 18-31 页，2006 年。

[79] ″第一届 JILP 数据预取锦标赛(DPC-1). ″https://www. jilp 或 g/IPC/online/papers/IPC-1-intro . pdf/，2009 年。

[80] H.张永利，金永利，金永利，郑永利，李永利，“无标记动态随机存取存储器的有效足印高速缓存”，载于《高性能计算机体系结构国际研讨会论文集》(HPCA)，第 237-248 页，2016 年。

[81] D.Jevdjic，G. H. Loh，C. Kaynak 和 B. Falsafi，“Unison Cache:一种可扩展且有效的芯片堆叠 DRAM 缓存”，载于《微体系结构国际研讨会论文集》(MICRO)，第 25 – 37 页，2014 年。

[82] A.Kolli，A. Saidi 和 T. F. Wenisch，“RDIP:返回地址堆栈指导的指令预取”，载于《微体系结构国际研讨会论文集》，第 260-271 页，2013 年。

[83] 页（page 的缩写）洛特菲-卡姆兰，m .莫达雷西和 h .萨尔巴兹-阿扎德，“服务器的近理想片上网络”，载于《高性能计算机体系结构国际研讨会论文集》(HPCA)，第 277-288 页，2017 年。

[84] X.余，休斯，萨蒂什和德瓦达斯，“IMP:间接存储器预取器”，载于《微体系结构国际研讨会论文集》，第 178-190 页，2015 年。

[85] A.Bhattacharjee，“翻译触发预取”，载于《编程语言和操作系统架构支持国际会议论文集》(ASPLOS)，第 63 – 76 页，2017 年。

[86] D.Kadjo，J. Kim，P. Sharma，R. Panda，P. Gratz 和 D. Jimenez，“B-Fetch:面向芯片多处理器的分支预测定向预取”，载于《微体系结构国际研讨会论文集》(MICRO)，第 623 – 634 页，2014 年。

[87] 米（meter 的缩写））哈希米，斯沃斯基，艾耶斯，李兹，张杰，科兹拉基斯和阮冈纳赞，“学习记忆访问模式”，在国际机器学习会议(ICML)，2018 年。

[88] 长度佩雷德，美国魏泽，和 Y. Etsion，“用神经网络实现内存预取:挑战和见解”，arXiv 预印本 arXiv:1804.00478，2018。

[89] A.Seznec，“解耦分区高速缓存:调和低标签实现成本”，载于《计算机体系结构国际研讨会论文集》(ISCA)，第 384-393 页，1994 年。

Authorized licensed use limited to: CHONGQING UNIVERSITY. Downloaded on August 31,2020 at 12:41:54 UTC from IEEE Xplore. Restrictions apply.