華中科技大學

研究生高级计算机体系结构课程论文（报告）

题目 HybridTier: an Adaptive and Lightweight CXL-Memory Tiering System

院　　系　计算机科学与技术

专业班级　硕 2502 班

姓　　名　崔皓奕

学　　号　M202574020

指导教师　曹强

2025 年 12 月 31 日

目 录

1 论文写作大纲

1.1 摘要

现代工作负载对内存容量的需求日益增长。基于 Compute Express Link（CXL）的内存分层技术已成为解决此问题的有前景的方案，其通过将传统 DRAM 与慢速层级的 CXL 内存设备结合使用。通过分析先前的分层系统，观察到高性能内存分层面临的两大挑战：在适应倾斜但动态变化的数据热度分布的同时，将分层导致的内存和缓存开销最小化。为应对这些挑战，我们提出了 HybridTier——适用于 CXL 内存的自适应轻量级分层系统。HybridTier 同时追踪数据的长期访问频率与短期访问动量，以准确捕捉并适应不断变化的热度分布。HybridTier 通过概率性地追踪数据访问来降低元数据内存开销，以较小的追踪精度损失换取更高的内存效率，而该精度损失对应用性能的影响可忽略不计。为减少缓存开销，HybridTier 采用优化数据局部性的轻量级数据结构来追踪数据热度。我们的评估表明，HybridTier 的性能比以往系统提升高达 91%（几何平均值 19%），内存开销降低 2.0–7.8 倍，缓存未命中降低 1.7–3.5 倍。

1.2 引言

现代应用对内存容量和带宽的需求日益增加。单个服务器内存容量和内存成本都成为限制，近十年 DRAM 密度的发展也相对缓慢。以 CXL 为基础的分层内存架构是解决上述问题的可靠方案。更大容量更低价格的 CXL 内存与本地 DRAM 相结合，能耗更低使用效率更高。另一方面，CXL 内存的延迟和带宽表现不如本地 DRAM，所以要将高速层级的 DRAM 和低速层级的 CXL Memory 巧妙结合。

然而，实现高性能分层颇具挑战。我们对大内存工作负载进行了分析，并得出两个观察结果：（1）实际工作负载通常表现出倾斜但动态变化的数据热度分布；（2）管理数据访问统计信息可能带来显著开销。理想的分层系统应满足三个要求：（1）通过将最热数据放置在快速层级内存中，准确捕获热数据集合；（2）快速适应热度分布的变化；（3）最小化分层元数据开销。

然而，现有系统无法满足全部三个要求。一类工作采用基于频率的分层，通过存储每个页面的访问计数来构建热度直方图。基于此直方图，分层系统将最热页面置于快速层级，满足要求 1。另一类工作是基于近期性的分层，它使用访问新近性来近似数据热度。此类系统使用连续缺页之间的时间等指标来做出数据放置决策。基于近期性的系统满足要求 3，因为与频率对应项相比开销更小。然而，基于近期性的系统无法满足要求 1 也不满足要求 2。

在本工作中，提出 HybridTier，一种应用透明的分层系统，它满足了这三个要求。HybridTier 为每个页面维护两个指标，以同时捕获长期访问历史和短期热度变化。使用计数布隆过滤器（CBF）追踪内存访问，这是一种概率性数据结构，以较高的内存效率换取较低的追踪准确性。

本文做出以下贡献：

- 分析了现有分层系统，揭示了三个新发现：1）适应变化的热度具有挑战性，导致在真实工作负载下性能次优；2）维护历史访问信息可能带来高元数据内存开销；3）分层系统可能因元数据更新期间数据局部性差而遭受大量缓存未命中；
- 介绍 HybridTier，一种应用透明的内存分层系统，具有自适应性和轻量级特性。采用新颖的访问追踪方法，同时捕获长期热度分布和短期数据热度变化。同时，通过采用概率性访问追踪和局部性优化的数据结构，显著降低了元数据内存消耗和缓存未命中；
- 比较了 HybridTier 在六种大内存工作负载上的性能，同时改变快速到慢速层级内存比例。HybridTier 的性能平均优于先前工作 19%，同时因分层产生的内存开销降低 2.0−-7.8 倍，缓存未命中降低 1.7–3.5 倍。

1.3 背景和动机

1.3.1 CXL-Enabled Memory Tiering Systems

CXL 技术定位：Compute Express Link (CXL) 是运行在 PCIe 物理层上的开放行业标准互连协议，旨在支持未来数据中心的异构计算和内存能力。

核心特性与权衡：

- 优势：CXL 连接的内存（如 DDR4/5）具有字节可寻址性，可直接被 CPU 访问，支持标准内存分配接口；相比本地 DRAM，容量更大、成本/GB 更低（CXL 总线功耗更低、形态更紧凑）
- 劣势：延迟高 50-100ns，带宽仅为本地 DRAM 的 20%-70% 分层目标：精准识别热/冷数据，将热数据置于本地 DRAM（快速层），冷数据置于 CXL 内存（慢速层）

1.3.2 Dynamic Data Hotness Distribution

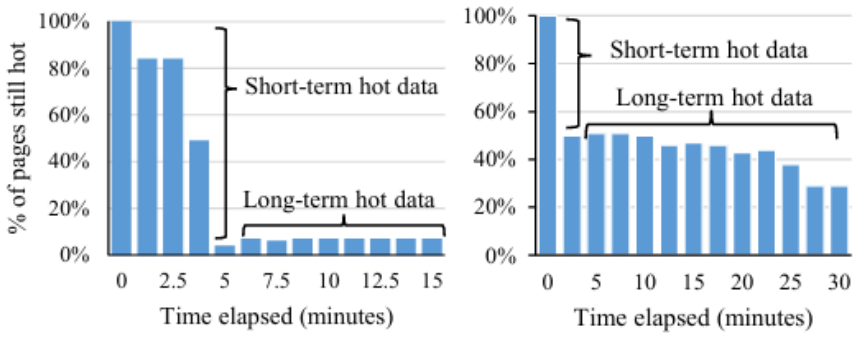
通过分析 Meta、Twitter 等生产环境，揭示大型内存工作负载的两大核心特征：

(1) 访问倾斜性（Skewed Distribution）

- 内存缓存普遍遵循 Zipf 或幂律分布，80% 访问集中在最热门的 10% 数据
- Meta 对象存储缓存的实证数据验证了此规律

(2) 动态变化性（Dynamic Variation）

- 时间尺度极短：热数据可在几分钟内变冷。Meta 报告显示，50% 热门对象在 10 分钟后不再热门
- 空间影响显著：图 21-1显示，在 PageRank 和 XGBoost 中，超过 90% 和 50% 的初始热页面在 5 分钟后失去热度
- TTL 机制：Twitter 生产缓存普遍采用分钟级生存时间，对象过期后即从缓存移除



(a) Graph analytic: Page Rank. **(b)** Machine learning: XGBoost.

图 1-1 负载热度分布变化

1.3.3 Prior Tiering Systems

本节批判性分析现有系统在三大核心要求上的不足。提出理想分层系统的三大要求：（1）准确捕获热数据：将最热数据精准放入快速层；（2）快速适应热度变化：及时识别新晋热/冷页面；（3）最小化元数据开销：降低内存和缓存资源消耗。

(1) 确捕获热数据（Requirement 1）：现有频率类系统（如 Memtis）通过维护访问频率直方图，基本满足此要求。

(2) 适应变化的热度分布（Requirement 2）

- 基于频率的系统（如 Memtis）无法快速适应热度变化，需等待访问计数更新。EMA 是滞后指标，图1-2显示某页面停止访问后，EMA 得分需 9 分钟才降至阈值以下; 减小冷却周期 C 可提升适应性，但会破坏热度分布准确性。

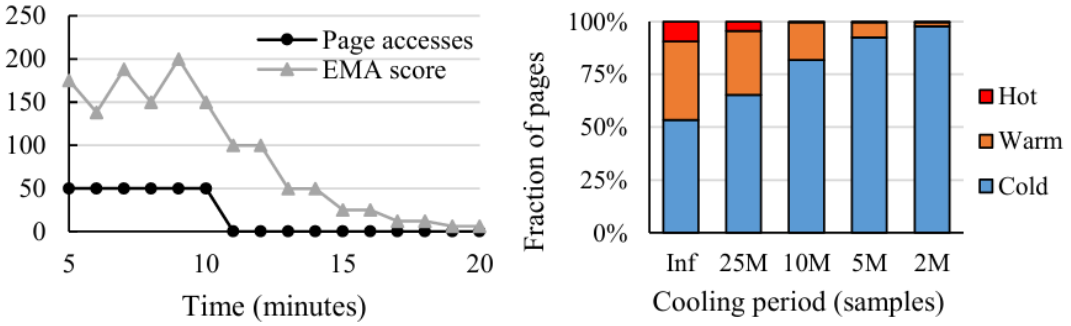


图 1-2　基于频率的系统适应热度分布变化

- 基于近期性的系统（如 AutoNUMA、TPP）准确性缺失, 仅考虑短时窗口，易将冷页面误判为热页面通过时间间隔等指标近似热度，但无法捕获长期访问模式, 如图1-3。

(3) 分层元数据开销（Requirement 3）

- 内存开销：每个 4KB 页面分配 16B 元数据，1TB 服务器产生 3.9GB 开销，是 Linux 内核启动内存的近 10 倍;10 万台 AWS t2.nano 实例的 100GB 额外内存开销，每年浪费 3180 万美元;
- 缓存开销：硬件计数器采样需频繁更新元数据，导致大量缓存未命中; 图 5 显示 Memtis 的分层活动平均消耗 L1 缓存未命中的 9% 和 LLC 未

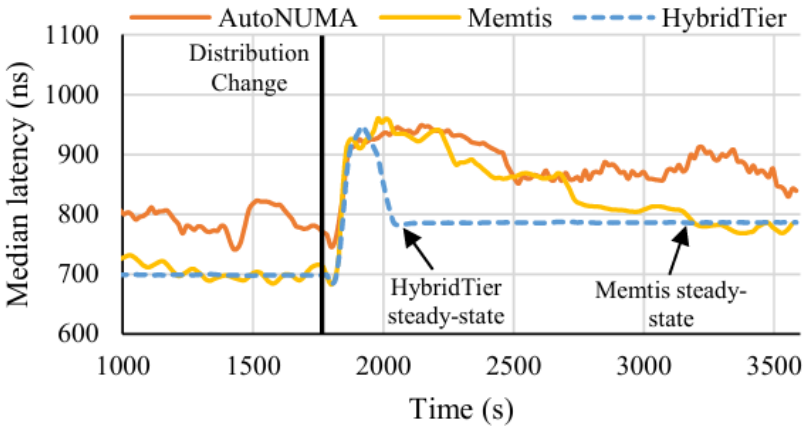


图 1-3　基于新近度的系统适应热度分布变化

命中的 18%（常规页面）

1.4　混合层级关键思想

本章提出 HybridTier 的三个关键创新，分别针对性解决第二章指出的三大缺陷。

适应动态热度分布，为每个页面同时维护两个独立指标——频率（长期访问历史，分钟到小时级）和动量（短期访问强度，秒级）。晋升时，只要频率或动量任一指标高即提升页面，快速响应新晋热页；降级时采用”二次机会”策略，避免历史热页因短暂冷却被错误淘汰。在 CacheLib 负载中，适应新分布的速度比 Memtis 快 3.2 倍。

降低元数据内存开销，用概率性数据结构替代精确数据结构——采用 Counting Bloom Filter (CBF) 追踪访问计数。每个计数器仅 4 位（最大计数值 15），因为访问 ≥15 次的页面都应置于快速层，无需精确区分。内存开销比精确哈希表降低 4.0–7.8 倍（128MB CBF 即可覆盖 50 万页面，误判率 <0.1%）。

减少缓存开销，采用 Blocked CBF，强制同一页面的 k 个计数器位于同一 64B 缓存行，确保每次查询最多 1 次缓存未命中。Memtis 每页 16B 元数据 → 每缓存行仅存 4 页；HybridTier 每缓存行存 32 页，且单级结构避免指针解引用。Tiering 活动的缓存未命中减少 1.7–3.5 倍。

1.5 HybridTier 混合层级设计

1.5.1 HybridTier 工作流

HybridTier 被部署在一个单独的用户线程中，不需要应用修改或内核补丁。工作流如下：

(1) HybridTier 动态透明地连接到运行的应用中，不需要重新编译应用；

(2) Intel PEBS 或 AMD IBS 以 ≈100 kHz 频率产生硬件事件，每条事件包含虚拟地址和本地 DRAM or CXL 标识；

(3) 更新时用双 CBF 计数器追踪访问频率和动量，跟踪误差控制：p=0.001，4-bit 计数器最大 15，超过不再细分（热页足够热即可）；

(4) 根据记录做出迁移决策；

(5) 最后从两层内存中发起迁移调动。

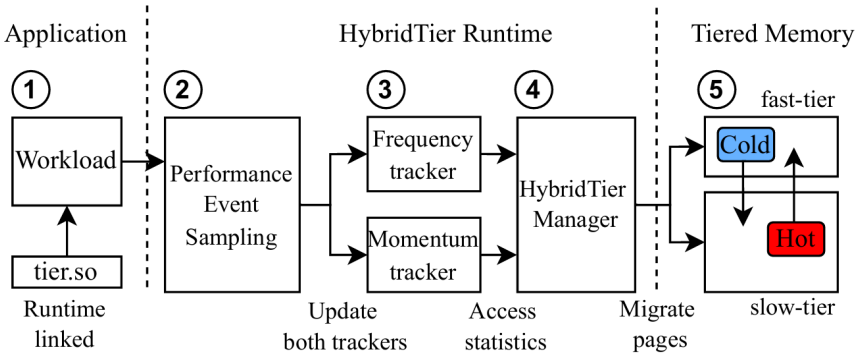


图 1-4 HybridTier 工作流

1.5.2 计数布隆过滤器设计

在 HybridTier 里，Frequency-CBF 与 Momentum-CBF 的“计数器数目”是同一份公式算出来的理论值，但实际分配的位数组大小可以按需截断，因为两条 CBF 的冷却速度不同导致有效页数不同。实际部署时动量 CBF 可截断 128×，因为秒级冷却让它只追踪当前瞬时热点，而频率 CBF 要保留全部历史热度。

表 1-1 Counting Bloom Filter (CBF) 关键参数与结果汇总

项目	数值 / 公式
哈希函数数 k	4
计数器位宽	4 bit (最大 15)
目标误判率 p	0.001
每页计数器期望 r	$-\frac{k}{\ln\left(1-e^{\ln p/k}\right)} \approx 9.6$
位数组大小 m	$\lceil n \cdot r \rceil$ (n= 快层页数)
64 B 缓存行槽数	128 × 4-bit
查询 LLC miss	≤ 1（Blocked-CBF）
实测决策正确率	99.6 %（64 MB CBF）
Frequency-CBF 内存	≈ 4.8 MB（1 M 页）
Momentum-CBF 内存	≈ 0.037 MB（比频率计数器少 128 x ）

1.5.3 页面迁移策略

- 晋升（Promotion）每采一页 → 双 CBF 计数更新 → 按 “频率/动量” 矩阵即时判断；10 万样本批处理后一次性系统调用批量晋升。
- 降级（Demotion）快层空闲低于水位时，线性扫描页表，立即降 “双低” 页；对 “高频率 + 低动量” 页打 Second-Chance 标签，1 分钟后复查未再升温即降。

1.5.4 大粒度页面支持

启用 THP 后，HybridTier 以 2 MB 粒度追踪/迁移，把 CBF 计数器升至 16 bit 防止溢出，同时元素总数 ↓512×，最终元数据内存再省 128×，性能与 4 KB 页持平。

1.6 实验设计

1.6.1 CXL 设备部署

CXL 真实设备稀缺，所以用双路 NUMA 节点模拟 CXL，实测延迟 124 ns、带宽 34 GB/s（≈ CXL 1.1 延迟）16 核 Xeon 4314 × 2，每节点 512 GB DDR4。

1.6.2 对比方案和实验配置

表 1-2 对比基线一览

系统	算法类型
AutoNUMA	新近度
TPP	新近度
Memtis	频率
ARC	混合 LRU | 自调参双链表，缓存策略 | TwoQ | 混合 LRU | 两次队列区分“仅一次”

1.6.3 工作负载

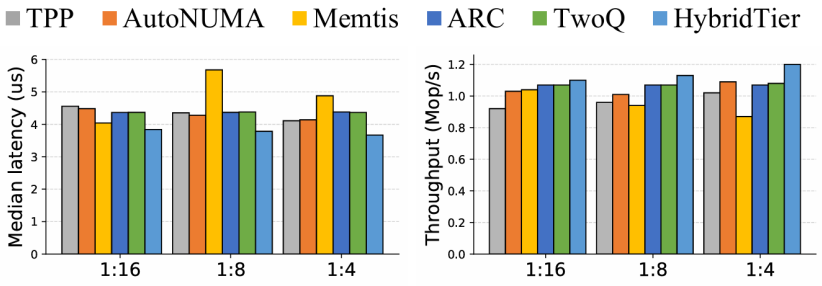
表 1-3 实验工作负载与数据规模

Application	**Input**	**Footprint**
Content-delivery network	CacheLib generator	267 GB
Social-graph	Kronecker graph	335 GB
Breadth-first search (BFS)		
Connected components (CC)	Uniform random graph	335 GB
Page Rank (PR)		
SPEC CPU 2017	603.bwaves	150 GB
	654.roms	150 GB
Silo	YCSB-C	208 GB
XGBoost	Criteo Click Logs	248 GB

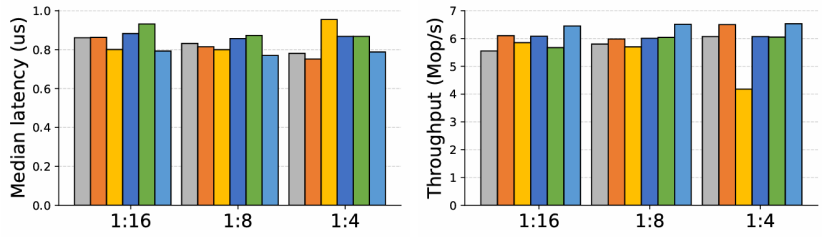
1.7 系统评估

1.7.1 Regular-Page 端到端性能

快层: 慢层 = 1:16 / 1:8 / 1:4（慢层固定 512 GB）。如图1-5和其他五个策略相比，HybridTier 在所有比例下均表现最佳，平均提升 19%（几何均值）。CacheLib：延迟 −18 %，吞吐 +23 %；1:16 就能打平别人 1:8。



(a) CacheLib CDN workload.



(b) CacheLib Social-graph workload.

图 1-5 Regular-Page 端到端性能

图 11 1-6展示了 HybridTier 相对于仅使用快层内存（即全 DRAM）基线的性能归一化结果，这代表了内存分层系统的性能上限。在 1:16、1:8 和 1:4 的内存配置下，HybridTier 平均仅比全快层内存慢 14%、9% 和 6%。

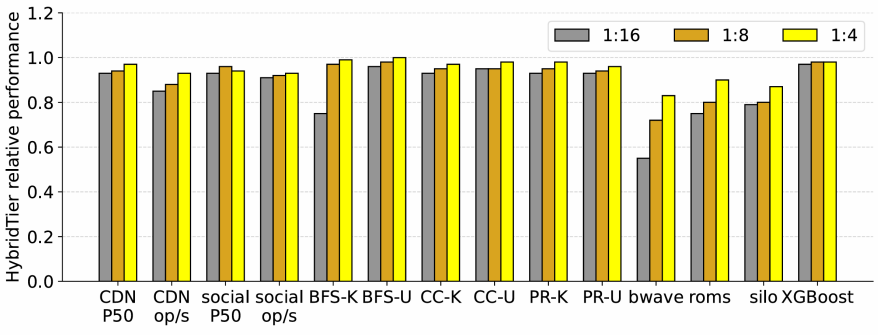


图 1-6 Regular-Page 快速层归一化性能

1.7.2 Huge-Page 端到端性能

为了评估 HybridTier 在巨页（huge pages）模式下的性能，我们在所有工作负载上与其对比 Memtis。图 12 1-7 显示，在 1:8 和 1:4 配置下，HybridTier 平均比 Memtis 分别高出 9% 和 11%，而在 1:16 配置下两者表现相当。HybridTier 在 CacheLib social-graph、BFS 和 Page Rank 工作负载上相对 Memtis 的性能提升最为显著。

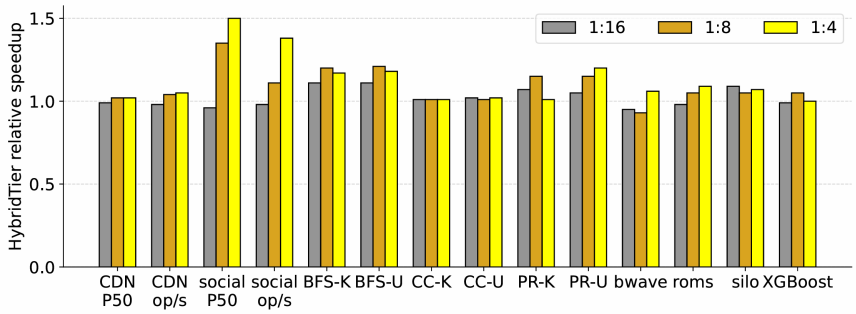


图 1-7 Huge-Page 端到端性能

1.7.3 详细对比 Memtis 测试

- 对动态分布适应性测试：在 CacheLib 工作负载中，HybridTier 适应新分布的速度是 Memtis 的 3.2 倍。

表 1-4 负载适应性测试

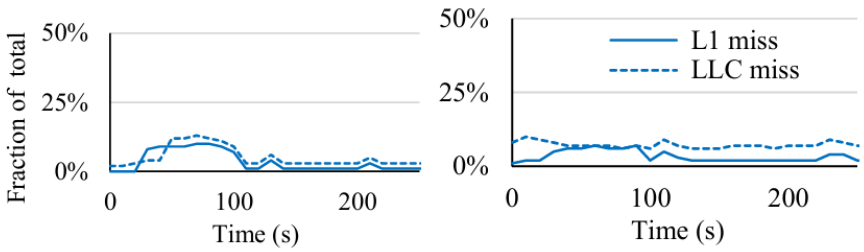
	CDN			Social-graph		
	1:16	**1:8**	**1:4**	**1:16**	**1:8**	**1:4**
Memtis	> 60	42.6	> 60	34.2	> 60	29.6
HybridTier	25.6	25.2	23.4	9.6	10.1	8.9
Relative Reduction	2.3X	1.7X	2.6X	3.6X	5.9X	3.3X

- 平均而言，HybridTier 比 Memtis 少产生了 4.6 倍的元数据开销。由于 HybridTier 的元数据大小随着快层内存大小的缩放而缩放，因此在较低的快层内存大小下实现了更大的内存节省。另一方面，Memtis 的元数据开销与总内存容量成比例缩放，因此在我们设置中保持不变。

表 1-5　元数据内存开销测试

Ratio	Mentis	HybridTier	Relative Reduction
1:16	0.39%	0.050%	7.8X
1:8	0.39%	0.097%	4.0X
1:4	0.39%	0.192%	2.0X

- 缓存开销测试：在常规和大页模式下，HybridTier 平均产生了总缓存未命中数的 5% 和 4%。总体上，使用常规页时，HybridTier 减少了 1.7 倍和 1.8 倍的 L1 和 LLC 缓存未命中数，而在大页模式下减少了 3.2 倍和 3.5 倍。图 14 进一步分解了在常规页下采用 4 位 CBF 和分块 CBF 的影响。使用标准 CBF（HybridTier-CBF）适度减少了缓存未命中数 12% 至 36%，应用分块 CBF（HybridTier-bCBF）进一步减少了 31% 至 72%。得出结论，两种优化都是有效的，而分块 CBF 提供了更大的减少。



(a) Under 4KB page.　　**(b)** Under huge page.

图 1-8　Cache Miss 比较

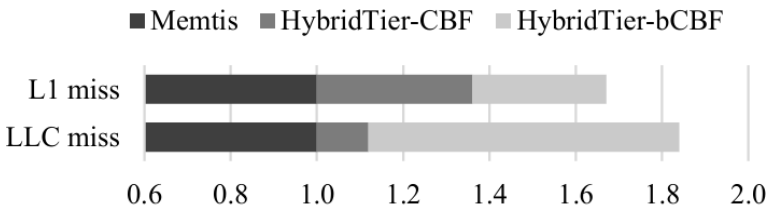


图 1-9　Cache Miss 分段延迟

1.7.4　理解 HybridTier 性能

- 频率-动量追踪（Frequency-Momentum Tracking）同时追踪频率和动量可以最有效地处理 CacheLib 和 XGBoost 工作负载，平均性能提升 8.5%。对于

BFS、CC 和 PR 内核，性能相似，因为这些工作负载的热集较小，可以轻松适应快速层内存。图 15 1-10展示了仅使用频率追踪器时的性能，1:8 配置下的性能。

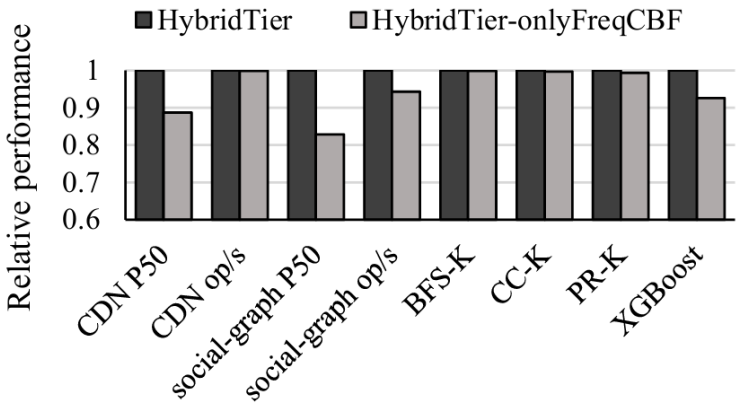


图 1-10　只用频率计数器的对比

- 计数布隆过滤器（Counting Bloom Filter）证明了 HybridTier 将每个访问计数器的大小限制为 4 位的设计决策是合理的，因为所有工作负载中超过计数器阈值的页面很少。表 5 显示，64 MB 的 CBF 在与哈希表进行比较时，超过 99.6% 的迁移决策是准确的。频率和动量追踪器都使用 CBF 实现。实际上，HybridTier 为动量 CBF 分配的内存比频率 CBF 少 128 倍。
- 分块布隆过滤器（Blocked CBF）k 个计数器可以映射到缓存行内的任何计数器。例如，图 8 显示每个缓存行包含 8 个计数器槽。实际上，4 位 CBF 的每个缓存行包含 128 个计数器槽。与标准 CBF 相比，分块 CBF 有略高的误报率 [61, 63]。然而，在实践中，我们发现性能收益是有利的权衡。