北京大学学报(自然科学版) 第 56 卷 第 6 期 2020 年 11 月
Acta Scientiarum Naturalium Universitatis Pekinensis, Vol. 56, No. 6 (Nov. 2020)
doi: 10.13209/j.0479-8023.2020.101

研究简报
Report

利用内存映射连续性提高TLB地址覆盖范围的技术评测

班义琨 张炜奇 周昱晨 易江芳†

北京大学信息科学技术学院系统结构研究所, 北京 100871; † 通信作者, E-mail: yijiangfang@mprc.pku.edu.cn

摘要 定义并评测典型基准测序程序内存映射中的连续性分布, 验证程序的内存映射中普遍存在多样的连续性(混合连续性)。对利用内存映射连续性提高 TLB 翻译覆盖范围的技术进行评测, 发现混合连续性的存在能够限制现有技术在真实场景中的实际效果。

关键词 虚拟存储; 混合连续性; 变换旁路缓冲器

Evaluation of Technologies Improving Translation Coverage of TLB Using Continuity of Memory Mapping

BAN Yikun, ZHANG Weiqi, ZHOU Yuchen, YI Jiangfang†

School of Electronics Engineering and Computer Science, Peking University, Beijing 100871;
† Corresponding author, E-mail: yijiangfang@mprc.pku.edu.cn

Abstract The authors define and evaluate the continuity distribution in memory mapping of some typical benchmark programs, and verifiy the existence of multiple types of continuity (mixed continuity) in memory mapping of programs. Furthermore, some technologies using continuity of memory mapping to improve translation coverage of TLB are evaluated. It is found that the existence of mixed continuity limits the actual effect of existing technologies in real scenes.

Key words virtual memory; mixed continuity; TLB (translation lookaside buffer)

虚拟存储技术在通用计算机系统中普遍使用, 并且逐渐应用到像图形处理器(graphics processing unit, GPU)等加速器中[1–3]。变换旁路缓冲器(translation lookaside buffer, TLB)对虚拟地址(VPN)到物理地址(PPN)的翻译性能十分关键。然而, 由于高内存占用的程序引发的存储容量需求逐渐增大, 地址翻译的时间开销也越来越大, 甚至可以达到实际执行时间的 50%[4]。为了减少页表遍历(page table walk)的开销, 许多方法都利用内存映射中存在的连续性来扩大 TLB 表项的覆盖范围[5–9]。

本文通过对广泛使用的基准测试程序进行连续性评测, 并定量地分析各种连续性的分布情况, 发现大多数程序中都存在不止一种连续性。通过分析和评测一些现存的扩大 TLB 地址覆盖范围的技术, 发现这些技术都采用合并表项或合并页面的 TLB 结构, 能够匹配的连续性是单一的甚至固定的, 未能充分利用存在的多种连续性, 所以在真实场景下不能获得理想的效果。

1 内存映射中的连续性

本文使用连续性来描述程序执行过程中内存映射中的连续块分布。

定义 1 连续块表示页表中多个页的集合, 这些页的虚拟地址和物理地址都是连续的, 不存在一

国家科技重大专项(2018ZX01029101)资助
收稿日期: 2019–11–28; 修回日期: 2020–02–04

个连续块包含另一个连续块的情况。连续块的大小指块中页的数量。

1.1 连续性的多样性

应用程序是复杂多样的，不同的应用程序在运行时，内存映射中的连续性也是多样的。根据连续块的大小，内存映射中存在的连续性分为3类。

第1类连续性指程序中存在大小为512个及以上页面的连续块，称为大连续性。透明超页(transparent huge page, THP)[5]是这类连续性的典型体现。如x86-64架构支持使用2 MB和1 GB的超页分别代替512和512×512个连续映射的4 KB基本页。

与大连续性相反，第2类连续性指程序中存在大小为32个及以下页面的连续块，称为小连续性。这类连续性在碎片化的内存映射中很普遍。当系统运行一段时间后，正在使用中的页(尤其是大连续块的分布)使得很难在内存中找到大型连续区域，限制了新的大连续块的分配。另外，常见的NUMA架构需要利用细粒度的内存映射，将经常访问的页面放在靠近内存的位置，如3D堆叠式DRAMs[10]、基于网络的混合存储立方体(HMC)[11]和非易失性存储器(NVM)[12]，这导致更严重的内存碎片[13]。

第3类连续性介于大连续性与小连续性之间，称为中连续性。在这类连续性下分布的连续块比细粒度的内存映射大，但比超页小。一些研究已在许多真实应用程序中发现中连续性[7–9]。

事实上，内存分配是杂乱且碎片化的，所以真实系统中几乎不会只存在单一类型的连续性。程序中包含不止一种连续性的情况称为混合连续性。

1.2 连续性评测

我们使用一台4核Intel Core i7-7700HQ，频率为2.8 GHz的x86-64机器进行评测，内存为4 GB，操作系统为Linux 4.16。预热后，周期性地记录内存映射中的连续块分布，用以评测应用程序运行时内存映射中的连续性。使用SPEC CPU 2006[14]和Graph500中的基准测试程序，其中Graph500工作集的大小设置为8 GB。

使用Linux提供的pagemap[15]接口获得虚拟地址和物理地址的映射。在程序执行过程中，每分钟扫描一次进程的页表，并记录连续性信息。

在执行过程中采用上述连续块分类。15个评测程序连续块大小分布的平均情况如图1所示。为了排除THP技术[5]的影响，统计THP开和关时的连续块大小分布。可以发现，无论THP如何设置，除hmmer程序只有小连续性外，几乎所有程序的内存映射中都会同时出现多种连续性。例如，应用程序mcf在执行过程中的连续块有小、中、大3种连续性，并且每种连续性的连续块数量都占据一定的比例。与不使能THP相比，使能THP时有更多的应用程序(如omnetpp)呈现大连续性，这是因为THP技术使得在程序执行过程中，一些小连续块或中连续块合并成为大连续块。

2 TLB失效评测

到目前为止，已经有许多技术考虑到利用内存映射的连续性来扩大TLB地址的覆盖范围。

2.1 评测技术

THP[5]是Linux操作系统中超页(2 MB)的一个实现，使用超大页面代替原本连续的一系列基本页面。但是，由于超页的大小是固定的，而待分配的连续块大小是多样的，所以势必造成超页空间的浪费，导致在扩大TLB覆盖范围时有很大的局限性，并只能匹配固定大小的连续块。

RMM[6]中引入基于硬件的段来完全覆盖连续块，排除了页的尺寸限制，但需要添加额外的段TLB。段TLB的硬件结构是全相联的，每个段都覆盖一个非常大的连续块，因此操作系统需要做出重大改变来保持这种分配。与THP类似，RMM可以覆盖大连续块，但会忽视小连续块和中连续块。因此，在具有混合连续性的真实场景中，段TLB技术的效果必然会受到影响。

CoLT[7]和Cluster[9]是两种基于硬件的合并技术，用于应对小连续性或者单一连续性。改进后的TLB表项最多覆盖大小为8的连续块，随着程序执行中连续块大小的增加，需要改进的表项数目不断增加。例如，一个大连续块(512)需要大量(至少64个)合并表项才能完全覆盖。然而，这些技术的TLB结构扩展性较差，无法满足连续性的不断增长。

Anchor[8]引入锚表项的概念。锚表项在普通页表项之间均匀分布，以便记录连续性。通过调整页表中最优的锚距离，可以适配连续块的大小。具体地，在页表中每N(锚距)个表项中都放置一个锚表项，记录连续页面的数量。例如，如果内存页被分配大小为16的连续块，那么最优的锚距离就是16。然而，对于比锚距大的连续块，就需要多个锚表项才能覆盖。对于比锚距离小的连续块，如果在块与相应的锚表项之间存在不连续的页，就会被忽

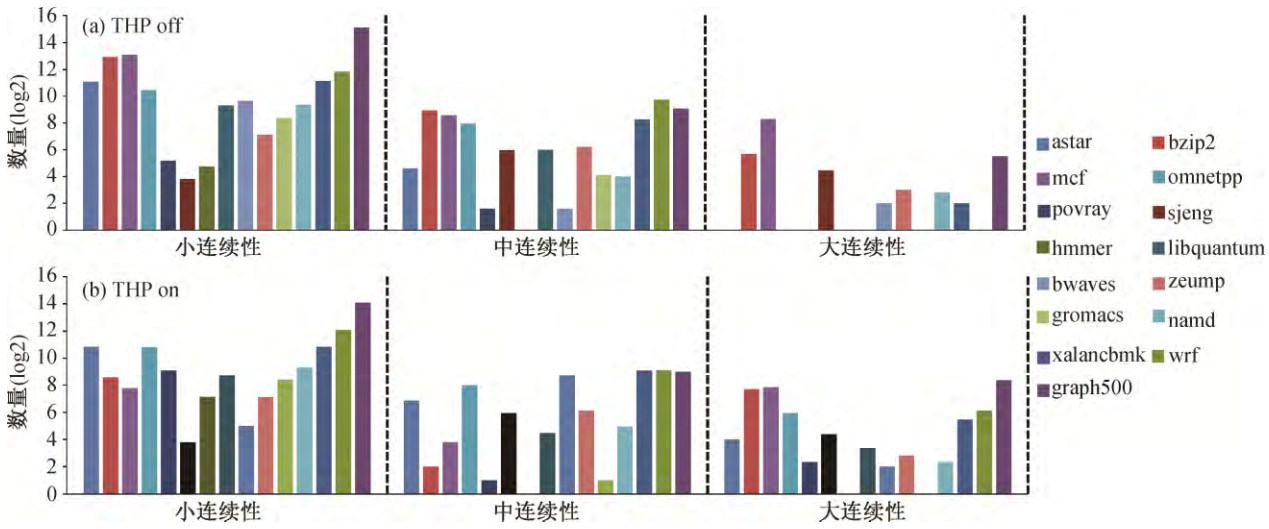

图 1　基准测试程序前 10 亿条指令范围内中连续块的分布

Fig. 1　Distribution of contiguity chunks at first billionth instruction boundary for the used benchmarks

略。所以在一个时期，Anchor 只能匹配一种类型的连续性，并且会使操作系统增加较大的开销。

2.2 TLB 参数设置

表 1 展示以上技术中 TLB 的配置，尽量保证各种技术间 TLB 配置的一致性以及每种方法在效果上的最优选择。所有方法 L1 TLB 的配置都相同，L2 TLB 容量设置为 1024 个表项。除常规的 L2 TLB 外，Cluster 需要额外的合并 TLB，RMM 需要增加一个 32 个表项的全相联段 TLB。

表 1　评估中使用的 TLB 的配置[8]

Table 1　TLB configuration used for evaluation[8]

技术	TLB 配置
L1 TLB	基本页 TLB: 64 表项, 4 路 超页 TLB: 32 表项, 4 路
基准和 THP L2 TLB 配置 CoLT	1024 表项, 8 路 基准 L2 TLB
Cluster	常规 TLB: 768 表项, 6 路 合并 TLB: 320 表项, 5 路
RMM	基准 L2 TLB 段 TLB: 32 表项, 全相联
Anchor	基准 L2 TLB

2.3 评测结果

对于现有技术，仍然使用评测连续性时的应用程序进行评测，用 TLB 失效率来评估每个技术的效果。TLB 配置与文献[8]中的现有技术相同(表 1)。THP 是 Linux 系统的自带实现，Cluster 和 RMM 需要额外的硬件结构来支持。以基本配置的 TLB 为基准，对所有应用程序在各种技术下的 TLB 失效进行归一化，结果见图 2。

对不同的基准测试程序，不同方法减少的 TB 失效也不同，原因是基准测试程序相应的连续性分布与不同方法对各类连续性的利用侧重点不同。操作系统分配的连续块越多，可以扩大的 TLB 地址覆

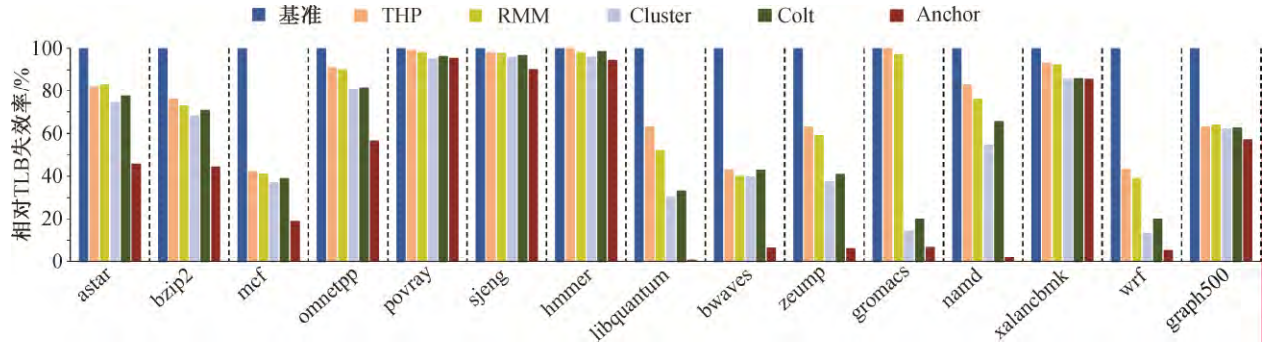

图 2　真实系统下几种方法的相对失效率

Fig. 2　Relative misses of all approaches for real system scenario

盖范围就越大，不同方法的性能差异也越大。

有些应用程序在各种技术下都会减少一定的TLB失效，呈现TLB失效的阶梯式下降。例如，应用程序zeusmp在执行过程中同时呈现3种连续性，所以THP和RMM可以利用大连续性来减少TLB失效，同时CoLT和Cluster可以利用小连续性进一步减少更多的TLB失效。有些程序在执行过程中连续性较单一，所以只有一类利用同种连续性的技术效果明显。例如，应用程序gromacs在执行过程中未呈现大连续性，并且中连续性对应的连续块数量也较少，所以THP和RMM几乎没有减少TLB失效。相反地，主要利用小连续性的CoLT和Cluster就取得很好的效果。应用程序graph500，THP和RMM，可以减少很大部分的TLB失效，但再使用CoLT和Cluster时，几乎没有进一步的效果。

作为目前最先进的技术，Anchor在一些应用程序上的效果很好。例如，应用程序libquantum，因其与zeusmp类似地呈现出多种连续性，所以关注于大连续性和小连续性的技术都取得一定的效果。Anchor技术在这个应用程序中的表现格外好，几乎消除所有的TLB失效。但是，Anchor在某些应用程序中的表现仍不如人意。例如，应用程序hmmer在所有技术下都很难减少TLB失效，包括Anchor。

3 结论

本文首先定义内存映射中存在的多种连续性；然后评测并定量地分析一些典型基准测序程序内存映射中的连续性分布，验证了程序的内存映射中普遍存在多样连续性(混合连续性)；最后对THP，RMM，CoLT，Cluster和Anchor等现有技术进行TLB失效评测，发现混合连续性的存在限制了现有技术在真实场景中的实际效果。综上所述，需要提出新的结构和技术，可以充分利用内存映射中的混合连续性，进一步改善虚拟存储系统的性能。

参考文献

[1] AMD Corporation. Compute cores [EB/OL]. (2014–01) [2019–10–01]. https://www.amd.com/Documents/Compute_Cores_Whitepaper.pdf

[2] Swapnil H, Mark D H, Michael M S. Devirtualizing memory in heterogeneous systems // ACM International Conference on Architectural Support for Programming Languages and Operating Systems. Williamsburg, 2018, 53: 637–650

[3] Olson L E, Power J, Hill M D, et al. Border control: sandboxing accelerators // IEEE/ACM International Symposium on Microarchitecture. Waikiki, 2015: 470–481

[4] Basu A, Gandhi J, Chang J, et al. Efficient virtual memory for big memory servers. Computer Architecture News, 2013, 41(3): 237–248

[5] Linux Kernel Documentation. Transparent hugepage support [EB/OL]. (2017) [2019–10–01]. https://www.kernel.org/doc/Documentation/vm/transhuge.txt

[6] Karakostas V, Gandhi J, Ayar F, et al. Redundant memory mappings for fast access to large memories // ACM SIGARCH Computer Architecture News. Portland, OR, 2015, 43: 66–78

[7] Pham B, Vaidyanathan V, Jaleel A, et al. CoLT: coalesced large-reach TLBs // IEEE/ACM International Symposium on Microarchitecture. Vancouver, 2012: 258–269

[8] Park C H, Heo T, Jeong J, et al. Hybrid TLB coalescing: improving TLB translation coverage under diverse fragmented memory allocations. Computer Architecture News, 2017, 45(2): 444–456

[9] Pham B, Bhattacharjee A, Eckert Y, et al. Increasing TLB reach by exploiting clustering in page translations // International Symposium on High Performance Computer Architecture. Orlando, 2014: 558–567

[10] Loh G H. 3D-stacked memory architectures for multi-core processors // ACM SIGARCH computer architecture news. Beijing, 2008, 36: 453–464

[11] Pawlowski J T. Hybrid memory cube (HMC) // Hot Chips 23 Symposium. Stanford, 2011: 1–24

[12] Dulloor S R, Roy A, Zhao Z, et al. Data tiering in heterogeneous memory systems // European Conference on Computer Systems. London, 2016: 1–16

[13] Park C H, Heo T, Huh J. Efficient synonym filtering and scalable delayed translation for hybrid virtual caching. ACM SIGARCH Computer Architecture News, 2016, 44(3): 217–229

[14] Henningjohn L. SPEC CPU2006 benchmark descriptions. ACM SIGARCH Computer Architecture News, 2006, 34(4): 1–17

[15] Linux Kernel Documentation. Pagemap, from the userspace perspective [EB/OL]. (2016) [2019–10–01]. https://www.kernel.org/doc/Documentation/vm/pagemap.txt