减少 Swizzling 的虚拟内存开销

Vivek Narasaya、Tze Sing Eugene Ng、Dylan McNamee、Ashutosh Tiwary、Hank Levy 华盛顿大学
（西雅图）。奈良，尤金能，迪伦，蒂瓦里，
{征@cs.washington.edu }

抽象的

Swizzling 是 OODB 和持久对象系统使用的一种机制，用于将指针从磁盘格式转换为更有效的内存格式。先前对 swizzling 的研究主要集中在分析指针转换的 CPU 开销以及研究不同 swizzling 方法的权衡。在本文中，我们证明了与调配相关的额外间接但重要的成本：调配只读页面会导致其对于操作系统来说是“脏”的。在分页开始时，这些只读页可能会不必要地写入交换文件。我们建议对操作系统进行简单的修改，以减少这种开销对应用程序性能的影响。

1 简介

近年来，面向对象数据库（OODB)的出现是为了满足计算机辅助设计和制造（CAD/CAM）、计算机辅助软件工程（CASE）、多媒体和网络管理等应用的需求。大多数 OODB 在访问内存中的持久对象时使用混合来提高性能。当 OODB 将持久对象放入内存时，调配会将指针从磁盘格式（对象标识符)转换为更有效的内存格式（虚拟内存地址）。

Swizzling 是一项具有多种维度的复杂技术 [7]。之前的调配研究分析了指针转换的成本 [5] 并探索了诸如软件与虚拟内存硬件调配、急切与惰性调配以及对象粒度与页面粒度调配等权衡 [8]。这些研究没有识别或量化由于调配页面与操作系统 (OS) 交互而发生的调配的重要间接成本。由于调配会修改页面上的指针，因此操作系统会认为调配页面是脏的，即使应用程序可能将其视为只读。当机器上的内存压力导致分页时，这些脏页可以被调出到不必要的交换文件。对于访问大型[1]数据集的应用程序，这种分页成本可能会主导其他调配成本并影响应用程序性能。我们将这些不必要的页面调出称为swizzling 的虚拟内存开销（VM 开销），因为它们是由 swizzling 和虚拟内存系统的交互引起的。

在下一节中，我们将在 Texas [6] 的背景下描述和量化 VM 开销问题，这是一个使用虚拟内存硬件进行指针混合的 OODB。

在第 3 节中，我们提出了解决此问题的方法，其中涉及对操作系统进行小的更改，并认为它可以用于提高主要只读工作负载的性能。在第 4 节中，我们讨论如何使用我们的技术来解决具有相同一般性质的其他问题。我们在第 5 节中进行总结和总结。

2 量化虚拟机开销

VM 开销可能出现在任何混合的 OODB 中。在本节中，我们将描述并量化德克萨斯州 OO7 工作负载的虚拟机开销。

2.1 Swizzling and Buffer Management in Texas

Texas 在页面错误时使用指针调配 [10] 来提供从持久内存到虚拟内存的映射。为了确保对页面的第一次访问被拦截，Texas 访问会保护该页面。当发生页面错误时，德克萨斯州通过用虚拟地址覆盖持久格式的指针来混合页面。由于调配会覆盖页面上的指针，因此即使应用程序从未直接修改页面，该页面对于 VM 系统而言也会变脏。德克萨斯州没有明确控制数据库页面的缓存。它将数据库页读入文件系统缓冲区，将其从那里复制到应用程序的内存中，然后混合它。这些脏页

[1] 请注意，许多 OODB 尝试通过管理自己的缓冲池来控制应用程序数据的分页。然而，如果操作系统将缓冲池中的脏页换出，即使这样的系统也可能执行不必要的分页。第 4 节对此进行了描述。

由交换文件支持并由操作系统管理。

我们确定了德克萨斯州调酒的三个主要成本：

- 信号处理成本。这是向德克萨斯州信号处理程序传送信号的成本。这显示为应用程序的 CPU 时间。

- 指针翻译成本。这是在页面中翻译指针的成本。它包括在页面上定位对象、获取每个对象的类型信息、可能为该页面中的指针指向的页面保留虚拟地址空间、记录持久页面和虚拟页面之间的映射以及最终将对象标识符转换为虚拟地址的成本。内存指针。这再次显示为应用程序的 CPU 时间。

- 虚拟机开销。当应用程序的只读页面在德克萨斯州发生故障并混合时，就会发生这种情况，从而使页面变脏。[2] 由于分页，该只读页可能会被写入交换文件。一个关键的观察结果是，该只读页面可能已被丢弃，而不是被调出；即，如果应用程序再次触摸该页面，则会从数据库中读取该页面并重新调整。这些只读页面的额外页面调出就是虚拟机开销。重新调配页面会带来更好的性能，因为调配页面的成本远低于页面调出的成本。

2.2 实验与结果

我们进行了一项实验，以确定德克萨斯州虚拟机开销对只读工作负载的影响。使用的硬件是运行 Digital Unix v.3.2 的 DECStation 3000/400，内存为 64 MB；数据库存储在 DEC RZ56C 磁盘上，交换文件存储在 DEC RZ26C 磁盘上。报告的数字是三个单独运行的平均值。

我们使用 OO7 基准 [2] 的 T1 遍历作为只读工作负载，并将数据库大小从 23 MB 更改为 86 MB。

只读应用程序永远不需要页出数据。因此，根据我们的定义，实验中测量的页面调出是德克萨斯州的虚拟机开销。表 1 显示了当访问的数据页数从 2759 增加到 7551 时的页出次数。对于前四种数据库大小，没有页出，因为数据适合内存。对于较大的数据库大小，比率

[2] 该页面从文件系统缓冲区高速缓存复制到应用程序的内存中。这个副本导致页面变脏；调酒也是如此。然而，即使德克萨斯州使用不同的方法来读取数据（例如映射文件），调配仍然会导致页面变脏。

[3] 注意操作系统占用3000到4000内存页。

页面调出数与应用程序访问的页面数之比稳步增加至 30% 左右。这意味着对于某些数据库大小，从数据库访问的所有页面中几乎有三分之一会在程序执行期间写入交换文件。给定程序的实际页面调出数由访问的页面数、访问模式的局部性、可用物理内存量以及页面替换策略的组合确定。在 T1 遍历中，访问模式存在大量局部性。因此，在局部性较低的工作负载（例如稀疏、顺序访问）中，我们预计页面调出与访问页面的比率甚至高于 30%。

表 1：OO7 T1 遍历。页面调出次数

数据库大小 (MB)	23	30	37	44	51	58	65	72	79	86
页数已访问	2759	3109	3722	4334	4878	5411	5957	6485	7021	7551
页面输出[3]		0	0	0	258	707	1134	1596	1902	2230

总而言之，只读应用程序永远不必页出数据。然而，我们的实验表明，swizzling 和德克萨斯州 VM 系统的交互可能会导致大量的页面调出。

3 提议的解决方案

3.1 清除页面脏状态

虚拟机开销的产生是因为操作系统不知道应用程序希望将特定页面视为干净的页面，即使它们已被修改。因此，我们提出了一个简单的系统调用，允许应用程序请求操作系统清除页面的脏状态。如果从内存中替换干净页面，它将被丢弃并且永远不会写入交换文件。如果应用程序引用这样的页面，则需要重新调整该页面。在德克萨斯州，这可以通过让系统调用同时重置页面的保护位来实现。如果再次引用该页面，则会发生错误，并且信号处理程序会重新调整该页面。在一般情况下，当引用已清理和丢弃的页面时，应用程序需要通知。该通知可以是一个信号，也可以是对应用程序的上行调用。

我们打算在 Digital Unix 中实现这个系统调用。
我们必须处理两种类型的页面：由 VM 系统管理的匿名内存页面，以及由统一缓冲区高速缓存 (UBC) 处理的映射文件页面。在这两种情况下，我们都需要重置脏数据

进程虚拟内存映射（vm map)中的位和物理映射（pmap)中的相应位。然而,在映射文件的情况下,页面也必须从UBC的脏列表移动到干净列表。我们预计执行系统调用本身的成本非常小（几个 usecs）。

VM 开销对应用程序的确切影响取决于通过与程序中的计算重叠可以屏蔽多少页出时间。由于这很难精确量化,因此我们使用单独的微基准来估计页面调出的平均时间。该程序分配了大量的页,并在每一页中修改了一个字节。它没有进行其他计算。我们还估算了在德克萨斯州调整一个页面的成本。结果如表 2 所示。应用程序使用我们的系统调用节省的时间取决于应用程序是否再次引用已清理和替换的页面。如果再次引用该页面,那么应用程序平均可以节省 10 毫秒;如果没有,它会节省整整 13 毫秒。

表 2:调配成本与页面调出成本

德克萨斯州每页平均 swizzling 成本	使用微基准估计页面调出时间
3毫秒	13毫秒

3.2 Alternatives

清除脏位的另一种方法是首先避免 swiz-zling。 OODB,例如 QuickStore [9] 和 ObjectStore [3],尝试通过每次将数据库段定位在应用程序的相同虚拟地址来避免混合。然而,仅仅避免调配并不能避免虚拟机开销,因为由于从文件系统缓冲区进行复制,页面仍然是脏的。将缓冲区管理与虚拟内存[4]集成并避免混合将消除这些系统的虚拟机开销。

避免 VM 开销的另一种方法是映射由 NFS 服务器支持的文件,该服务器向应用程序提供干净的混合页面。该方案的缺点是需要对 NFS 服务器进行大量修改。或者,像 Mach 这样的操作系统允许使用外部寻呼机,它可用于混合和分页并将其以干净的状态呈现给应用程序。这种方法的缺点是此类操作系统设施的不可移植性。

4 我们技术的其他应用

我们的脏位清除技术可以与不同的缓冲区管理策略结合使用。例如,QuickStore 在虚拟内存中管理固定大小的缓冲池,并假设该缓冲区始终由物理内存支持。 （这类似于关系数据库使用的技术）。只要不违反这个假设,就不会有VM开销,因为数据库系统负责缓冲池中的分页,并且它知道哪些页是只读的。然而,在存在内存竞争的情况下,这个假设可能会被违反,并且缓冲池中由于调配而脏的只读页面可能会被调出。清理缓冲池中的页面可以防止对只读数据进行分页。

通过将数据库映射到应用程序的虚拟内存中,OODB 缓冲区管理可以与虚拟内存 [1] [4] 集成。虚拟内存系统负责将数据分页进出映射文件。

调配再次导致只读页变脏。
为了正确性,不能允许这些混合页面被分页回数据库。因此,OODB 必须将它们视为读写页,这会导致更高的开销。我们的技术可以再次用来避免这种开销。

清除页面的脏状态的能力可以用于与VM开销问题具有相同一般性质的其他问题。一般来说,只要“重新计算页面”比调出页面更便宜,我们的技术就适用。一个示例是图形应用程序,其中页面上的数据被解压缩以用于显示目的。

5 结论

在本文中,我们确定了 swiz-zling 的间接成本,称为 VM 开销,当只读页面被 swizzled 并相对于操作系统变脏时,就会出现这种情况。

我们已经量化了德克萨斯州背景下的虚拟机开销。
我们提出了一个简单的系统调用来清除页面的脏状态,这可以用来减少VM开销。我们打算在 Digital Unix 中实现这个系统调用。

参考

[1] H. Boral、W. Alexander、L. Clay、G. Copeland、S. Danforth、M. Franklin、B. Hart、M. Smith 和 P. Vlduriez。原型设计 bubba,一个高度并行的数据库系统。 IEEE 知识与数据工程汇刊， 1(2):4–24,1990 年。

[2] MJ Carey、DJ Dewitt 和 JF Naughton。 oo7 基准测试。 1993 ACM 西格莫德。国际数据管理会议， 2(22):12–21,1993 年 5 月。

[3] C. Lamb、G. Landis、J. Orenstein 和 D. Weinred。这
对象存储数据库系统。 ACM 的通讯，
10(34),1991 年 10 月。
[4] D. McNamee、V. Narasaya、A. Tiwary、H. Levy、J. Chase，
和高勇。客户端缓冲区的虚拟内存替代方案
交易系统管理。提交出版， 1996 年。

[5] 杰布·莫斯。使用持久对象：调配或
不要搅拌。 IEEE 软件工程汇刊，
8(18):657–673,1992 年 8 月。
[6] V. Singhal、SV Kakkad 和 PR Wilson。 Texas：高效、可移植的持久存储。
第五届持久对象系统国际研讨会论文集,九月

1992年。
[7] SJ怀特.面向对象的指针调配技术
数据库系统。博士论文。威斯康星大学，
麦迪逊， 1994。
[8] SJ 怀特和 DJ 德威特。替代对象故障和指针混合策略的性能研究。第
18 届 VLDB 会议论文集,加拿大温哥华，

1992 年 8 月。
[9] SJ 怀特和 DJ 德威特。 Quickstore：高性能映射对象存储。 1994年
会议记录
ACM-SIGMOD 数据管理会议,明尼苏达州明尼阿波利斯， 1994 年 ，
5 月。
[10] 公关威尔逊。页面错误时指针混合：有效
支持标准硬件上的巨大地址空间。 ACM
SIGARCH 计算机体系结构新闻， 4(19),1991 年 6 月。