高技术通讯 2021 年 第31 卷 第5 期:464-478

doi:10.3772/j.issn.1002-0470.2021.05.002

外部高速缓存与非易失内存结合的混合内存体系结构特性评测①

潘海洋②[*,**] 刘宇航[*,**,***] 卢天越[*] 陈明宇[*,**,***]

([*]中国科学院计算技术研究所先进计算机系统研究中心 北京 100190)

([**]中国科学院大学 北京 100049)

([***]鹏城实验室 深圳 518055)

摘 要 以非易失性存储器(NVM)作为主存且以动态随机存取存储器(DRAM)作为片外高速缓存(EC),是一种可以满足大数据应用内存容量需求的新型混合内存结构(EC-NVM)。该结构同时具有NVM的大存储容量和DRAM的低存取延迟的优点。传统的结构是以片内SRAM作为片内高速缓存(IC)且以DRAM作为主存(IC-DRAM)。与IC-DRAM相比,EC-NVM在容量比、延迟比方面均有显著不同,导致在IC-DRAM场景下的设计方法和优化策略直接迁移到EC-NVM上未必有良好的效果。本文评测了EC-NVM的体系结构特性(包括高速缓存粒度、关联度、替换算法、预取算法等),获得了指导EC-NVM结构的设计和优化的一系列发现。

关键词 非易失内存(NVM);动态随机存取存储器(DRAM)高速缓存;大数据应用

0 引 言

随着大数据应用的数据访问愈加密集,内存计算变得更加重要,它通过将热点数据存放在内存中从而提升数据访问性能。例如,内存数据库(如SAP HANA[1])和内存计算框架(如Apache Spark[2])几乎将全部数据都直接存放在内存中,这样虽然避免了访问磁盘的开销,提升了系统整体的性能,但却需要巨大容量的内存,甚至是和待处理的数据规模相当的内存容量。然而,基于传统动态随机存取存储器(dynamic random access memory, DRAM)技术的内存很难满足如此巨大的容量需求。

非易失性存储器(non-volatile memory, NVM)的出现为满足大数据应用的内存容量需求提供了机会。与DRAM相比,NVM具有更大的存储密度,因此,非易失性存储器可以提供更大的主存容量。例如,相变内存(phase-change memory, PCM)作为一种广泛应用的非易失性存储器,其存储密度是传统DRAM的4倍[3],进而相变内存可以满足大数据应用的内存容量需求。但是,相较于DRAM,非易失性存储器具有更高的访问延迟,直接使用非易失性存储器作为系统主存会降低系统的整体性能。

为了解决非易失性存储器访问延迟高的问题,研究人员提出了一种新兴的混合内存结构,在使用非易失性存储器作为主存来提供巨大的内存容量的同时,使用DRAM作为高速缓存来加快访问速度。需要注意的是,这里的高速缓存区别于传统结构的处理器内部高速缓存,DRAM高速缓存处于处理器外部,被称为片外高速缓存。在本文后续的描述中,将这种以片外DRAM为高速缓存同时以NVM为主存的结构称为EC-NVM(external cache + NVM)内存,将传统的由中央处理器(central processing unit, CPU)内部高速缓存和DRAM主存组成的结构称为

① 国家自然科学基金面上项目(61772497)和国家重点研发计划(2017YFB1001600)资助项目。

② 男,1990年生,博士生;研究方向:计算机系统结构;联系人,E-mail: panhaiyang@ict.ac.cn

(收稿日期:2020-04-10)

IC-DRAM(Internal Cache + DRAM)内存。

作为一种新兴的混合存储结构,EC-NVM 受到了学术界和工业界的广泛关注[4]。在学术界,研究人员利用 EC-NVM 结构构建了比传统 DRAM 内存容量大 4 倍的系统,进而将系统整体性能提升了 3 倍[3]。在工业界,英特尔公司(Intel)最新发布的傲腾内存(Optane Memory)是典型的 EC-NVM 内存,它使用相变内存作为主存,DRAM 作为高速缓存,同时拥有了非易失性存储器容量大、DRAM 访问延迟低的优点[5],一经推出,就受到了业界的关注。

与传统的片内高速缓存(internal cache, IC)-DRAM (IC-DRAM)结构相比,EC-NVM 结构在容量比(capacity ratio, CR)和延迟比(latency ratio, LR)方面都有显著不同。(1)容量方面,外部高速缓存比内部高速缓存的容量大得多;(2)延迟方面,外部高速缓存比内部高速缓存的访问延迟高,NVM 主存比 DRAM 主存访问延迟高。

考虑到 EC-NVM 在延迟和容量上的巨大变化,传统 IC-DRAM 结构上的设计对 EC-NVM 结构还是否有效? 本文从 3 个方面阐述 EC-NVM 的体系结构特性变化给设计带来的问题。

第一,外部高速缓存的访问延迟比内部高速缓存更高,这会影响到缓存粒度的选择。缓存粒度是高速缓存和主存进行数据交换时的数据块大小。使用大的高速缓存粒度可以有效减少高速缓存的冷缺失,提高整体的命中率,减少访问主存的次数,从而提升系统整体性能。但是使用大的高速缓存粒度又会使访问缓存本身的开销变大,传统 IC-DRAM 采用 64 字节缓存粒度来平衡缓存粒度和缓存延迟,以达到最优的性能。然而,在 EC-NVM 场景下,外部高速缓存的访问延迟和容量都有明显增加,因而是否能直接沿用 IC-DRAM 场景下的结论,需要重新探索。

第二,大容量的外部缓存需要维护并保存大容量的高速缓存标记(cache tag,用于识别内存数据块是否在高速缓存中)。这些标记的存储开销远超过现有处理器片的片内空间,以至于这些高速缓存标记只能被保存在外部 DRAM 中,这和 IC-DRAM 将高速缓存标记保存在片内是完全不同的。在一个使用了高速缓存的内存系统中,一次正常的缓存操作需要先查询高速缓存标记,确定是否是高速缓存命中,然后再访问高速缓存(如果缓存命中)或者主存(如果缓存缺失)。将高速缓存标记保存在外部 DRAM 中直接导致在缓存命中的情况下增加了一次额外的外部 DRAM 访问,从而导致命中延迟的增加。尤其当高速缓存按照组相联的方式组织的时候,需要先把整个高速缓存组的所有缓存标记都查询一遍才能确认是否是访存缺失或命中,这为缓存命中的过程引入了更多次外部 DRAM 访问,进而导致更大的性能开销。为了解决这个问题,Alloy Cache[6]把直接相联的缓存标记和数据保存在一起,读取高速缓存标记的同时将相邻的数据也读取进入处理器,这样可以节省一次外部 DRAM 的访问。LH-Cache[7]是将同一高速缓存组的数据放在同一个 DRAM 行上,同时将处于同一缓存组的缓存标记也都放在一起,这样可以将一个缓存组的标记访问次数减小到一次。比较这两种方法,采用直接相联的高速缓存一次命中需要一次外部 DRAM 访问,但缓存命中率更低;采用了组相联的高速缓存命中率更高,但是其一次缓存命中需要两次外部 DRAM 访问。因此,在 EC-NVM 中,关联度的选择还需要重新考虑。

第三,主存的延迟会严重影响预取算法的设计。预取技术是指通过对历史访问请求的分析,预判出后续访问可能会请求某个数据块,提前将该数据块取到高速缓存中,而不用等到请求来临时才去取数据,这有效地隐藏了访问延迟,是一种广泛使用的访存优化技术。但是在实际系统中,完成预取操作是需要一定延迟的,如果对某个数据块的预取还未完成,而对该数据块的访问请求已经来临,这种情况下,即便预取地址是准确的,但这种预取操作并没有达到提前取回数据的目的,也就无法加速内存访问,这种想象被称为“预取迟到”。预取迟到严重影响预取的效果,并且主存的访问延迟越高,完成一次预取请求的延迟越高,预取迟到程度也就越严重。在 EC-NVM 场景中,主存变成了 NVM,比 IC-DRAM 的主存的访问延迟变得更高,这意味着一个预取访问需要等待更长的时间,因此预取迟到率会明显增加。

为了降低预取迟到率,现有算法大多采用增大预取提前量的方式,例如,反馈式预取(feedback directed prefetching, FDP)[8]预取器通过改变预取距离降低预取迟到率。FDP 监控预取正确率和迟到率,一旦发现预取迟到率增加,则增大预取距离,对更"远"的数据进行预取;另外,一旦发现预取正确率降低,则降低预取距离,保证足够高的正确率。文献[9]的预取器先按照多种预取距离进行预取,通过使用不同预取距离的预取结果,选择最优的预取距离。然而,现有的预取算法大多只考虑了 IC-DRAM 的场景,对 EC-NVM 场景中更长的预取完成时间是否适用还需要重新评测。

考虑到上述 3 个问题,本文从关联度、缓存粒度、替换算法、预取算法等 4 个方面,探索了 EC-NVM 结构变化对体系结构设计带来的影响。本文主要回答了以下几个问题。

(1)EC-NVM 的高速缓存粒度应该如何选择?IC-DRAM 通常使用 64 字节作为高速缓存粒度,这一粒度在 EC-NVM 场景下是否仍然是最好的选择?

(2)考虑到额外访问缓存标记的开销,组相联算法是否一定比直接相联效果好?能否继续优化组相联访问缓存标记的开销?

(3)现有的替换算法是否依然能在 EC-NVM 中发挥明显的作用?现有替换算法在 EC-NVM 场景下是否有较大的提升空间?

(4)EC-NVM 更大访问延迟的主存是否带来严重的预取迟到问题?现有的预取算法是否能明显提高性能以及是否有较大的改进空间?

通过回答以上 4 个问题,本文发现了以下 EC-NVM 的体系结构特性,对 EC-NVM 的结构设计和优化具有指导意义。

(1)关于 EC-NVM 的缓存粒度。经过上层多级缓存的过滤,大数据应用在第 4 级缓存上具有较好的空间局部性,因而增大缓存粒度可以显著提高缓存命中率,并且越小的缓存容量下,大粒度提高命中率的效果越明显。由于 EC-NVM 缓存在处理器外部,增大缓存粒度导致增加的命中延迟不可忽略,因此缓存粒度的选择要综合考虑大粒度对命中率的提高和命中延迟的增加。实验表明,大部分应用的最佳缓存粒度为 128 字节或 256 字节。

(2)关于 EC-NVM 的缓存关联度。组相联可以提高缓存命中率,但是 EC-NVM 的大容量缓存降低了冲突缺失,使得组相联提高命中率的效果大大降低,无法抵消组相联导致的额外访问缓存标记的开销,因此,组相联缓存性能反而低于直接相联的缓存。由于 EC-NVM 的缓存容量较大,使用有限容量的缓存标记的缓存(Tag Cache)无法发挥明显作用。Tag Cache 的设计需要妥善解决高效利用有限容量的 Tag Cache 问题。

(3)关于 EC-NVM 的缓存替换算法设计。EC-NVM 的大容量缓存降低了缓存压力,使重用距离大大减小,因此现有替换算法相比随机替换无法明显提升缓存命中率。

(4)关于 EC-NVM 的预取算法设计。由于大数据应用在 EC-NVM 场景下良好的空间局部性,现有多种预取算法都可以对 EC-NVM 结构有明显的加速效果。由于非易失性主存存在较大的访问延迟,EC-NVM 结构存在较明显的预取迟到问题。但是,以地址连续的预取为主的预取算法可以有效改善预取迟到问题。

1 EC-NVM

图 1 描述了 EC-NVM 内存的结构。在该结构中,非易失性内存和 DRAM 高速缓存都通过传统的内存总线(比如 DDR4)和处理器相联,并且都通过内存控制器进行管理。

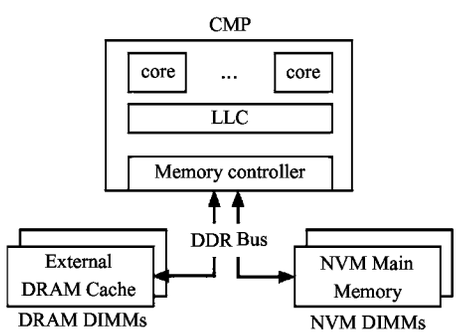

图 1 EC-NVM 结构图

表 1 描述了 EC-NVM 内存结构和传统 IC-DRAM 内存结构在容量和延迟上的不同。IC-DRAM

包括两种情况,处理器内部SRAM作最后一级高速缓存和3D堆叠式DRAM作最后一级高速缓存。容量方面,主存和缓存的容量比例有了明显变化。对于处理器内部SRAM作最后一级高速缓存的情况,主存容量是高速缓存容量的4096倍;对于主存是3D堆叠式DRAM的情况,主存容量是高速缓存容量的512倍;而EC-NVM情况下,主存容量是高速缓存容量的8倍,相较于传统结构的主存与高速缓存的容量比减小了多个数量级。延迟方面,主存和缓存的延迟比最多从1变成了10,这意味着高速缓存缺失的开销增加了9倍。另外,外部高速缓存相当于是比内部最后一级缓存(last level cache, LLC)还要低一层的缓存,也就是说,到达外部高速缓存的访问要经过更多层的缓存的过滤,其空间和时间局部性都会发生一定变化。因此内存访问模式会受到EC-NVM体系结构特性的影响。

表1 由不同缓存和主存构成的内存结构参数对比

类型	大小	延迟	容量比	延迟比
传统处理器内部最后一级缓存(IC-DRAM1)				
SRAM	16 MB	20 ns	4096	3
外部DRAM	64 GB	60 ns		
3D堆叠式DRAM缓存(IC-DRAM2)				
3D堆叠DRAM	128 MB	60 ns	512	1
外部DRAM	64 GB	60 ns		
外部DRAM缓存(EC-NVM)				
外部DRAM	32 GB	60 ns	8	10
非易失性内存	256 GB	600 ns		

2 研究方法概述

2.1 大内存基准测试程序构建

正如在引言中所述,大数据应用大都对内存容量有巨大的需求,而EC-NVM正是为了满足大数据应用的内存容量需求而产生的,因此在对EC-NVM结构的评测时,也需要使用具有巨大内存容量的需求的应用。本文用“工作集(WorkingSet)”表示程序在运行过程中需要访问的内存容量的总和[10],将具有较大工作集的应用统称为大内存应用。

表2列出了本文选择的大内存应用。选择标准是:(1)测试程序需要有足够大的工作集来模拟大数据应用对内存容量的需求。本文采用的大数据应用在实际运行中,其工作集会根据输入数据的大小而变化,因此具有巨大的内存容量需求。值得注意的是,本文采用了部分来自SPECCPU2005[11]的应用,为了使测试可以达到更大的工作集,将多个SPEC程序混合在一起运行(SPEC-MIX),这是SPECCPU经常使用的做法,通过变换应用的组合和数量,可以得到不同大小的工作集。(2)除了具有巨大的内存容量需求,测试程序还应该能够代表各个领域的大数据应用。本文选取了来自社交网络、电子商务、搜索引擎、高性能计算等多个领域的典型大数据应用。并且,本文选择的应用都是来自工业界真实的应用程序。

表2 大内存基准测试程序

应用	工作集/GB	描述
K-Means	95.2	聚类应用
Bayes	206.8	推荐系统
SPECJbb	87	JVM标准测试程序
Soapdenovo2	160.5	基因序列分析
SAP HANA	111.9	内存数据库
TeraSort	253.2	排序算法
HPCG	145.8	高性能计算
HPL	173.7	高性能计算
SPEC-MIX	52.3	SPECCPU 2006
GraphX	88.1	图计算

关于测试数据集的选取,本文用TPC-DS[12]来生成SAP-HANA[1]的数据和查询。对于K-Means、Bayes、Terasort等比较成熟的测试程序,本文直接采用了来自BigDataBench[13]的测试数据集。Soapdenovo2的数据文件来自公开网站[14]。

2.2 实验方法介绍

本文采用内存访问踪迹(trace)驱动的内存模拟器NVMain[15]来模拟EC-NVM结构,输入为预先经过聚类工具[16]选择的有代表性的内存访问trace片段,输出为单位时间内可以完成的内存访问请求数(accesses per cycle, APC)[17]。

表3展示了评测配置。评测主要分成两部分,包括获取内存访问trace以及基于trace的分析和评测。

表 3 实验配置

内存 trace 获取平台配置	
处理器	Intel Xeon E5@2.1 GHz 24-core
操作系统	CentOS(默认), SLES12.1(SAP HANA)
内存大小	256 GB
外部 DRAM 高速缓存配置	
容量	32 GB
结构	1 Channel, 8 Banks, 2 kB 行大小
总线	64 位总线宽度, 1333 MHz
时序	tRCD-tTP-tBURST-tRTP-tCCD = 9-9-4-3-2
非易失主存配置	
容量	大小: 256 GB
结构	1 Channel, 8 Banks, 2 kB 行大小
总线	64 位总线宽度, 800 MHz
时序	tRCD-tTP-tBURST-tRTP-tCCD = 90-270-4-3-2

在获取 trace 阶段,本文将大内存应用部署在服务器上正常运行,同时使用硬件内存访问追踪工具(HMTT[18])记录真实的内存访问地址。服务器是 Intel Xeon E5 平台,运行了 CentOS7 操作系统,内存容量是 256 GB。

在评测阶段,本文使用了 Alloy Cache[6] 来模拟直接相联的组织方式,使用了 LH-Cache[7] 来模拟组相联的组织方式。

3 不同缓存粒度对 EC-NVM 结构的影响

3.1 缓存粒度对缓存命中率的影响

缓存粒度是缓存结构设计的一个重要参数,增大缓存粒度可以有效地减少缓存冷缺失,尤其对于空间局部性好的应用,大粒度可以有效提高缓存命中率。

图 2 展示了缓存粒度对于 EC-NVM 结构缓存命中率的影响。每一个子图表示一个应用的结果。每条线表示固定主存缓存容量比下不同缓存粒度大小的命中率变化。如“CR8”表示缓存容量为主存容量的 1/8。

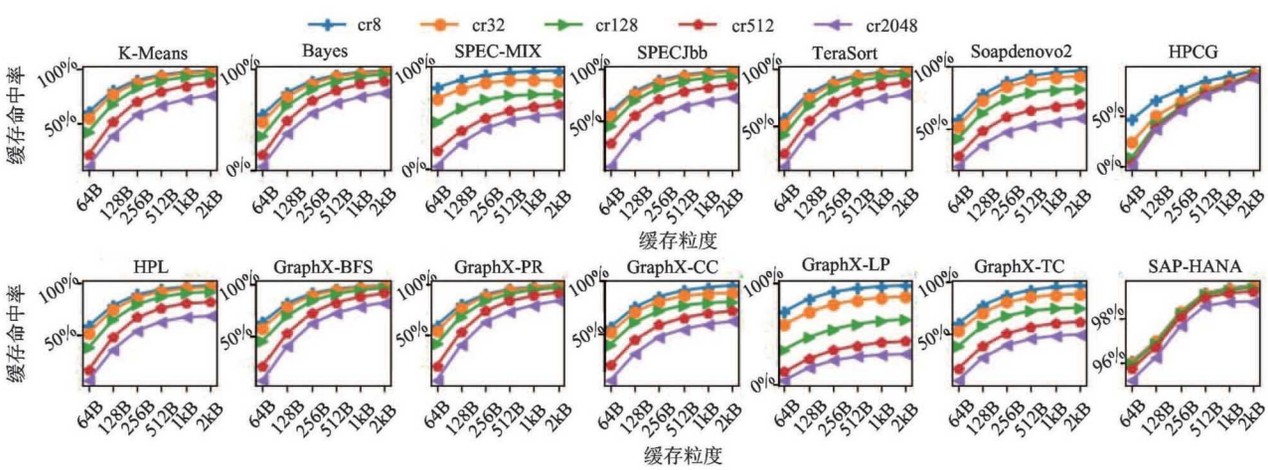

图 2 不同缓存粒度对缓存命中率的影响

从图 2 可以看到,增大缓存粒度可以明显提高缓存命中率,且对不同容量的缓存都有相似的效果。(1)缓存粒度最初从 64 字节开始变大的时候,缓存命中率明显增加,这说明随着缓存粒度的增大,强制缺失得以明显减少。例如,对于 SPEC-MIX,当缓存容量为主存容量的 1/2048 时,将缓存粒度从 64 字节增大到 256 字节,缓存命中率从 3% 提升到 39%。(2)大容量缓存情况下,增大粒度对提高缓存命中率的作用会比小容量缓存情况下低很多。例如,对于 SPEC-MIX,当缓存容量为主存容量的 1/8 时,将缓存粒度从 64 字节增大到 256 字节,缓存命中率从 79% 提升到 91%。(3)当缓存粒度增加到一定程度,命中率的增加开始变得平缓甚至个别应用出现了下降,这是由于大部分的强制缺失已经被消除,继续增大缓存粒度的效果比较有限,同时大粒度会导致更多冲突缺失。例如,对于 SPEC-MIX,缓存容量

为主存容量的1/2048时,将缓存粒度从256字节增大到512字节,命中率从39%提升到53%。这种现象在大容量高速缓存中更加明显,例如高速缓存容量为主存容量的1/8时,高速缓存命中率只能提升4%。

粒度对命中率的影响是由大数据应用的空间局部性决定的。本文中,用页内地址空间间隔来表示程序的空间局部性,所谓页内地址空间间隔是指同一页面的相邻两个访问的地址差值。图3表示了空间局部性。"32 < =stride <64"表示地址间隔位于32和64之间。可以看到,EC-NVM下的大数据应用具有较好的空间局部性。平均超过40%的访问的地址间隔为1,这意味着超过40%的访问是连续的,超过65%的间隔是小于8个块(块大小为64字节)的,理论上,用512字节的缓存粒度可以将用64字粒度的高速缓存的命中率从20%提升到32.5%。

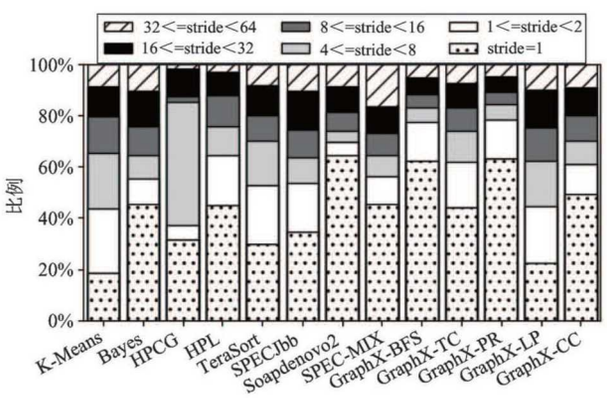

图3 空间局部性分析

这说明即便EC-NVM的缓存是第4级缓存,经过上层多级缓存的过滤之后,落在第4级缓存上的访问仍然具有较好的空间局部性。

3.2 缓存粒度对综合性能的影响

增大缓存粒度虽然可以提高命中率,但是也会增大缓存命中延迟。由于EC-NVM的缓存在处理器外部,相比于处理器内部的高速缓存,其访问延迟不可忽略,而且该延迟是和缓存粒度的大小成正比的。例如,在使用64位的DDR3内存总线的情况下,内存控制器从DRAM的感应放大器读取64字节的数据需要一次总线突发(burst)操作即可完成,而获取256字节的数据则需要4次总线突发,即256字节缓存粒度的命中延迟是64字节的4倍。由此可见,虽然增大粒度可以提高缓存命中率,减少访问主存的次数从而提升性能,但是大粒度也导致了更高的命中延迟,又会使系统性能有所降低。因此,缓存粒度的选择需要平衡缓存命中率和命中延迟。

图4展示了缓存粒度对内存系统性能的影响。和图3类似,每个子图表示一个应用,每条线表示固定缓存容量情况下的APC。为了便于对比,本文以主存缓存容量比是2048且粒度是64字节的APC为基准,得到其他容量比和粒度下相对基准的APC。

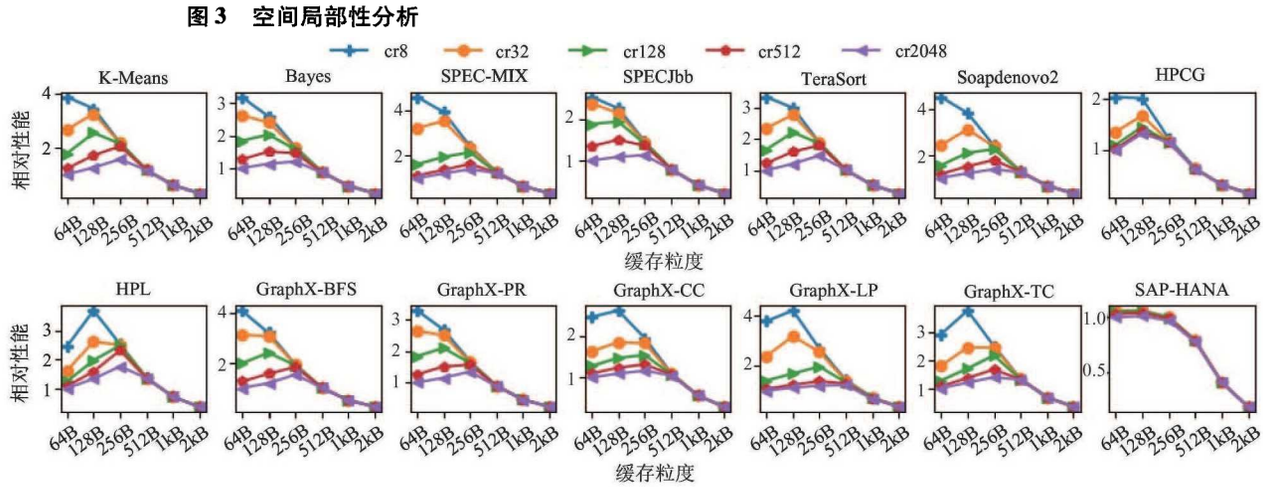

图4 不同缓存粒度对系统综合性能的影响

通过图4可以得知,对于缓存容量偏小的EC-NVM,增大缓存粒度可以迅速提高缓存命中率,因此,增加缓存粒度可以明显提高综合性能。而随着缓存粒度的继续增大,命中率的提升趋势变得平缓,而命中延迟还在继续增大,综合性能又会变低。因此对于小容量的高速缓存,几乎全部应用(除SAP-

HANA 外)的最佳粒度都位于中间位置(128 字节、256 字节或 512 字节)。例如,在缓存容量为主存容量的 1/2048 的情况下,使用 256 字节的缓存粒度可以将 K-Means 的综合性能相比使用 64 字节时提升 56%。

对于容量偏大的 EC-NVM,一方面增加粒度对性能带来的提升趋势变缓,而另一方面大粒度引入的额外的命中延迟继续增加,因此大容量缓存情况下,最佳粒度会变小。例如,将容量比从 2048 变成 8 时,TeraSort 的最佳粒度逐渐从 256 字节变成 64 字节。然而,即便容量比增加到了 8,仍然有一半的应用的最佳粒度大于 64 字节。

结合上述结果和分析,得出以下结论。

结论 1 经过上层多级缓存的过滤,大数据应用在第 4 级缓存上具有较好的空间局部性,因而增大缓存粒度可以显著提高缓存命中率,并且越小的缓存容量下,增大粒度对于提高命中率的效果越明显。

结论 2 由于 EC-NVM 缓存在处理器外部,增大缓存粒度会明显增加命中延迟,因此缓存粒度的选择要综合考虑大粒度对命中率的提高和命中延迟的增加,传统 64 字节大小对 EC-NVM 场景不再适用。

4 不同缓存关联度对 EC-NVM 结构的影响

本节讨论不同高速缓存关联度对 EC-NVM 结构的影响。为了排除替换算法的影响,本节的测试都采用最优替换算法。所谓最优替换算法是文献[19]于 1966 年提出的一种已经在理论上被证明的最优替换算法,其所选择的被淘汰缓存块,将在最长(未来)时间内不再被访问,从而达到最优替换的效果。

4.1 缓存关联度对缓存命中率的影响

图 5 展示了关联度对命中率的影响。和图 2 类似,每一个子图表示一个应用的结果。每条线表示在固定主存缓存容量比下不同缓存关联度的命中率变化。

当高速缓存容量较小(即 CR 较大)时,对绝大多数应用而言,增加关联度可以明显提高命中率。除了 HPCG 和 GraphX-LP 两个应用对增加关联度不敏感之外,其他应用在将关联度从 1 增加到 2 时,命中率平均可以提高 20% ~30%。由此可见,对于小容量的高速缓存,增加关联度可以有效降低冲突缺失,提高命中率。

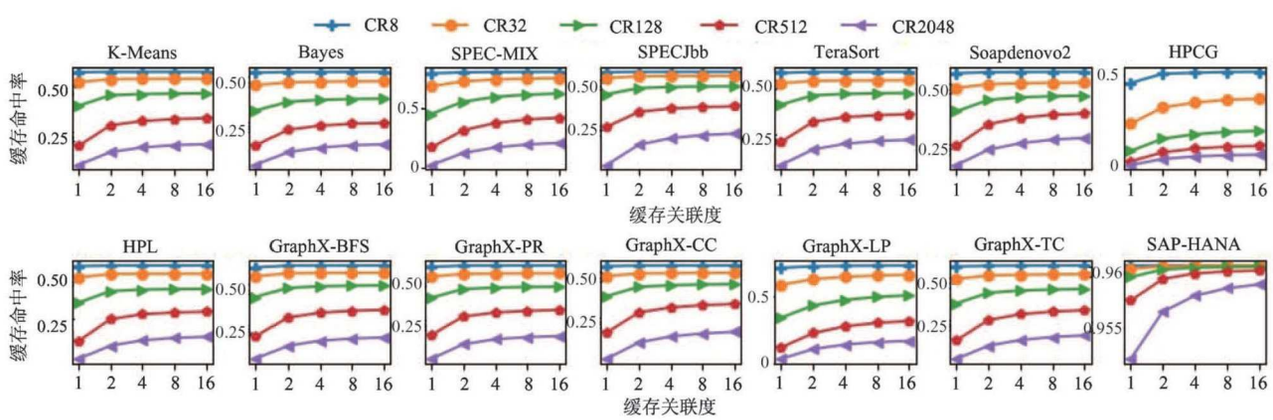

图 5 不同缓存关联度对缓存命中率的影响

然而,随着高速缓存容量的加大,冲突缺失的程度会逐渐减轻,增加关联度对提高命中率的作用也逐渐减小。如图 5 所示,当缓存容量较大(即 CR 较小)时,即使是对关联度比较敏感的应用,如 SPECJbb、SPEC-MIX 等应用也只能提高不到 5% 的命中率。

4.2 缓存关联度对综合性能的影响

组相联除了拥有降低冲突缺失从而提高命中率的优势,还存在引入访问缓存标记开销的劣势。正如 AlloyCache[6] 讨论的那样,直接相联的缓存可以通过将缓存标记与数据放在一起,一次性地将缓存数据和缓存标记一起读出,从而消除了缓存命中时

的访问缓存标记的开销。但是由于组相联是多路数据存储在一起,无法一次性地将整个缓存组的标记和数据取出,而是必须先访问缓存标记,再访问数据块,至少需要两次片外高速缓存的访问,因此理论上组相联的缓存命中延迟至少是直接相联的两倍。

图 6 展示了采用不同缓存关联度的内存系统的综合性能。当缓存容量较小时,增加关联度提升缓存命中率的效果占主导作用,因此高关联度可以明显提升综合性能。例如当缓存容量为主存容量的 1/128 时,用 16 路组相联的缓存要比直接相联的缓存在性能上提高 50%。当缓存容量变大时,增加关联度带来的好处减弱,命中率提升减小,而组相联带来的额外缓存标记的访问开销依然存在,因此组相联缓存的综合性能反而会低于直接相联。从图 6 的结果可以看出,在大容量缓存条件下,几乎所有的应用(GraphX-TC 除外),直接相联的综合性能要优于组相联。

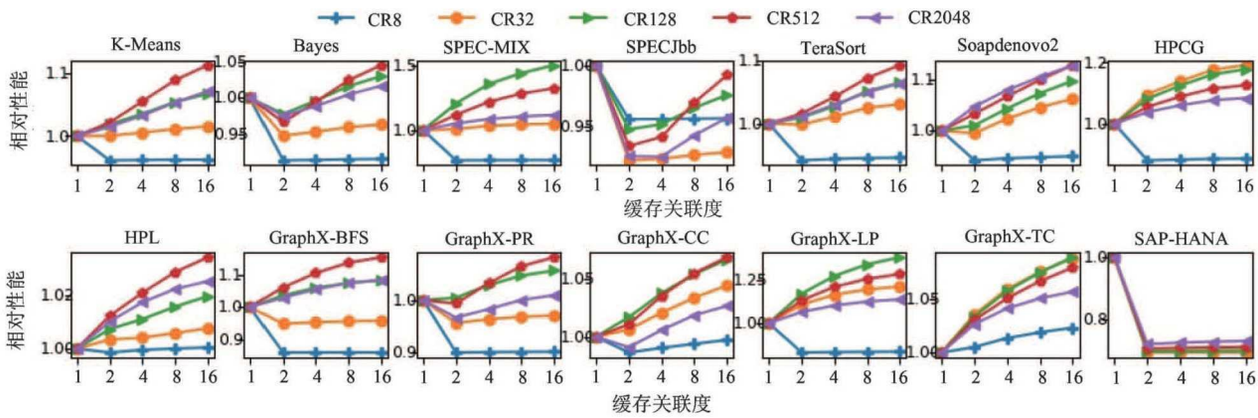

图 6　不同缓存关联度对综合性能的影响

4.3　Tag 缓存对组相联缓存的影响

为了降低组相联的命中延迟,缓存标记的缓存是一种有效并广泛使用的方法。如 ATCache[20] 描述的那样,通过保存一部分的缓存标记在 SRAM 中,作为缓存标记的缓存,如果缓存标记的缓存命中,则可以避免访问片外的缓存标记。本文评测了缓存标记的缓存在 EC-NVM 场景下的效果。本文使用了 32 路的组相联的缓存,得到了使用不同容量下的缓存标记的缓存的命中率以及内存系统整体的加速比。

从图 7 中可以看出,缓存标记的缓存可以有效减少访问片外的缓存标记,从而减少命中延迟,但是减少的效果要取决于缓存标记的缓存的容量。对 HPCG 和 SAP HANA 来说,即使增大了 Tag Cache 的容量,也无法有效地减少片外缓存标记的访问。对绝大部分应用来说,至少需要在 SRAM 中保存 1/8 的缓存标记,才能发挥比较明显的效果。

然而,缓存标记的缓存是保存在SRAM中的,

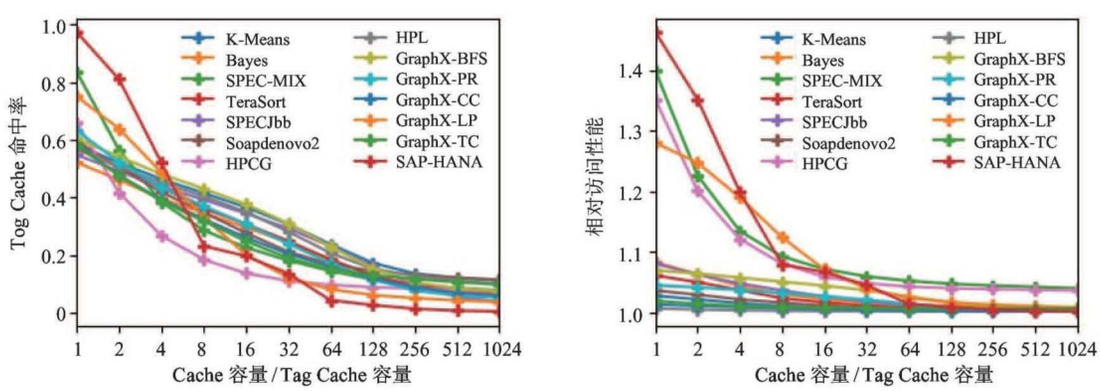

图 7　不同容量的 Tag 缓存下的 Tag 缓存命中率和综合性能

实际系统中，SRAM 的空间宝贵而且稀少。在本文的配置中，外部缓存的容量是 32 GB，关联度是 32，如果缓存粒度选择为 64 字节，需要的缓存标记存储空间大概为 448 MB。如图 7 所示，花费 56 MB 的 SRAM 可以消除 40% 的 Tag 访问，将综合性能提升 15%。而只使用较小容量的缓存标记的缓存，其作用会大大折扣，如图 7 所示，只用全部缓存容量的 1/1024 时，花费 448 kB 的 SRAM，可以降低不足 19% 的缓存标记访问，性能提升不到 3%。在实际系统中，整个 L3 的容量大多在 20 MB 左右，因此无法为缓存标记的缓存提供足够的容量。换言之，若想要缓存标记的缓存发挥明显的作用，必须在其容量开销方面进行有效的优化。

基于以上结果和分析，得到如下结论。

结论 3 组相联可以提高缓存命中率，但是 EC-NVM 的大容量缓存降低了冲突缺失，使得组相联提高命中率的效果大大降低，且无法抵消组相联引入的额外访问缓存标记的开销，因此，在大容量高速缓存场景下，组相联缓存性能反而低于直接相联缓存。

结论 4 由于 EC-NVM 的缓存容量较大，需要的缓存标记容量也更大，因此有限的缓存标记的缓存无法发挥明显作用。其设计需要妥善解决 SRAM 容量限制问题。

5 不同替换算法对 EC-NVM 结构的影响

本节评测了现有系统广泛使用的替换算法，包括随机替换算法（Random），先入先出替换算法（first in first out，FIFO），最近最少使用替换算法（least recently used，LRU），静态重用预测算法（static re-reference interval prediction，SRRIP）。此外，Belady OPT[19] 算法已经被证明是在降低缺失率方面的最优替换算法，本文称之为 OPT 算法。在本节的评测中，配置缓存为 16 路组相联，为了测试不同内存压力情况下的替换算法的效果，进行了 3 组不同高速缓存容量下的测试，结果如图 8 所示。

当高速缓存容量较大（即 CR8）时，高速缓存压力较小，即冲突缺失较少，各替换算法效果几乎没有差别。

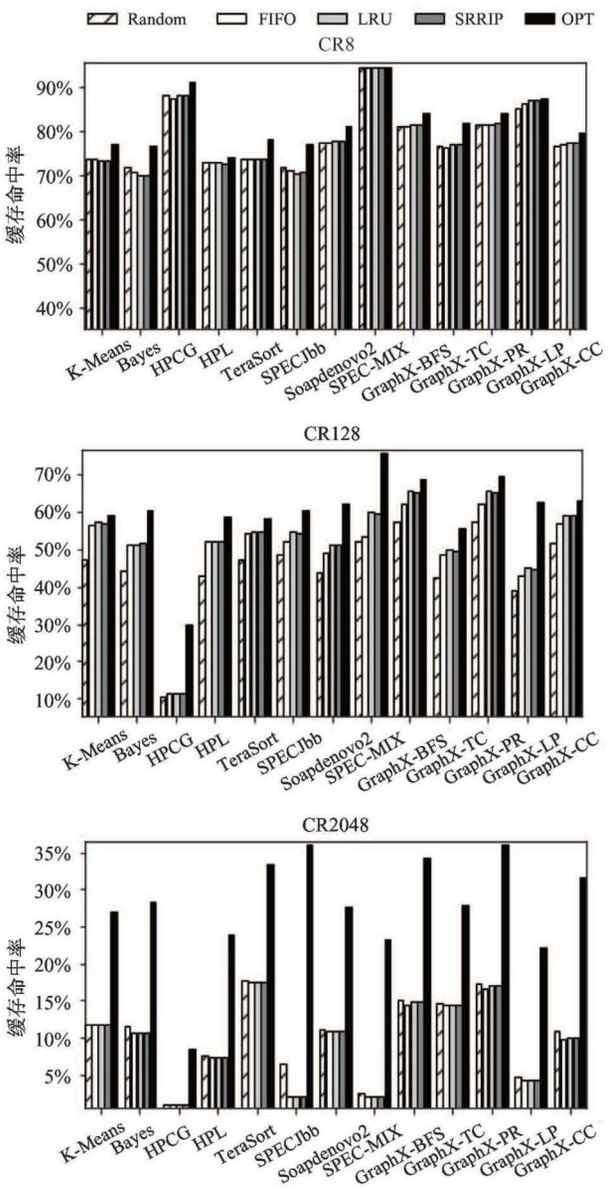

图 8 不同替换算法对 EC-NVM 缓存命中率的影响

当高速缓存容量减小（即 CR128）时，冲突缺失增多，这时替换算法减少冲突缺失的能力得以体现。Random 作为纯随机算法，在发生替换时，没有选择性地随机选取数据块进行替换，而 LRU、SRRIP 都通过观察访问历史，保护将来有可能要访问的数据块，而 OPT 算法则是在假设可以获知将来要访问的数据块的情况下的最优选择。因此相比于随机替换，LRU 和 SRRIP 都能显著提高高速缓存命中率。例如，使用 LRU 的高速缓存可以将 K-Means 的命中率相比使用 Random 的高速缓存提升 10%。

当高速缓存容量继续减小（即 CR2048）时，冲

突缺失继续增加,甚至出现“缓存冲刷”现象,即数据块进入高速缓存中还未来得及被使用,就被后续的数据块“挤走”。对这种情况下,现有的替换算法效果都无法应对,甚至 LRU、SRRIP 的效果比 Random 还要差。

本文通过重用距离(reuse distance, RD)来分析原因。文献[21]定义了重用距离,它表示的是在同个缓存组中,重复出现的两个访问之间间隔了多少不重复的数据块。重用距离是高速缓存性能分析的一个重要工具,可以表征程序的时间局部性行为。只有当重用距离超过高速缓存的关联度时才会发生高速缓存替换。在本节的实验中,计算了程序在 16 路组相联缓存下的重用距离。图 9 展示了重用距离分布。

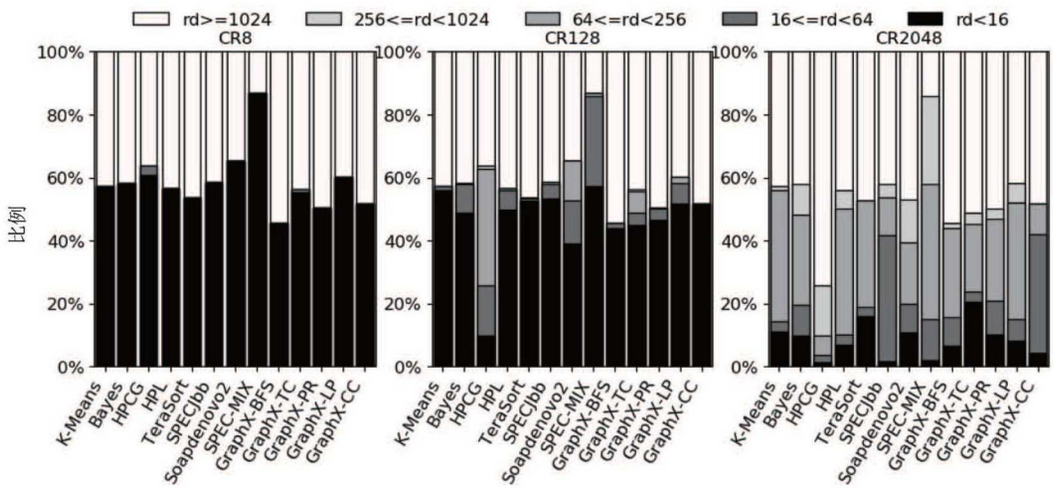

图 9　重用距离分布

不同高速缓存容量下的重用距离呈现较大差别,这是因为当高速缓存容量减小时,缓存组减少,平均分配到各个缓存组的访问增多,冲突缺失增加。例如,对一个访问序列[A, B, C, A, B, D, A, C],如果缓存组的数量为 1,得到的重用距离分布是[rd3 = 3, rd4 = 1]。但是如果缓存容量增加,缓存组的数量得以增多,假设增加到两个缓存组,那么该访问序列将会被分到两个缓存组[A, B, A, B, A]和[C, D, C]上,此时重用距离的分布将变成[rd2 = 4]。从这一例子可以看到,缓存容量的减小,会使得重用距离变得更大。

当 CR 为 8 时,重用距离分布呈现“两极化”。即超过一半的访问的重用距离小于关联度,这种情况下不需要缓存替换。而其余访问的重用距离远大于关联度,这种情况下,当前访问的数据块距离下一次访问相隔距离太远,即便对该块进行保护也会被中间的数据块挤掉,因此无法很好地发挥作用。

当 CR 为 128 时,虽然重用距离相比 CR 为 8 时变大,但是仍有超过 50% 访问的重用距离小于 64 个数据块,这时 LRU 和 SRRIP 等替换算法可以有效识别出需要保护的数据块,且这时对这些数据块的保护是有意义的,也就是说,LRU 和 SRRIP 只能对重用距离超过关联度不太多的情况有效果。

当 CR 为 2048 时,超过 80% 的访问的重用距离大于 256,这远远超出了关联度,这时候 LRU 和 SRRIP 并不能有效识别出需要保护的数据块,替换效果甚至还不如 Random。

基于以上结果和分析,得出如下结论。

结论 5　现有替换算法对重用距离较大的情况支持不好,距离最优算法还存在较大的优化空间。而重用距离较小时,各个替换算法的效果差距减小。

结论 6　EC-NVM 的大容量缓存降低了缓存压力,使重用距离大大减小,因此现有替换算法相比随机替换算法无法明显提升缓存命中率。

6　不同预取算法应用在 EC-NVM 结构上的效果

预取器可以预测将来要访问的内存地址,提前

发出对内存的访问请求，利用空闲的内存带宽提前将数据取到缓存中，避免真正需要数据时候的等待，是一种有效的加速内存访问的方法。为了研究 EC-NVM 场景下的预取算法的效果，本文评测了 4 种典型的预取算法，包括 SBP[22]，AMPM[23]，SPP[24] 和 BOP[9]。基准系统是没有预取的系统。

图 10 展示了预取算法的效果。对于绝大多数应用，现有的几种预取算法都可以明显提升性能。平均而言，AMPM 表现最好，可以将整体性能提高 19.6%；SBP 次之，可以将性能提高 18%；SPP 可以提升 14.5%，BOP 可以提升 10%。相较与其他算法，AMPM 最稳定，在所有测试中，其预取效果都是最优或者次最优的。

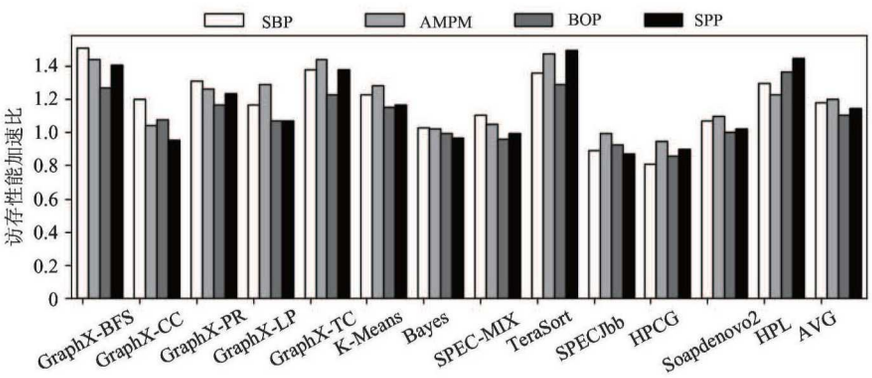

图 10　不同预取算法的系统性能加速比

结合图 3 分析，对于空间局部性较好的程序（BFS、CC、PR），SBP 算法效果最好，这是由于 SBP 对地址连续访问有较好效果。而对于 HPCG、SPECJbb 等空间局部性不太明显的应用而言，SBP 的预取效果则不明显。

预取算法的效果主要受 3 个因素影响。（1）预取利用率：如果一个数据块被预取回来之后被后续访问用到，称这个预取操作是能有效的预取（useful prefetch）。预取利用率是有效的预取数量与发出的总预取数量的比值。（2）预取迟到率：如果一个数据块的预取请求已经发出且还没有完成，而后续又有内存请求访问数据块，称这样的预取是迟到的预取（late prefetch）。预取迟到率是迟到的预取数量与发出的总预取数量的比值。根据 FDP[8] 的描述，有效的预取是对访存性能的完整加速，而迟到的预取对访存性能只能做到部分加速甚至降低性能。（3）预取错误率：如果一个数据块被预取回来之后不会被用到，对该数据块的预取为错误的预取。预取错误率是错误的预取数量与发出的总预取数量的比值。

图 11 展示了对影响预取效果的 3 个因素的分析。每个应用对应 4 个数据条目，分别对应 4 种预取算法：SBP、AMPM、BOP、SPP。平均而言，SBP 和 AMPM 拥有较高的准确率，但是 SBP 比 AMPM 多 5% 的无效预取，因而综合性能仍不如 AMPM。对于预取迟到率，现有预取算法都有较明显的迟到现象，AMPM 平均有 8% 的迟到的预取，BOP 平均有 16% 的迟到预取，SPP 有 15% 的迟到预取。相比之下，SBP 的迟到预取最少，这是因为 SBP 识别的内存访问都为连续访问，其发出的预取和访问也都是地址连续的。美国加州大学圣迭戈分校（UCSD）非易失内存系统实验室（NVSL）的研究人员已经通过对 EC-NVM 内存结构评测发现，非易失主存和片外 DRAM 高速缓存的性能差距主要体现在非连续访问的性能[25]。也就是说地址连续的预取可以更快地被完成，因而其迟到率也会更低。

基于以上实验结果和分析，得出如下结论。

结论 7　由于非易失性主存较高的访问延迟，EC-NVM 结构存在较明显的预取迟到问题。但是，以地址连续的预取为主的预取算法可以有效改善预取迟到问题。

结论8　由于大数据应用在EC-NVM场景下具

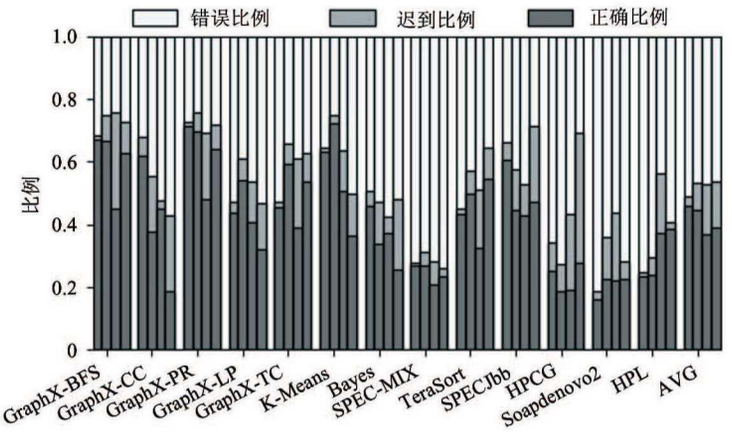

图 11　预取效果分析

有良好的空间局部性,现有多种预取算法都可以对 EC-NVM 结构有明显的加速效果。

7　相关工作

关于 DRAM 缓存的工作大多关注的是堆叠式的 DRAM 缓存。但是,相比较于堆叠式的缓存,外部 DRAM 缓存无论在容量还是延迟上都有很大差异。这些差异造成了评测方法及体系结构特性上的不同。

关于评测方法,现有研究对内存结构的评测大多采用 SPECCPU2006[11]、PARSEC[26] 等标准测试集。这些应用能够较好地评测传统内存结构的体系结构特性,但是对 EC-NVM 结构的评测,它们无法提供足够大的工作集。一些对 3D 堆叠式 DRAM 缓存的测试用到了 CloudSuite[27] 等应用,可以提供比 SPECCPU2006 等大一些的工作集。但是相对于 EC-NVM 动辄数十 GB 的高速缓存容量,这些应用还是不能提供足够规模的工作集。BigdataBench[13] 选取了多种有代表性的大数据应用,可以很好地对大数据应用场景进行评测。但是 BigdataBench 是对各种类型的大数据应用的多方面评测,相当一部分的应用并非内存密集型,而是计算密集型或 IO 密集型。对于这些应用,优化内存系统结构并不能明显提高系统的综合性能,因而无法直接使用 BigdataBench 的全部应用,需要进行选择和补充。因此,考虑到应用的多样性和足够大的工作集,本文选择了更多的大数据应用,相比于其他对 DRAM 高速缓存的评测,本文使用的评测应用可以提供更大的工作集,也更具有多样性和代表性。

关于高速缓存粒度的选择,近年来一直有研究人员进行探索和改进。例如 Amoeba Cache[28] 讨论了高速缓存粒度对内存系统性能的影响,其实验表明,在 IC-DRAM 场景下,对于大多数程序,使用 64 字节可以得到系统的最优性能。但是对于很多空间局部性比较好的程序,适当增大高速缓存粒度可以有效提高访存性能。该文章提出使用变粒度的高速缓存而不是局限于某个固定的粒度,并且提出了一种动态调节高速缓存粒度的方法,可以根据应用特点选择最优的缓存粒度。Unison Cache[29] 和 Bimodal Cache[30] 都提出,对于 3D 堆叠式的 DRAM 高速缓存,应用可以从空间局部性中获取较大的好处,因此采用更大的缓存粒度(4 kB)是比较好的选择。对于 EC-NVM 场景下的缓存粒度,文献[25]实测了使用不同访问粒度时英特尔傲腾内存的实际性能表现。结果表明,随着访问粒度的增加,系统性能会明显提高,但是当粒度增加到一定程度,性能提高的趋势会逐渐变缓,当粒度超过1 kB时,继续增大粒度则无法提高性能,甚至粒度超过 16 kB 时,性能会有所降低。这和本文通过模拟器评测得出的结论比较接近。

关于高速缓存的关联度,现有的工作也有一些探讨。例如,文献[31]发现虽然增大缓存容量会弱化增加关联度对提升命中率的作用,但是个别 SPECCPU2006 中的应用在大容量高速缓存的场景下,仍然可以通过增加关联度提高命中率。这跟本

文在 EC-NVM 场景下得到的结果略有不同,这主要是由于 EC-NVM 场景下,高速缓存容量比 IC-DRAM 场景更大,增加关联度提升命中率的作用会更弱。文献[32]发现关联度从 1 增加到 2 时,命中率会有明显提高,但是关联度继续增加时,命中率提高得并不明显。这和本文在小容量高速缓存情况下得到的结论比较接近。关于缓存标记的缓存,ATCache[20]用 0.4% 的缓存标记的容量代价,减少了 60% 的缓存标记的访问。但是本文的实验结果显示在 EC-NVM 中,缓存标记的缓存要想像 ATCache 描述的那样发挥明显的作用,需要更大的 SRAM 空间,即由于 EC-NVM 的缓存标记总量更大,有限的 SRAM 空间无法使缓存标记的缓存发挥明显的作用。

关于高速缓存的替换算法,学术界有很多此方面的研究,工业界也已经有很多著名的替换算法被广泛使用。其中很多算法都是利用访问历史,预测将来更大可能不会被用到的数据块进行替换,例如 LRU 替换距离上一次访问时间最长的数据块。SRRIP 在 LRU 的基础上,一定程度上保护了重用距离比较大的数据块。相比 LRU,SRRIP 将性能提高大约 10%[33]。但是,根据本文的实验结果,在 EC-NVM 场景下,由于冲突缺失减弱,更多的缓存组导致每个缓存组访问压力减小,重用距离变小,使得现有替换算法相比于随机替换算法不再有明显的效果。

关于预取算法,尤其是针对大数据应用的预取算法的研究一直受到学术界的关注。例如美国德州大学奥斯汀分校的研究人员评测了大数据程序尤其是云应用的预取效果[34]。研究者发现对于大数据云应用,大容量的高速缓存以及简单的流式预取即可取得较高的性能提升。本文选取了 4 种有代表性的预取算法,其中 SBP[22]是一种经典且被广泛使用的预取算法,拥有多个变种,它通过跟踪各个内存区域的访问历史,探测可能存在的访问流(地址连续的序列),一旦找到,则发出预取。AMPM[23]使用位图的方式记录多个内存区域的访问模式,一旦某个区域的模式再次出现,则根据已经学习到的历史模式进行匹配并发出预取。SPP[24]是一种典型的查询预取算法。它通过跟踪和记录每个页面的“签名”的方式,利用签名将页面分类,记录每个类别的页面的访存模式,一旦某个类别的模式被频繁访问,则触发预取。BOP[25]是一种考虑预取迟到率的算法,为了缓解预取迟到率的问题,现有预取器通常采用增大“预取距离”的方式。在预取算法中,发出的预取和当前访问的内存地址间隔被称为“预取距离”。BOP 同时使用多个预取距离进行预取,根据预取效果反馈选择最合适的预取间隔,从而达到准确率和迟到率的平衡。本文的评测结果显示预取算法对 EC-NVM 结构优化有显著效果,且仍有较大的提高空间。

8 结 论

本文评测了大数据应用背景下的 EC-NVM 结构的体系结构特性。为了更好地模拟 EC-NVM 的使用场景,本文构建了一套大内存应用程序测试集,通过评测发现了 EC-NVM 的一些新的体系结构特性。第一,EC-NVM 场景下的大数据应用具有不同的空间局部性,因此缓存粒度的选择不能直接沿用现有的 64 字节,需要综合考虑高速缓存命中率和命中延迟的平衡,本文的实验结果显示适当增大缓存粒度(128 B、256 B)对 EC-NVM 是更好的选择;第二,由于片外高速缓存标记的访问开销,直接相联在综合性能上要优于组相联;第三,由于冲突缺失的减少,高级替换算法相比随机替换算法没有明显的作用;第四,预取技术对 EC-NVM 结构的性能优化具有显著效果,且预取算法的优化仍有较大的提高空间。非连续访问造成的预取迟到现象仍然比较严重,可以作为后续预取算法研究的一个重要改进方向。此外,本文还通过分析空间局部性和时间局部性解释了上述观察结果。所有的这些发现都为 EC-NVM 结构的优化设计提供了有价值的参考。

参考文献

[1] Färber F, Cha S K, Primsch J, et al. SAP HANA database: data management for modern business applications [J]. *ACM Sigmod Record*, 2012, 40(4): 45-51

[2] Zaharia M, Xin R S, Wendell P, et al. Apache spark: a unified engine for big data processing[J]. *Communica-*

tions of the ACM, 2016, 59(11): 56-65

[3] Qureshi M K, Srinivasan V, Rivers J A. Scalable high performance main memory system using phase-change memory technology[C]//Proceedings of the 36th Annual International Symposium on Computer Architecture, Austin, USA, 2009: 24-33

[4] Intel. Revolutionizing memory and storage[EB/OL]. https://soundcloud.com/intelchipchat/intel-optane-technology: Intel, 2018

[5] Intel. Optane DC persistent memory DIMMs push latency closer to DRAM[EB/OL]. https://www.pcper.com/news/Storage/Intels-Optane-DCPersistent-Memory-DIMMs-Push-Latency-Closer-DRAM: Intel, 2018

[6] Qureshi M K, Loh G H. Fundamental latency trade-off in architecting dram caches: outperforming impractical SRAM-tags with a simple and practical design[C]//2012 45th Annual IEEE/ACM International Symposium on Microarchitecture, Washington DC, USA, 2012: 235-246

[7] Loh G H, Hill M D. Efficiently enabling conventional block sizes for very large die-stacked DRAM caches[C]//Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture, Porto Alegre, Brazil, 2011: 454-464

[8] Srinath S, Mutlu O, Kim H, et al. Feedback directed prefetching: improving the performance and bandwidth-efficiency of hardware prefetchers[C]//The IEEE 13th International Symposium on High Performance Computer Architecture, Phoenix, USA, 2007: 63-74

[9] Michaud P. Best-offset hardware prefetching[C]//2016 IEEE International Symposium on High Performance Computer Architecture, Barcelona, Spain, 2016: 469-480

[10] Nitu V, Kocharyan A, Yaya H, et al. Working set size estimation techniques in virtualized environments: one size does not fit all[J]. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 2018, 2(1): 19

[11] Henning J L. SPEC CPU 2006 benchmark descriptions[J]. *ACM SIGARCH Computer Architecture News*, 2006, 34(4): 1-17

[12] Barata M, Bernardino J, Furtado P. An overview of decision support benchmarks: Tpc-ds, TPC-H and SSB[J]. *New Contributions in Information Systems and Technologies*, 2015: 619-628

[13] Wang L, Zhan J, Luo C, et al. Bigdatabench: a big data benchmark suite from internet services[C]//2014 IEEE 20th International Symposium on High Performance Computer Architecture, Orlando, USA, 2014: 488-499

[14] Gene Data. http://www.bio8.cs.hku.hk/dataset

[15] Poremba M, Zhang T, Xie Y. Nvmain 2.0: a user-friendly memory simulator to model (non-) volatile memory systems[J]. *IEEE Computer Architecture Letters*, 2015, 14(2): 140-143

[16] Hamerly G, Perelman E, Lau J, et al. Simpoint 3.0: faster and more flexible program phase analysis[J]. *Journal of Instruction Level Parallelism*, 2005, 7(4): 1-28

[17] Liu Y, Sun X H. LPM: a systematic methodology for concurrent data access pattern optimization from a matching perspective[J]. *IEEE Transactions on Parallel and Distributed Systems*, 2019(1): 2478-2493

[18] Bao Y, Chen M, Ruan Y, et al. HMTT: a platform independent full-system memory trace monitoring system[J]. *ACM SIGMETRICS Performance Evaluation Review*, 2008, 36(1): 229-240

[19] Puzak T R. Analysis of Cache Replacement-Algorithms[D]. Amherst: University of Massachusetts, 1985

[20] Huang C C, Nagarajan V. ATCache: reducing DRAM cache latency via a small SRAM tag cache[C]//Proceedings of the 23rd International Conference on Parallel Architectures and Compilation, Edmonton, Canada, 2014: 51-60

[21] Ding C, Zhong Y. Predicting whole-program locality through reuse distance analysis[C]//Proceedings of the ACM SIGPLAN 2003 Conference on Programming Language Design and Implementation, San Diego, USA, 2003: 245-257

[22] Liu G, Huang Z, Peir J K, et al. Enhancements for accurate and timely streaming prefetcher[J]. *J Instr Level Parallelism*, 2011, 13: 1-15

[23] Ishii Y, Inaba M, Hiraki K. Access map pattern matching for high performance data cache prefetch[J]. *Journal of Instruction-Level Parallelism*, 2011, 13(2011): 1-24

[24] Kim J, Pugsley S H, Gratz P V, et al. Path confidence based lookahead prefetching[C]//The 49th Annual IEEE/ACM International Symposium on Microarchitecture, Taibei, China, 2016: 1-12

[25] Izraelevitz J, Yang J, Zhang L, et al. Basic performance measurements of the Intel Optane DC persistent memory module[J]. *arXiv*:1903.05714, 2019

[26] Bagrodia R, Meyer R, Takai M, et al. Parsec: a parallel simulation environment for complex systems [J]. *Computer*, 1998, 31(10): 77-85

[27] Ferdman M, Adileh A, Kocberber O, et al. Clearing the clouds: a study of emerging scale-out workloads on modern hardware[J]. *ACM SIGPLAN Notices*, 2012, 47(4): 37-48

[28] Kumar S, Zhao H, Shriraman A, et al. Amoeba-cache: adaptive blocks for eliminating waste in the memory hierarchy[C] // The 45th Annual IEEE/ACM International Symposium on Microarchitecture, Vancouver, Canada, 2012: 376-388

[29] Jevdjic D, Loh G H, Kaynak C, et al. Unison cache: a scalable and effective die-stacked DRAM cache[C] // 2014 47th Annual IEEE/ACM International Symposium on Microarchitecture, Cambridge, UK, 2014: 25-37

[30] Gulur N, Mehendale M, Manikantan R, et al. Bi-modal dram cache: improving hit rate, hit latency and bandwidth[C] // The 47th Annual IEEE/ACM International Symposium on Microarchitecture, Cambridge, UK, 2014: 38-50

[31] Tschirhart P, Stevens J, Chishti Z, et al. The case for associative dram caches[C] // Proceedings of the 2nd International Symposium on Memory Systems, Alexandria USA, 2016: 211-219

[32] Young V, Chou C, Jaleel A, et al. Accord: enabling associativity for gigascale dram caches by coordinating way-install and way-prediction[C] // 2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture, Los Angeles, USA, 2018: 328-339

[33] Jaleel A, Theobald K B, Steely Jr S C, et al. High performance cache replacement using re-reference interval prediction (RRIP)[J]. *ACM SIGARCH Computer Architecture News*, 2010, 38(3): 60-71

[34] Wang J, Panda R, John L K. Prefetching for cloud workloads: an analysis based on address patterns[C] // 2017 IEEE International Symposium on Performance Analysis of Systems and Software, Santa Rosa, USA, 2017: 163-172

Characterizations and architectural implications of NVM's external DRAM cache

Pan Haiyang [*,**], Liu Yuhang [*,**,***], Lu Tianyue [*], Chen Mingyu [*,**,***]

([*] Advanced Computing System Laboratory, Institute of Computing Technology, Beijing 100190)

([**] University of Chinese Academy of Sciences, Beijing 100049)

([***] Peng Cheng Laboratory, Shenzhen 518055)

Abstract

To satisfy the memory capacity requirement of big-data applications, a promising architecture, in which the non-volatile memory (NVM) is used as main memory and dynamic random access memory (DRAM) is taken as external cache (EC), is proposed, referred to as EC-NVM. Compared to traditional memory architecture that involves the last level cache (LLC) and DRAM (referred to as IC-DRAM), EC-NVM is significantly different in terms of capacity and latency, thus traditional design cannot be directly borrowed without reconsideration. This paper explores the characteristics and architectural implications of EC-NVM in diverse dimensions including cache line granularity, associativity, replacement policy and prefetching methods. All findings provide valuable hints for the designers of EC-NVM.

Key words: non-volatile memory (NVM), dynamic random access memory (DRAM) cache, big-data application