BATMAN：混合存储系统的带宽最大化利用

摘要：高带宽存储技术例如 HMC,HBM 和 WideIO 虽然提供了比标准型 DRAM 高 4 倍-8 倍的带宽，但保留了相似的随机访问延迟。高带宽的存储器的有限容量要求它们需要同传统的 DRAM 存储器联合构建。例如一个系统被设计为将高带宽存储器作为 Near Memory（NM）以及将标准型 DRAM 作为 Far Memory(FM),并且要依靠能提高 NM 命中率的技术。我们研究显示，传统的优化 NM 命中率的方法效率很低，它并不能最大化地利用整个系统的可使用带宽。例如，当被申请的工作集全部都在 NM 中，全部的访问仅仅靠 NM 服务，然而 FM 的带宽根本没被利用。我们研究显示，要改进系统的性能，需要优化整个系统的带宽而不是 NM 的命中率。

我们观察到当访问每一个存储系统中 NM 和 FM 的带宽成一定比例时，性能能达到最大化。我们的提出 Bandwidth Aware Tiered-Memory Management(带宽感知分层内存管理)简称 BATMAN，它能在运行时不需要依靠预先了解关于申请访问模式信息就能完成访问分配。一种是系统在 NM 和 FM 之间不用执行数据迁移，BATMAN 在 NM 和 FM 之间随机划分工作集从而保证目标访问分配。一种是系统在 NM 和 FM 之间需要执行数据迁移（例如基于操作系统的页面迁移或者基于硬件的高速缓存块的迁移），BATMAN 能完成目标访问分配通过监控运行中的访问分配，并且利用这些信息去调节数据迁移速率。我们在对带有 4GB 的 NM 和 32GB 的 FM 的系统的评价显示 BATMAN 改善了大约 40%性能，（平均来看，10%在于没有数据迁移，10%在基于操作系统的页面迁移，22%在硬件高速缓存上），同时只需要少于 12 字节的存储开销。

I 介绍

存储器技术例如混合式存储晶体管（HMC），高带宽存储器（HBM）和宽 I/O（Wide I/O）都是为了迎合高带宽存储器的发展需求。虽然这些存储技术有较高的存储带宽，但是它们都有相似的内存数组访问延迟，如图 1 所示。与标准型 DRAM 对比来看设这些模块被设计为有一定限度的容量（1GB-8GB）。因此，它们不能完全替代标准型 DRAM。相反，这些高带宽存储器技术被用来与标准型 DRAM 结合去组成一个混合存储系统或者一个分层存储系统。

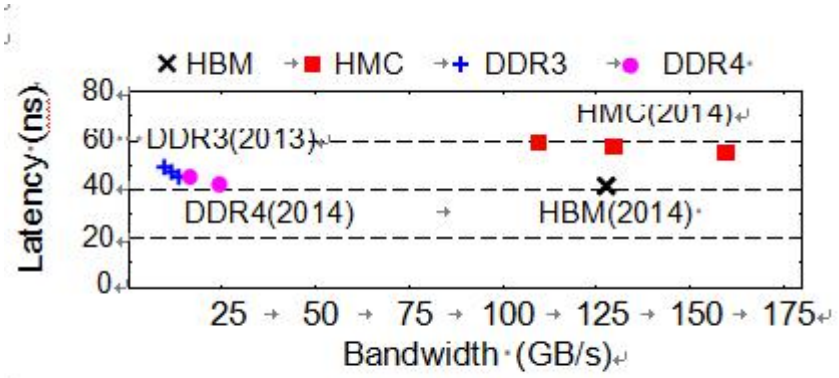


Fig. 1. Latency and bandwidth of conventional and emerging DRAM technologies (data obtained from [1, 2, 3, 4, 5])

一个混合存储系统通常由低容量高带宽的存储器，我们称之为 NM，和高容量低带宽的标准型 DRAM 组成，我们称之为 FM。在混合存储系统中的 NM 既被配置作为硬件管理的 DRAM 高速缓存，也作为操作系统可视的主存的一部分。在这

两种情况下，这个系统试图去最大化由 NM 去满足的总存储器的请求所占的分数。例如，如果 NM 被用作硬件管理的高速缓存，则对 FM 中的 cache 块的访问将在 NM 中设立请求的高速缓存块，目的是高速缓存块的后续引用将由 NM 满足（提高 Cache 命中率）。相似地，如果 NM 被作为操作系统可视的主存储器，那么操作系统将试图最大化 NM 中被分配给给定的应用程序的页面的数量。在这两方面潜在的评估是对服务这个工作集高带宽 NM（并不是低带宽的 FM）将提供最高的系统性能。我们研究显示这些传统的方法去使 NM 命中率最大化是低效的，因为它们不可能完全利用整个系统的带宽。例如，当应用程序的工作集适合在 NM 中时，FM 的带宽保持并未使用。我们研究显示，要改进系统的性能，需要优化整个系统的带宽而不是 NM 的命中率。

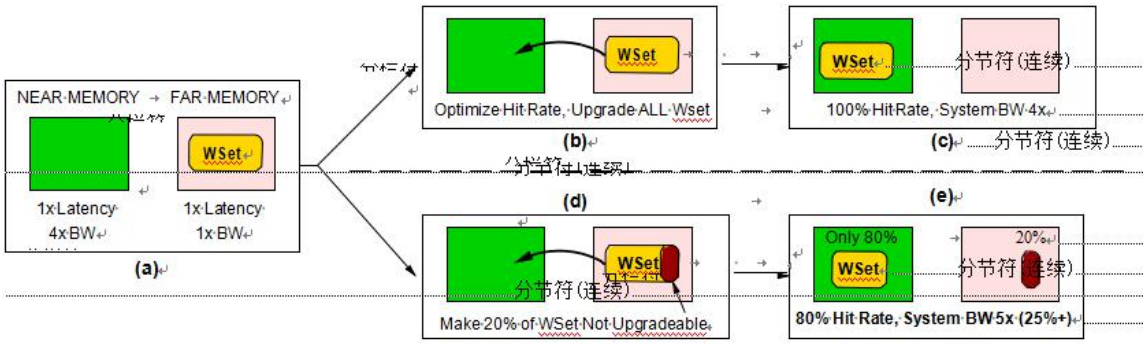


Fig. 2. Optimizing for Hit Rate versus for System Bandwidth. (a) a system with near memory and far memory, both having the same latency but near memory has 4x the bandwidth, (b) and (c) traditional systems that try to optimize for hit rate gives up to 4x system bandwidth, (d) and (e) explicitly controlling some part of the working set to remain in far memory results in up to 5x system bandwidth.

我们用一个例子解释优化系统带宽的而不是 NM 的命中率。图 2 显示的是一个混合存储系统，其中 NM 和 FM 有相同的延迟，但是 NM 有 4 倍于 FM 的带宽。不失一般性，我们考虑到一个应用，它的工作集较小，足够去适应在 NM 和 FM 中。图 2（b）和（c）显示了传统的方法去最大化 NM 的命中率。在访问 FM 中的数据时数据元素将迁移至 NM 中，正如图 2（b）所示。随着时间推移，该应用的整个工作集都将迁移到 NM，并且所有的访问都将仅靠 NM 服务。FM 将不再收到任何访问，并且整个系统的带宽将由 NM 决定，变成最多 4 倍，正如图 2（c）所示。

现在，考虑一个可供选择的方法去最大化整个系统的带宽。例如，我们能识别工作集中占 20%的负责给定应用程序存储器访问的部分，并且标记它为不合适的去从 FM 升级到 NM。一次访问 FM 中的数据项将不会迁移这个数据到 NM 中。因此，在稳定状态下，在平均情况下，NM 将处于服务状态，占总访问的 80%，并且 FM 处于服务状态，占总访问的 20%。因此，整个系统的带宽将达到 5 倍，其中存储器业务在 NM 和 FM 之间平衡。因此，尽管它意味着更低的 NM 命中率，但显式地控制由 NM 和 FM 服务的存储器访问的分数可以导致更高的聚合系统带宽，因此也改善了系统性能。我们提出了 BATMAN，它的目的就在于最大化整个存储系统的带宽。

为了获得最高的性能，BATMAN 必须分配每个存储器的分别访问 NM 和 FM 的带宽成一定比例。对于不需要在 NM 和 FM 之间执行数据迁移的系统，这个目标访问划分在 NM 和 FM 中通过监控 NM 中页面的访问分配得以实现，这样它们仅仅占总内存访问的被给定百分比（据说 80%）。很不幸，页面访问计数通常来说是不可用的，并且每一页在运行过程中试图生成访问计数是相当复杂的。理想情况下，

我们想要不依靠页面访问计数实现目标分离。BATAMAN 想出一种解决方法，对存储带宽来说如果一个应用的工作集被随机划分成一定比例，对 NM 和 FM 的访问也将能被划分成相应的比例。换句话说，如果 NM 在存储系统中提供整个带宽的 80%，那么保留从 NM 工作集中随机选择的页面的 80％应该提供接近 80％的访问（不管在页面访问模式的变化，只要这些变化不是高病理性的）。我们说明了提出的通过 BATMAN 随机划分工作集的方式在获得目标访问速率通过利用分析模型和广泛地试验评价是相当有效的。

我们也分析 BATMAN 设置 NM 作为操作系统可视的主存储器的这类系统，操作系统可以在 NM 和 FM 中迁移页面。在这类系统，BATMAN 达到目标访问速率通过显示地控制在运行过程中被 NM 服务的访问数量，并且利用这个信息去动态地调节 NM 和 FM 的迁移速率和迁移方向。我们研究显示我们提出的设计，仅产生 12 字节的开销，在达到 NM 目标访问速率上是相当有效的。

最后，我们也研究研究用于 OS 透明系统的 BATMAN，它完全用 NM 作为硬件管理高速缓存。在这种情况下，BATMAN 监控访问率（包括读和写）。如果访问率超出了预先设定的临界值，BATMAN 动态地禁用 cache 的一部分去减少访问率以致到达期望的水平。这样做确保了 FM 能够服务于工作集的给定比例，最大化了整个系统的带宽和性能。

本文作出以下贡献：

1、我们认识到达到更高的性能是优化整个系统带宽而不是靠 NM 的命中率，并且提出了 BATMAN，即是对于每个存储器带宽，按一定比例分配访问 NM 和 FM。

2、我们展示了一个静态地映射 NM 和 FM 之间的页面的系统的 BATMAN。我们通过分析和实验显示，我们的工作集的随机划分方式在适应 NM 中的应用达到目标访问速率上是相当有效的。

3、我们设计的在 NM 和 FM 中发生数据迁移的 BATMAN（基于操作系统页面调度和基于硬件的高速缓存块调度），并且研究显示在达到目标访问速率上基于运行时候的活动控制数据迁移率是有效的。

我们在带有 4GB 的 NM(在 100 GBps)和 32GB 的 FM（在 25 GBps）的 16 核系统中评估 BATMAN，在没有数据迁移的系统中 BATMAN 平均提速 10%，基于操作系统页面迁移的的系统中平均提速 10%,在用 NM 作为高速缓存器的系统中提速 22%。

Ⅱ 背景和动机

高带宽存储器技术的出现，例如 HBM,HMC 和 WideIO 能帮助高性能系统克服内存带宽的问题。这些技术提供了比标准型 DDR 存储器高 4 倍-8 倍的带宽改进，同时也保留了相似的随机访问延迟。不幸的是，制造和成本约束限制了这些高带宽存储器技术变成小存储容量（几 GB），因此高带宽存储器技术是广泛地联合标准型的 DRAM 一起使用。在一些系统中，低容量高带宽被用作 NM，标准型的存储器被用作 FM。NM 能被构建为操作系统管理的主存储器也能被构建为硬件管理的 DRAM 高速缓存器。在这两种情况下，决定将 NM 并入系统的有效性的关键因素是 NM 服务的访问的百分比。

A 传统的方式:优化命中率

当应用的工作集大于 NM 的可用容量时，NM 和 FM 的数据总线都提供数据给处理器。但是，当应用的工作集小于 NM 的可用容量时，传统的方法是试图去保留所有的数据在 NM 中，正如存储器从 NM 中请求服务可以获得更高的带宽。因此，传统的方法试图去最大化 NM 的命中率。当应用的工作集适合在 NM 的可用容量中时，仅仅只有 NM 的数据总线被利用，然而 FM 的数据总线则仍处于空闲状态。因

此，传统的设计是试图去优化命中率通过未充分利用牺牲系统带宽。理想情况下，为了最大化整个系统的带宽和性能，我们更愿意充分利用 NM 和 FM 的可用带宽。

B 我们的观察：优化系统带宽

我们能通过调节 NM 提供服务的访问的百分比确保 NM 和 FM 的带宽都能得以利用。我们通过实验证实这一假设。我们使用类似于图 2 的存储器系统配置来研究一组内存密集型 STREAM 基准（实验方法在 SectionⅢ）。正如 STREAM 基准的工作集是少于 NM 的容量，基线配置完全从 NM 服务 STREAM 基准的所有访问。我们进行敏感性研究，其中我们通过在 FM 中显式地分配工作集的一部分来控制由 NM 服务的访问的百分比。图 3 显示了与基准系统相比的加速，因为从 NM 维护的工作集的分数从 10%增加到 100%。我们研究 NM 和 FM 之间的带宽比为 2X，4X 和 8X。对于所有系统，性能的峰值出现得早于 100%，验证了我们的假设。

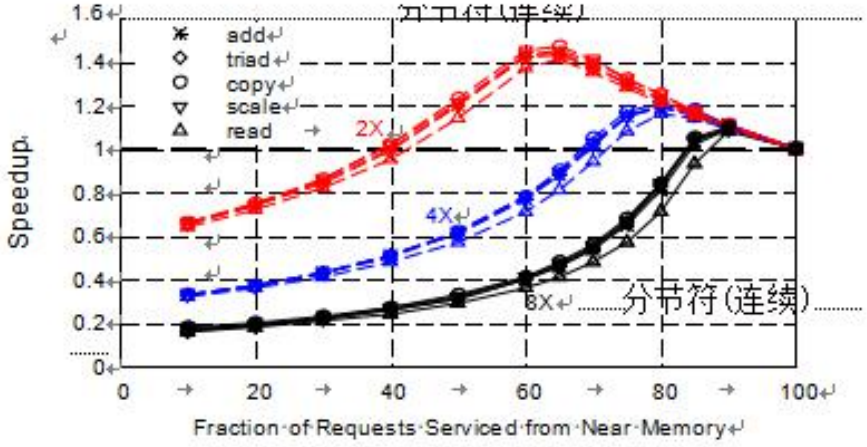


Fig. 3. Speedup versus Service Rate of Near Memory. Note the peak performance occurs much earlier than 100% (baseline system).

当 NM 与 FM 的带宽比为 4 倍时，我们得到 1.2 倍的性能峰值，这接近于我们通过将系统带宽提高 25%所期望的 1.25 倍的理想加速比（系统带宽从 4 倍增加到 5 倍）。类似的观察结果适用于具有 2 倍和 8 倍带宽比的系统。因此，当 FM 占整个系统带宽的很大一部分时，通过优化系统带宽有更大的改进潜力。

C 确定访问的最佳分布

当 NM 具有 2 倍的 FM 带宽时，我们观察到当 NM 的命中率大约为 66%时，出现性能峰值，这意味着 NM 服务的访问为 2/3 和 FM 服务的访问为 1/3。 同样，当 NM 具有 4 倍的带宽，当 NM 的命中率大约为 80%时出现性能峰值：NM 服务的访问为 4/5，FM 服务剩余的访问为 1/5。对于 8 倍 NM 带宽，当 8/9 访问由 NM 服务且 1/9 由 FM 服务时，出现性能峰值。这些结果显示，对于最高的整体系统性能，访问必须按照与各个存储器子系统提供的相对带宽成比例地分布在 NM 和 FM 之间。

D 目标：简单有效的访问分配

我们寻求一种机制，可以分配在 NM 和 FM 之间的存储器访问，与每个存储器的带宽比成比例。如果我们对给定工作集中的所有页面进行页面访问计数，则可以决定哪个页面在 NM 中以在 NM 中获得期望的访问速率。不幸的是，页面访问计数要么需要配置文件信息，要么需要复杂的运行时机制。理想情况下，我们希望

我们的机制在没有应用程序访问模式的任何预先信息的情况下也有效。为此，我们提出带宽感知分层内存管理（BATMAN）。我们针对在 NM 和 FM（第 IV 节）之间不执行数据迁移的系统开发 BATMAN，针对在 NM 和 FM（第 V 节）之间执行基于操作系统的页面迁移的系统以及使用 NM 作为硬件管理的缓存的系统 （第六节）。我们首先描述我们的实验方法，然后描述我们的解决方案。

Ⅲ 实验方法

A 系统配置

我们使用详尽的事件驱动 x86 模拟器来建模 16 核系统。核心参数（表 1），缓存层次组织和延迟松散地分布在基于英特尔酷睿 i7 处理器。每个核心是一个 4 倍宽的无序处理器，带有运行在 3.2GHz 的 128 条目 ROB。内存系统包含一个三级缓存层次结构（L1 和 L2 是私有的，L3 在所有内核之间共享）。缓存层次结构中的所有层次都使用 64B 行大小。

TABLE I

BASELINE SYSTEM CONFIGURATION

Processors	
Number of Cores Frequency Core Width	16 3.2GHz 4 wide out-of-order
Prefetcher	Stream Prefetcher
Last Level Cache	
Shared L3 Cache	16MB, 16-way, 27 cycles
Near Memory (Stacked DRAM)	
Capacity Bus Frequency Channels Banks Bus Width tCAS-tRCD-tRP-tRAS	4GB 800MHz (DDR 1.6GHz) 8 16 Banks per rank 64 bits per channel 36-36-36-144 CPU cycles
Far Memory (Commodity DRAM)	
Capacity Bus Frequency Channels Banks Bus Width tCAS-tRCD-tRP-tRAS	32GB 800MHz (DDR 1.6GHz) 2 16 Banks per rank 64 bits per channel 36-36-36-144 CPU cycles

我们的异构存储器子系统包括使用 HBM 技术的 4GB 的近存储器（NM）和使用常规 DDR 技术的 32GB 的远存储器（FM）。我们的 DRAM 模拟器类似于 USIMM，具有用于每个存储器通道的读取和写入队列。DRAM 控制器优先进行写入读取，并且批量发出写入。我们使用 Minimalist-Open 页面作为 DRAM 地址映射，并使用开放页面策略作为默认值。我们假设两种 DRAM 技术中有相同的随机存取延迟。然而，NM 的带宽被建模为高于 FM。在我们的基线系统中，NM 具有 4 倍带宽的 FM（4 倍通道）。第七节介绍了带宽比的灵敏度研究。

我们对虚拟内存系统进行建模，以执行虚拟到物理地址转换。我们假设页面大小为 4KB。我们研究三种系统配置：第一，一个操作系统可见的 NM 系统，不支持 NM 和 FM 之间的页面迁移（第四部分）;第二，支持 NM 和 FM 之间的页面迁移的 OS 可见的 NM 系统（第 V 节）;第三，其中 NM 被配置为硬件管理的 L4 高速

缓存（第 VI 节）的系统。对于第一个系统，默认页面映射策略是 NM-First，它首先在 NM 中分配页面，并且仅当它超出 NM 容量时，才分配 FM 中的页面。对于第二个系统，我们从随机页面映射开始;然而，在访问 FM 中的页面时，系统执行从 FM 到 NM 的页面迁移，从而优化数据局部性。对于第三个系统，我们将 NM 配置为具有高速缓存命中命中预测器的直接映射 Alloy Cache。为了确保高缓存性能，虚拟内存系统使用页面着色来减少直接映射缓存中的冲突未命中次数。

B 工作量

我们使用 Pin 和 SimPoints 从各种基准套件（包括 SPEC CPU2006,Stream 和高性能计算（HPC）工作负载）执行 10 亿条指令。表二显示了我们研究中使用的 20 个工作负载的特性。我们通过在速率模式下执行基准来评估我们的研究，其中所有 16 个内核执行相同的基准。由于我们的方案的有效性取决于应用程序工作集相对于 NM 容量的相对大小，我们将 SPEC 应用程序分为两类：工作集小于 256MB 的应用程序分为 SPEC S，其他为 SPEC L 。

TABLE II
WORKLOAD CHARACTERISTICS (FOR SINGLE CORE).

Category	Name	L3 MPKI	Footprint(MB)
HPC	hpc1	82	134
	hpc 2	48	81
	hpc 3	46	46
	hpc 4	43	132
	hpc 5	30	120
SPEC L	milc	65	428
	lbm	64	399
	Gems	53	727
	mcf	43	1186
	bwaves	39	423
SPEC S	soplex	64	54
	omnetpp	50	140
	libq	48	32
	leslie	43	78
	astar	25	37
STREAM	add	83	229
	triad	73	229
	copy	71	152
	scalar	64	152
	read	43	152

C 优点

1）加速：我们测量总执行时间作为品质因数。当我们以速率模式运行工作负载时，工作负载内单个基准的执行时间差异可以忽略不计。加速报告为基线的执行时间与给定方案的比率。

2）NM 访问速率:我们还报告 NM 的访问速率，定义为对 NM 的访问数除以对 NM 和 FM 的存储器访问的总数。每个数包括读和写。

IV.没有迁移的 BATMAN 系统

在本节中，我们考虑由 NM 和 FM 组成的系统，并且在 NM 和 FM 之间没有数据迁移（支持迁移的系统在后续章节中进行分析，对于这样的系统，确定 NM 的带宽利用的决定是页映射策略。例如，NM-Agnostic 策略可以随机地将页面分配给存储器中的任何位置。不幸的是，考虑到我们的 NM 比 FM 小 8 倍（4GB 比 32GB），这个策略将使用 NM 只有 11%（1/9）的页面，即使应用程序的工作集足够小 NM。一种替代策略，NM-First 可以通过首先在 NM 中分配页面，并且仅当 NM 的容量

耗尽然后在 FM 中分配页面来优化 NM 带宽。NM 首先尝试通过确保应用从 NM 尽可能多地接收页面来最大化 NM 的命中率。当应用工作集适合于 NM 时，该策略将确保所有的接入由 NM 服务，然而 FM 的带宽将保持未使用。

为了最大化系统带宽，我们可以调节分配给 NM 的页面数量，以确保只有目标分数的访问进入 NM。在我们的基线中，NM 具有 FM 的带宽的 4 倍，因此 NM 的目标接入速率（TAR）为 80%。理想情况下，我们希望 NM 只服务 80%的访问和 FM 服务 20%的访问。我们展示我们如何提出实施 BATMAN 可以有效地实现 80%的 NM 的 TAR。

A.BATMAN：工作集的随机分区

存储器访问可以在页面上显示出显著的不均匀性。因此，总页面的给定部分不总是导致从这些页面获得服务的所有存储器请求的相同部分的原因。例如，如果 90%的访问来自仅 10%的页面，则将 NM 的 90%的冷页面，给出 NM 的命中率仅为 10%。如果我们有页面访问计数可用，我们可以在 NM 中分配确切数量的页面，使得来自这些页面集合的总访问量等于 TAR。不幸的是，每页的访问计数通常不可用，并且在运行时获得它们往往是复杂的。

理想情况下，我们希望 BATMAN 有效，而不需要任何有关工作负载的准确访问模式的信息。BATMAN 通过一个很好的想法来实现这一点，就是如果我们在 NM 和 FM 之间随着每个存储器的带宽成比例地对工作集合进行分区，则对每个这样的分区的访问速率也可以预期与分配的数目成比例页面。当应用程序的工作集适合在 NM 中时，会很好地工作，因为我们具有决定页面是否在 NM 或 FM 中的余地。我们通过分析模型和广泛的实验评估来证明 BATMAN 在实现 TAR 时的有效性，即使 BATMAN 没有任何关于访问模式的明确信息。

B 随机分区的分析模型

让应用由 N 个页面组成，其中每页的访问的平均数量由 μ 表示，并且每页的访问数量的标准偏差由 σ 表示。令 CoV 表示每页访问的变化系数，定义为 σ 与 μ 的比率，并且用作在不同页面上的访问的变化性中的指示符。应用程序中的存储器访问的总数，我们表示为 AccessTot，由等式 1 给出。

$$Access_{Tot} = N\mu \qquad (1)$$

令 NM 中的目标接入速率为 α （在我们的情况下 α =0.8 英寸）。让每个页面以概率 α 随机分配到 NM，以概率（1-α ）随机分配到 FM。令由 NM 服务的接入的数目为 AccessNM。考虑到随机分配到 NM 的页面的数量很大（α •N），我们可以使用中心极限定理[24]将 AccessNM 近似为高斯分布，而不考虑原始工作负载的访问模式的确切分布。AccessNM 的期望值，其由 μ SumNM 表示，由等式 2 给出。AccessNM 的标准偏差，其由 σ SumNM 表示，由等式 3 给出。

$$\mu_{SumNM} = \alpha N\mu = \alpha \cdot Access_{Tot} \qquad (2)$$

$$\sigma_{SumNM} = (\sqrt{\alpha N}) \cdot \sigma \qquad (3)$$

访问 NM 总次数的变化系数，CoVSumN M 可以通过等式 4 确定：

$$CoV_{SumNM} = \frac{1}{\sqrt{\alpha N}} \cdot \frac{\sigma}{\mu} = \frac{1}{\sqrt{\alpha N}} \cdot CoV \qquad (4)$$

从公式 2，我们可以预计 NM 的命中率接近 α （假设 NM 足够大以容纳所有页面）。从等式 4，总接入的变化作为所有寻呼的数目的平方根下降，这意味着随

机化分配可以预期在到 NM 的接入速率中具有可忽略的小变化。例如，考虑假设的工作负载，其中所有访问都来自仅 1%的页面。对于该工作负载 σ 将是 μ 的约 10 倍（CoV 为 10）。如果我们从这个应用程序中抽取 100 万页，这个总和将增加一百万倍的平均值，然而，总和的标准偏差将增加原来的标准偏差的千倍。因此，总和的 CoV 将从 10 减少到 0.01。从置信区间理论，我们知道高斯随机变量可以期望在 95%置信度的平均值的 2 个标准偏差内。因此，考虑到总和的 CoV 仅为 1%，我们可以预期（以 95%的置信度），即使原始工作负载在每页访问中具有相当高的变化，对 NM 的访问速率仍保持在 TAR 的 2%内。

表 III 显示了我们研究的工作负载的不同特性，包括 N（页数），μ（每页的平均访问次数），σ（每次访问的标准偏差获得平均值，标准差，最大值和最小值从平均值和标准偏差中，我们根据实验数据计算 95%置信区间图 4 显示了我们的 1000 个实验的 95%置信区间范围，最大值和最小值，以及我们的实验的平均值对于所有工作负载，95%的置信区间，最大值和最小值都在目标接入速率的 80%的 2%内，因此提供了强有力的证据表明随机分区可以帮助 BATMAN 调节接入速率 NM 页面），以及从等式 4 导出的 NM 中的访问速率的 95%置信范围。对于该分析，我们假设 NM 85 足够大以容纳应用的所有页面（因为随机分区本身不能保证 80 如果 NM 没有足够的容量来提供 80%的命中率，则命中 NM 的命中率为 80%。在所有工作负载中，近存储器中的最大访问范围非常接近 75%，达到我们 80%的目标。这指示应用工作集的随机化划分在获取 NM 中的目标接入速率方面非常有效。

TABLE III
BOUNDS ON NM ACCESS RATE USING ANALYTICAL MODEL.

Name	N (# Pages)	μ (Mean)	σ (Stdv.)	95% Conf. Range (%)
hpc1	34475	2,949	3,246	78.9 -81.1
hpc2	20859	2,503	3,292	78.4 -81.6
hpc3	11812	4,500	7,200	77.4 -82.6
hpc4	33849	1,380	3,376	77.6 -82.4
hpc5	30899	1,161	1,579	78.6 -81.4
milc	109597	705	180	79.9 -80.1
lbm	102391	590	362	79.7 -80.3
Gems	186166	324	154	79.8 -80.2
mcf	303798	168	1,202	77.7 -82.3
bwaves	108314	484	313	79.6 -80.4
soplex	13855	5,375	9,030	77.4 -82.6
omnetpp	35954	1,689	1,698	79.1 -80.9
libq	8200	7,149	866	79.8 -80.2
leslie	19969	2,409	2,170	78.9 -81.1
astar	9625	3,459	3,453	78.2 -81.8
add	58634	1,606	516	79.8 -80.2
triad	758650	1,404	456	79.8 -80.2
copy	39105	1,990	588	79.7 -80.3
scale	39120	1,751	579	79.7 -80.3
read	39105	1,563	79	80.0 -80.0

C 随机分区的实验验证

我们还展示了使用实验评估在实现 TAR 时随机分配的有效性。我们进行了以下实验：在页面未命中，我们在 NM 中分配具有概率 α（在我们的研究中 α 等于 80%）的页面;否则，页面在 FM 中分配。我们设置 FM，以便它可以保存分配给它的所有页面，并且测量 FM 服务的访问的百分比。我们运行这个实验 1000 次（从不同的随机获得平均值，标准差，最大值和最小值从这 1000 个数据点）从平均

值和标准差，我们计算基于实验数据的 95%置信区间图 4 显示了 95%置信区间范围，我们的 1000 个实验的最大值和最小值，以及我们的实验的平均值，对于所有工作负载，95%置信区间，最大值和最小值均在 2%目标接入率为 80%，因此提供了强有力的证据表明随机分区可以帮助 BATMAN 规范对 NM 的接入速率。

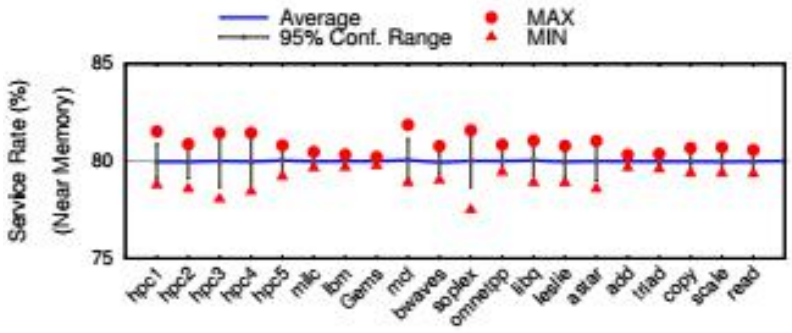
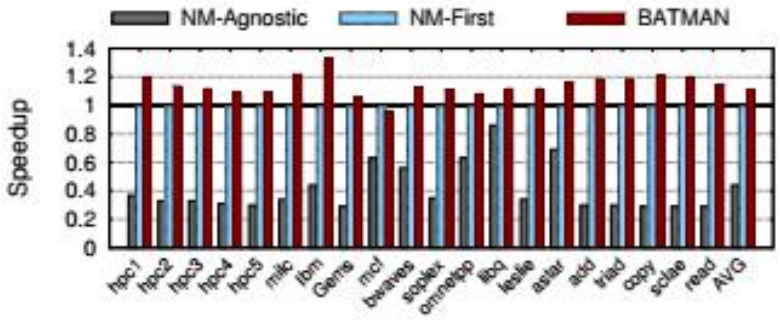


图 4　随机分区的实验验证。实线是平均值；误差条表示基于实验数据的 95%置信区间；点和三角形是在 1000 次试验中观察到的各自的最大值和最小值。

D.使用 BATMAN 的性能改进

图 5 示出了 BATMAN 和 NM-Agnostic 的加速，均被归一化为执行 NM-First 映射的基线系统。标记为 AVG 的条表示在所有 20 个工作负荷上测量的几何平均加速。由于 NM-Agnostic 可以从内存中的任意位置任意分配页面，因此即使工作负载完全适合 NM，它也只能在 NM 中分配 11%的应用程序页面（给定 NM 为 4GB，FM 为 32GB）。因此，它显着地在利用 NM 的带宽，并且因此与 NM-First 相比遭受几乎 60%的性能降级。



另一方面，NM-首先尝试最大化 NM 带宽，但是当工作集合适合 NM 时，漏掉 FM 带宽。BATMAN 使用随机分区来明确地控制到 NM 的访问速率到 TAR 值的 80%，因此它有效地使用 NM 和 FM 的带宽，这将平均性能提高了 10%。

E.在 NM 达到 TAR 时 BATMAN 的有效性

BATMAN 设计用于降低 NM 的命中率，如果它超过 TAR 值。因此，如果基准系统的 NM 的命中率超过 TAR 值，则可以期望在 NM 中成功地实现 TAR，正如当工作集适合于 NM 时的情况。如果 NM 不足够大去容纳开始的大部分工作集并且甚至对于基线系统在 NM 中实现小于 80%的命中率，则 BATMAN 将不能到达 TAR，因为 BATMAN 只能降低或保持 NM 的命中率，并且不被设计为增加 NM 的命中率。

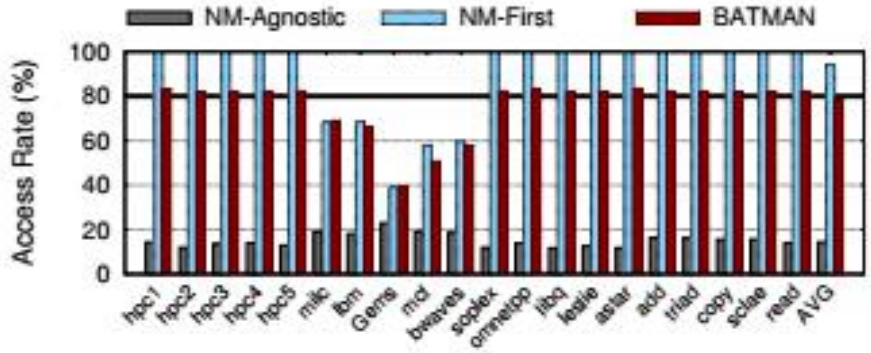


图 6.NM-First（基线系统），NM-Agnostic 和 BATMAN 的近存储器的访问速率。

图 6 显示了基线系统（NM-First），NM-Agnostic 和 BATMAN 在 NM 中的命中率。对于 NM，NM-Agnostic 具有小于 15％的平均命中率，因为它仅将所有页面的一小部分（11％）分配给 NM。除了 SPEC L 类别（milc-bwaves）之外，NM-First 对于所有工作负载的 NM 具有几乎 100％的命中率。对于这些应用，工作集远远大于 NM 的容量，因此命中率远低于 70％。

BATMAN 对所有工作负载有效达到 80％的 TAR，基线的命中率超过 80％。对于剩余的工作负载，BATMAN 保留类似于基准系统的命中率。总的来说，我们的结论是，如果 NM 的基线命中率超过 TAR 值，这是系统正在利用 FM 的指标，并且是我们试图通过 BATMAN 解决的关键低效率，BATMAN 实现 TAR。

Ⅴ.BATMAN 系统与页面转移

对于不支持页面迁移的系统，一旦页面映射到特定类型的内存区域，映射将继续，直到页面从内存中逐出。因此，这种系统的性能往往对页面的初始映射非常敏感，例如 NM-First 或 NM-Agnostic。允许页面在 NM 和 FM 之间动态迁移可以减轻系统对页面初始放置的敏感性。例如，对 FM 的访问可以发起所访问的页面到 NM 的传送，这可以允许 NM 对该页面的后续引用。这样的系统利用参考流中的时间和空间局部性以最大化 NM 中的命中率。当应用工作集小于 NM 的容量时，这样的迁移将最终将整个工作集移动到 NM，并且因此导致几乎所有的访问由 NM 服务。不幸的是，这样做将使 FM 的带宽未被利用。理想地，我们想要进行页面迁移，并且还通过确保所有接入的目标接入速率（TAR）部分不被 NM 服务来确保 NM 和 FM 的带宽得到利用。我们展示如何为支持页面迁移以满足 TAR 的系统设计 BATMAN。

A.想法：规范移民方向以满足 TAR

我们观察到 NM 的命中率取决于页面迁移的能力。将页面从 FM 迁移到 NM 会增加 NM 的命中率。类似地，将最近访问的页面从 NM 降级到 FM 降低了 NM 的命中率。因此，BATMAN 可以简单地通过监视由 NM 服务的访问的数量并使用该信息来确定页面迁移的方向来调节命中率。如果在 NM 中测量的访问速率低于 TAR，BATMAN 应该尝试将访问的页面从 FM 升级到 NM。如果在 NM 中测量的访问速率高于 TAR，BATMAN 可以将一些最近访问的页面从 NM 降级到 FM。这可以允许 BATMAN 确保 NM 的访问速率保持接近 TAR。

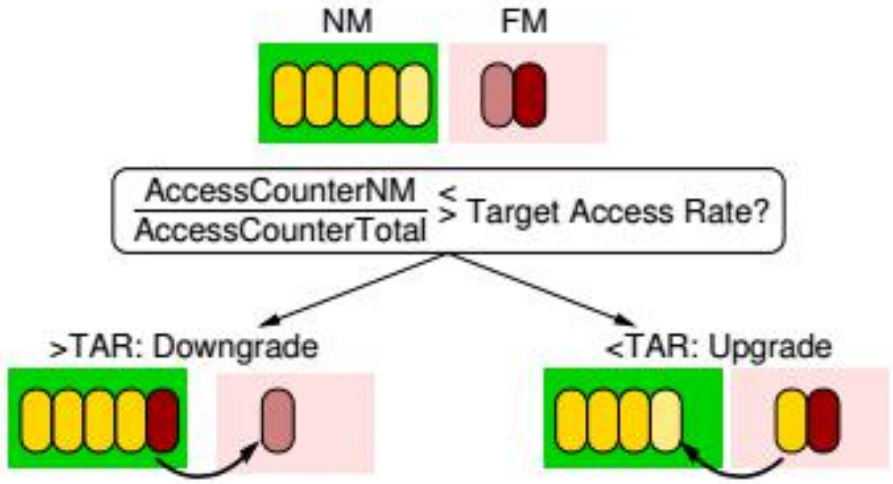


Fig. 7. Overview of BATMAN: monitoring the access rate of NM for determining the direction of migration to meet TAR in NM.

B.BATMAN：设计和实施

图 7 显示了 BATMAN 如何动态控制页面迁移方向以满足 TAR 的概述。我们的系统具有 4x 带宽的 NM，因此 TAR 为 80％。 我们提供系统来监视 NM 的命中率并且进行升级和降级决定，如下所述：

1）结构:BATMAN 需要动态地监视由 NM 服务的总内存访问的比例。BATMAN 通过使用两个计数器来完成此操作：AccesssCounterNM 和 AccessCounterTotal，它们分别计算对 NM 和系统（NM 和 FM）的访问次数。AccessCounterNM 与 Access-CounterTotal 的比率是由 NM 服务的总请求的分数的指示符。这两个计数器根据需要，预取或写回请求增加。我们对两个计数器使用一个 16 位寄存器，当 AccessCounterTotal 溢出时，我们将两个计数器减半（右移一位）。

2）操作:在每次访问时，我们计算 AccesCounterNM 与 AccessCounterTotal 的比率。如果这个比率小于 TAR，我们想增加 NM 的命中率，因此如果请求是 FM，我们将请求的页面从 FM 升级到 NM。类似地，如果比率大于 TAR，则我们想要减小 FM 的命中率。因此，如果请求是在 NM 中的页面，我们将请求的页面从 NM 降级到 FM。基于运行时信息调节此类降级和升级可确保 FM 的访问速率接近 TAR。

3）滞后在阈值:一旦 NM 的访问速率接近 TAR，系统可以在升级和降级之间连续切换。为了避免这种振荡行为，我们在升级和降级的决策中在任一方向上提供 2％的保护带。因此，页面升级仅在 NM 的测量访问速率小于（TAR-2％）时发生，而降级仅在 NM 的测量访问速率超过（TAR + 2％）时发生。在中间区域中，既不执行升级也不执行降级，因为到 NM 的访问速率已经非常接近 TAR。

4）存储开销:提议的 BATMAN 实现需要仅四个字节（两个 16 位计数器）的存储开销来跟踪对 NM 和系统的访问。BATMAN 利用现有的 NM 和 FM 之间的页面迁移支持，并且需要对数据迁移逻辑进行可忽略的修改。

C.性能改进从 BATMAN

图 8 显示了与在 NM 和 FM 之间执行页面迁移的基准系统相比，BATMAN 的加速。BATMAN 提高了所有工作负载的性能，平均提高了 10％。性能改善来自两个因素。首先，对于工作集适合近内存的工作负载，BATMAN 有效地使用 NM 和 FM 带宽，因此具有更高的这些工作负载的吞吐量。例如，HPC 和 STREAM 中的所有

工作负载以及来自 SPEC S 套件的一些工作负载都属于此组，并且具有更高的性能。其次，对于工作集大于近内存的 SPEC L 类别（例如 bwaves，mcf，Gems 和 milc）的工作负载，BATMAN 减少了页面迁移的数量，因为 BATMAN 只允许特定数量的页面升级到近内存。由于页面迁移消耗了大量的内存带宽，BATMAN 有助于降低这组工作负载的带宽利用率，并有助于获得更高的性能。

D.在 NM 达到 TAR 时 BATMAN 的有效性

图 9 显示了页面迁移的基准系统和 BATMAN 的命中率 NM。在基线中，即使对于工作集大于 NM 容量的 SPEC L 基准套件，所有工作负载也具有接近 100％的命中率。这是因为这些工作负载在页面内具有高空间局部性。例如，对 FM 的访问将页面升级到 NM，然后 15 个更多的引用可以去往页面中的不同行，导致 NM 的命中率为 15/16，或接近 94％.1 对于其他应用 ，因为它们的工作集适合于 NM，所有页面被传送到 NM 并且在稳定状态下获得 100％的 NM 命中率。

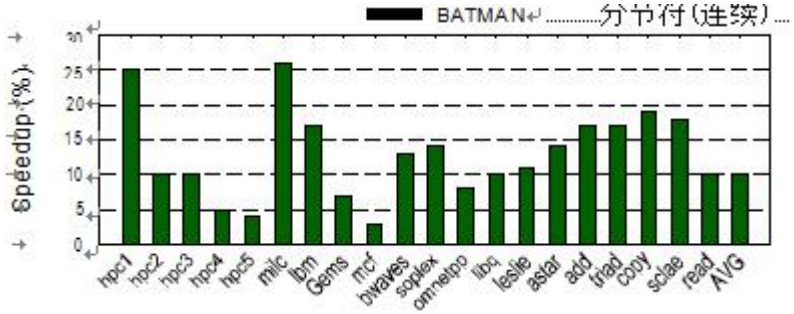


图 8.对于具有页面迁移的系统，BATMAN 的加速。

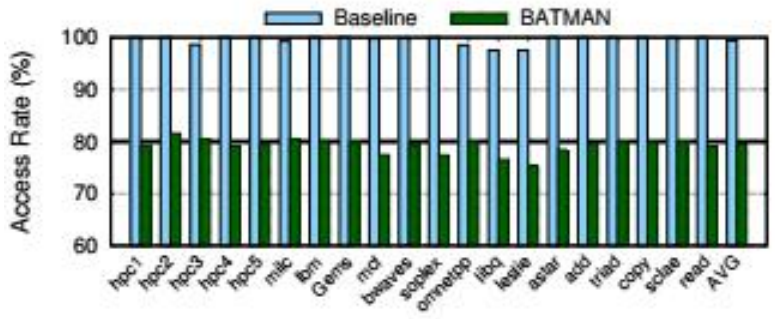


图 9.近存储器的存取速率。BATMAN 获得接近所有工作负载的目标访问速率 NM（80％）的访问速率。

BATMAN 将一些内存流量重新分配给 FM，并根据 NM 和 FM 的带宽比平衡系统。对于所有工作负载，BATMAN 有效地获得接近 80％目标值的 NM 命中率，这有助于 BATMAN 平衡内存流量，其中 80％从 NM 处理，20％从 FM 处理。因此，我们提出的 BATMAN 的实现在通过在运行时使用简单的监视电路在 NM 获得 TAR 是有效的。

VI.BATMAN 系统与 DRAM 缓存

NM 还可以被构造为硬件管理（L4）高速缓存，其可以服务于处理器的最后级高速缓存（LLC）和 FM 之间的中间高速缓存。使用 NM 作为硬件管理的缓存允许以软件透明的方式容易地部署。与 OS 可见的近存储器类似，当大多数存取由 DRAM 高速缓存服务时，DRAM 高速缓存也无效率地利用总有效存储器系统带宽。例如，当应用程序工作集合适合 DRAM 缓存时，所有 FM 带宽保持未使用，因为所有访问仅由 DRAM 缓存提供服务。 即使对于这样的系统，存在通过动态调节由 NM-Cache 服务的工作集的分数来最大化总可用系统带宽的机会。

A.想法：通过部分缓存禁用调节命中率

BATMAN 对于 DRAM 缓存的核心思想是通过控制 DRAM 缓存命中率来调节从 DRAM 缓存服务的工作集的分数。例如，100%DRAM 缓存命中率使 FM 带宽不可用。相反，我们的基准系统中 80%的 DRAM 缓存命中率可以通过为来自 FM 的 20%的请求提供服务来有效利用总系统带宽。 因此，用于 DRAM 高速缓存的 BATMAN 必须调节 DRAM 高速缓存访问速率以提高整体系统性能。调节 DRAM 缓存访问速率可以通过动态禁用DRAM缓存的一部分直到DRAM缓存访问速率匹配期望的目标访问速率来实现。BATMAN 支持通过预选 DRAM 缓存集的子集以仅服务来自 FM 的数据来禁用 DRAM 缓存集。禁用集合减少了执行标签查找的开销，以确定行是否存在于缓存中。

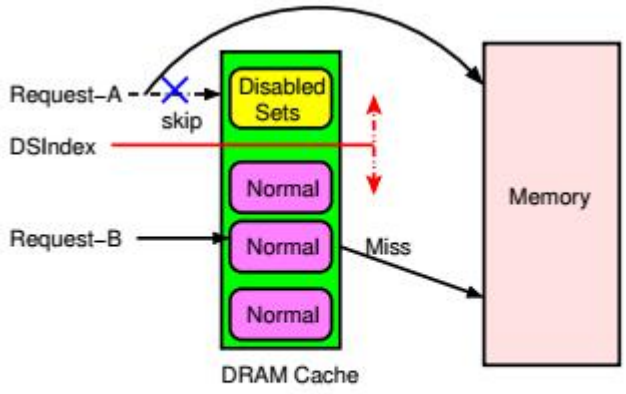


Fig. 10. BATMAN for DRAM caches. All sets at index lower than DSIndex are deemed "Disabled Sets" and do not service any cache request. DSIndex moves to modulate the hit-rate closer to the target.

B.用于 DRAM 缓存的 BATMAN 的设计

图 10 显示了具有硬件高速缓存的 BATMAN 的概述。此设计的关键属性是禁用集索引（DSIndex），它控制禁用的高速缓存集的分数。通过移动 DSIndex，我们可以控制保持启用的高速缓存集的分数，从而控制高速缓存的命中率。 BATMAN 通过动态监视缓存命中率并将其与目标访问率（TAR，本例中为 80%）进行比较来调节 DSIndex 的移动。我们提供系统来监视高速缓存的命中率并增加和减少 DSIndex，如下所述：

1）结构:BATMAN 使用两个 16 位计数器来监视 DRAM 缓存的访问速率：AccessCounterCache 和 AccessCounterTotal。AccessCounterCache 用于跟踪高速缓存访　　问（例如读取，回写，探测等）的总数，而 AccessCounterTotal 用于跟踪对DRAM高速缓存和内存的总访问次数。当AccessCounterTotal溢出时，我们将两个计数器减半（右移一位）。

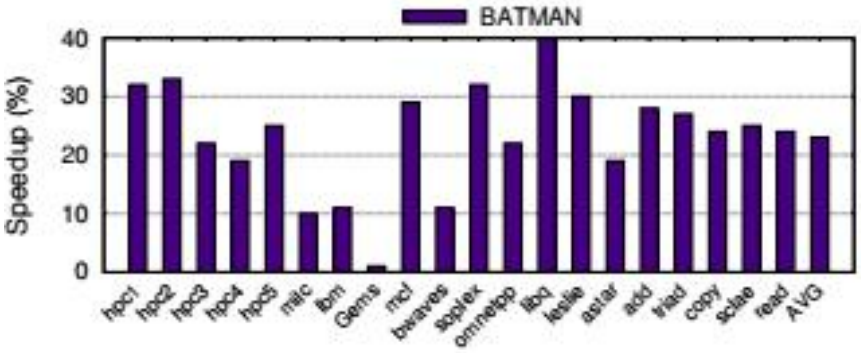
2）操作:如果请求映射到 DSIndex（图 10 中的请求 B）之外的集合索引，则它以正常方式被服务 - 查找缓存，如果找到该行，则返回它，如果没有获得线从内存并安装在集合中。如果请求将设置的索引映射为小于 DSIndex（图 10 中的请求-A），则请求直接去往存储器，而不需要查找高速缓存。

3）调节 DSIndex 和滞后:BATMAN

（通过计算 AccessCounterCache 与 AccessCounterTotal 的比率）来确定 DRAM 缓存访问是否超过 TAR。如果比率超过 TAR，则 BATMAN 增加 DSIndex，直到两个计数器的比率在期望的 TAR 的 2%保护带内。如果观察的比率低于 TAR，

BATMAN 减小 DSIndex，直到两个计数器的比率在 2%保护带内。在增加 DSIndex 之前，BATMAN 首先使 DSIndex 引用的集无效，并确保将脏行写回 FM.

4）存储开销:建议的 BATMAN 实现需要只有 8 个字节的总存储开销：两个 16 位计数器和一个 32 位 DSIndex。 BATMAN 利用现有硬件进行缓存管理，并且对缓存访问逻辑的修改最少。



C.从 BATMAN 改进性能

图 11 显示了 BATMAN 的加速。BATMAN 将所有工作负载的性能平均提高了 23%。用于高速缓存的 BATMAN 以两种方式改进带宽：第一，使 FM 的带宽可用，第二，遗漏在基线中将被丢失的禁用的集合现在将避免 DRAM 缓存的标签查找带宽。这意味着，来自 BATMAN 的整体带宽优势超过仅仅由 FM 提供的带宽。这解释了为什么多个工作负载的性能超过 25%。对于工作集大于 NM 的应用程序,BATMAN 能够通过启用所有缓存集来调节访问速率，因此性能改进很小。

D.使用 BATMAN 的硬件缓存的访问速率

由于 BATMAN 绕过了一部分内存访问的高速缓存查找，而不是报告高速缓存命中率，我们只需报告 NM 高速缓存（而不是 FM）满足的内存请求的百分比（来自 L3）。图 12 显示了我们的基准系统和 BATMAN 的 NM 的访问速率。在基线，NM 的访问率平均为 85%，几个工作负载超过 90%。使用 BATMAN，访问速率降低到 80%的目标访问速率。平均来说，BATMAN 平均有 79%的 NM 的接入速率，因此我们提出的设计在控制 NM 的接入速率方面是相当有效的。

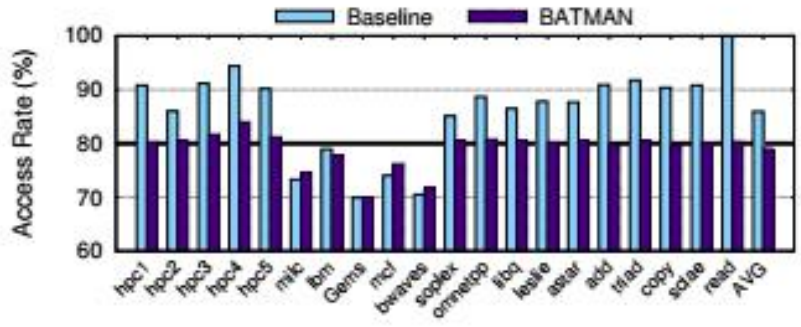


Fig. 12. Access Rate for the baseline and BATMAN.

VII.RESULTS 和分析

A.带宽改进

我们的目标是最大化系统中的带宽利用率。 我们通过明确测量每个配置中

使用的带宽，展示 BATMAN 在最大化内存系统带宽方面的有效性。我们通过监视内存通道总线的繁忙时间来测量内存带宽，并报告每个基准套件的平均带宽改进。图 13 显示了 BATMAN 在三种配置中的带宽改进：无迁移（无迁移），页迁移（页迁移）和缓存线迁移（Cache）的系统。我们观察到，BATMAN 平均将内存带宽利用率分别提高了 11％，10％和 12％。最大带宽增加始终是从 17-19％的 STREAM 基准。注意，改进数字是相对于每个配置中的基线。

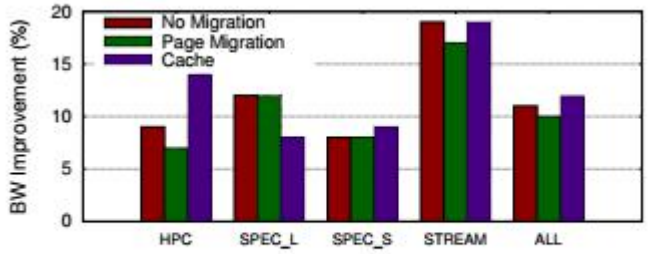


Fig. 13. System Bandwidth Improvement. No Migration, Page Migration and Cache are with respect to respective baselines

B.带宽比的灵敏度研究

在我们的默认系统中，我们假设近存储器具有 4 倍带宽的远存储器。我们还进行灵敏度研究，以将近存储器和远存储器带宽比率从 2 倍改变为 8 倍。结果如图 14 所示。当远存储器是近存储器的半带宽时，BATMAN 能够将性能提高 30％以上，因为聚合带宽是近存储器带宽的 1.5 倍，从而大大提高了可用带宽。当比率为 8X 时，使用远存储器可以将总带宽提高 12.5％，这意味着性能提高了 5-9％。

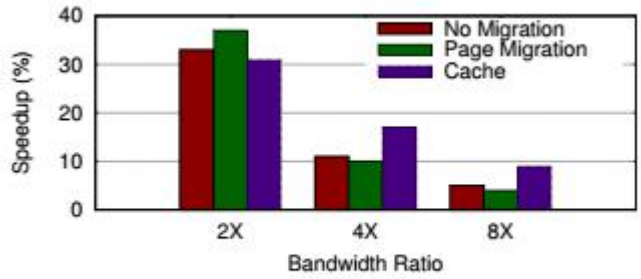


Fig. 14. Sensitivity Study of Different Bandwidth Ratio

C.跨越一大组工作负载的研究

由于空间限制，我们仅显示 20 个工作负载的详细结果;然而，我们对更大的一组 35 个工作负载进行研究。我们使用 s-curve 来显示我们在图 15 中使用的所有 35 个工作负载的性能改进。请注意，所有配置都归一化为各自的基准配置，我们通过性能改进的值对基准进行排序。平均而言，BATMAN 将性能提高 11％，在没有迁移的系统中最多可提高 33％，在带有页面迁移的系统中最多可提高 26％，在配置缓存的系统中最多可提高 17％（最多 40％）。注意，对于具有 35 个工作负载（总共 105 个数据点）的三组实验，我们仅观察到一种性能降级的情况（对于没有数据迁移的系统，mcf 为 4％的减速）。因此，我们提出的方案不仅是高性能，而且在工作负载和系统上也是稳健的。

VIII 相关工作

A.DRAM 缓存

几个最近的研究已经研究使用高带宽 DRAM 作为高速缓存。大多数工作尝试提高缓存命中率，并且不知道 FM 提供的额外带宽。因此，这些方案不会使整个系统带宽最大化。

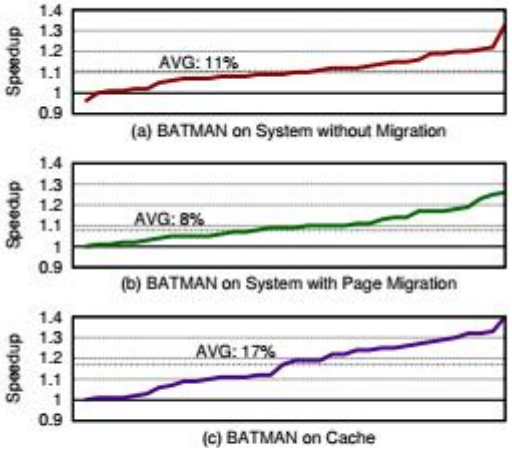


Fig. 15. Performance Improvement of BATMAN over a larger set of 35 workloads for the 3 systems. Note the performance shown on the S-curve is normalized with respect to the respective baseline.

考虑使用远存储器带宽的先前工作之一是 Mostly-Clean DRAM cache [10]，其使用未命中预测器与高速缓存并行地访问远存储器，以便缓解高速缓存具有大量的争用。然而，对 NM 和 FM 的并行访问增加了存储器业务，并且消耗更多的带宽（每个请求两次访问）。因此，聚合存储器系统带宽不增加。相比之下，BATMAN 尝试除了 NM 带宽之外还使用 FM 带宽，并且因此不增加总体存储器通信量。

B. Cache 优化

已经投入了大量的研究工作用于高速缓存优化。在管芯上缓存的水平上，传统的做法是提高命中率，因为高命中率通常与提高性能相关。因此，大多数传统方法试图提高缓存命中率。然而，我们的建议在 DRAM 缓存或混合存储器系统级别上优化缓存命中率，以便最大化存储器系统中的总带宽。我们表明，当约束不同时，如果缓存命中率高于所需的值，减少缓存命中率可能是有意义的。

C. 软件优化

在过去几十年中已经提出了几种页面迁移策略来改善非一致性存储器架构（NUMA）的性能，其中本地节点在等待时间方面比远程节点快一个数量级。该工作主要集中在如何提高共享内存模型中的系统性能。例如，减少共享内存架构中的乒乓效应，这两个节点访问同一个页面。

其他软件优化包括高速缓存阻塞算法，NUMA 环境中的用户级线程调度被提出以优化延迟减少。然而，在具有与 FM 相比具有相似延迟的高带宽存储器的混合存储器系统的情况下，我们示出这些提议可能需要重新访问。

IX. 总结

诸如 HBM，HMC 和 WIO 的新兴存储器技术在类似的随机存取延迟方面提供比传统存储器高 4-8 倍的带宽。这些技术用于分层存储器系统，其中近存储器（NM）通常由低容量高带宽存储器构成，而远存储器（FM）通常由高容量低带宽传统存储器构成。分层内存系统专注于通过在 NM 和 FM 之间迁移数据来服务来自 NM 的

大多数请求。我们显示这种方法浪费 FM 带宽，特别是当应用工作集适合 NM。在这种情况下，可以通过在 NM 和 FM 之间拆分访问来提高性能。

本文表明，在 NM 和 FM 之间的接入分离与各个存储器系统提供的相对带宽成正比。我们建议在三种不同类型的系统中动态优化目标访问速率的带宽感知分层内存管理（BATMAN）（a）将 NM 配置为 OS 可见内存并静态映射 NM 和 FM 之间的页面的系统（b）将 NM 配置为 OS 可见但是在 NM 和 FM 之间动态迁移页面的系统（c）将 NM 配置为硬件管理的 DRAM 缓存的系统。我们证明 BATMAN 的原则适用于所有三个系统，我们提出的实施既简单又高效。我们对具有 4GB NM 和 32GB FM 的 16 核系统的研究表明，BATMAN 平均根据配置将性能提高多达 40％和 10-22％，同时需要少于 12 字节的硬件开销。

随着新的内存技术的出现和一系列新的约束的发展，我们表明重新思考内存层次结构的管理可以产生显着的效益。虽然我们评估 BATMAN 用于仅由堆叠的 DRAM 和商用 DRAM 组成的混合存储器系统，但是洞察和解决方案也适用于其他混合存储器系统。探索这样的扩展是我们未来工作的一部分。