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 11 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

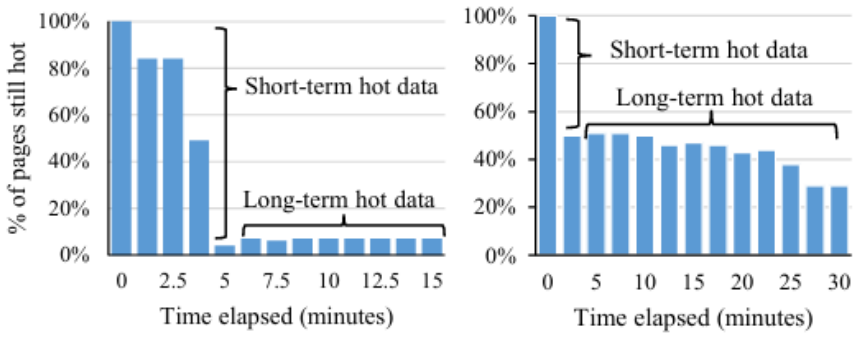
通过分析 Meta、Twitter 等生产环境，揭示大型内存工作负载的两大核心特征：

(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。

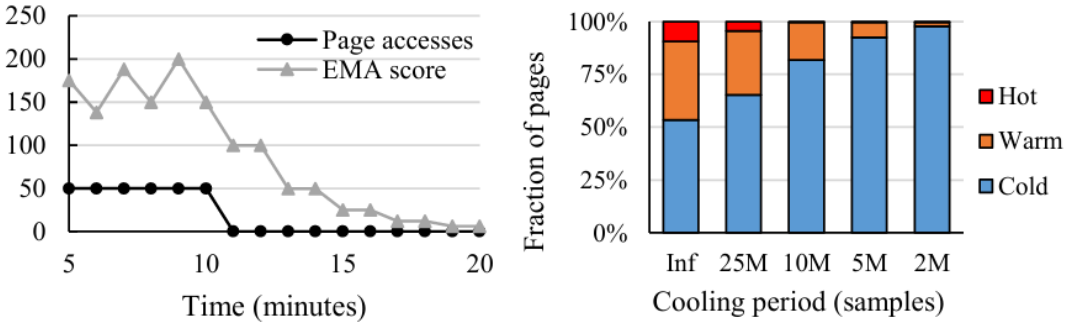


图 1-2　基于频率的系统适应热度分布变化

- 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

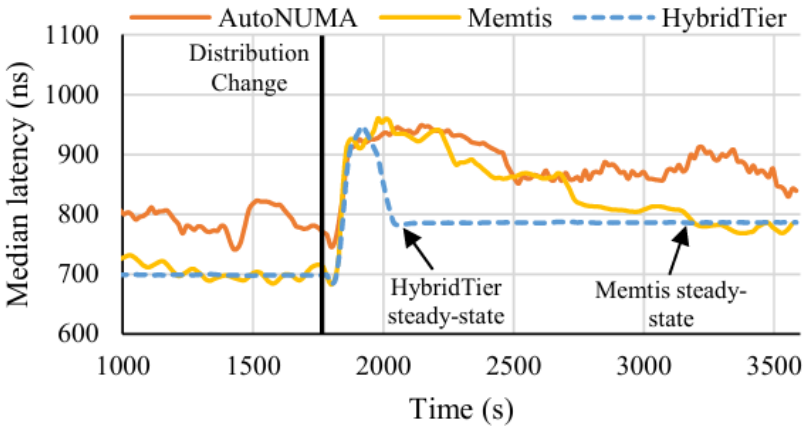


图 1-3　基于新近度的系统适应热度分布变化

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。

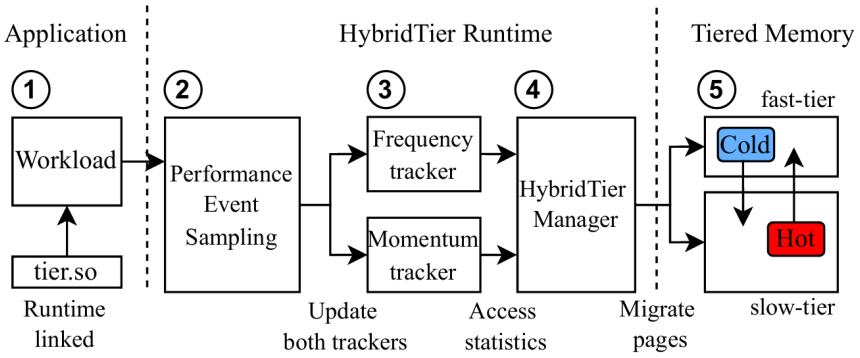


图 1-4 HybridTier 工作流

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

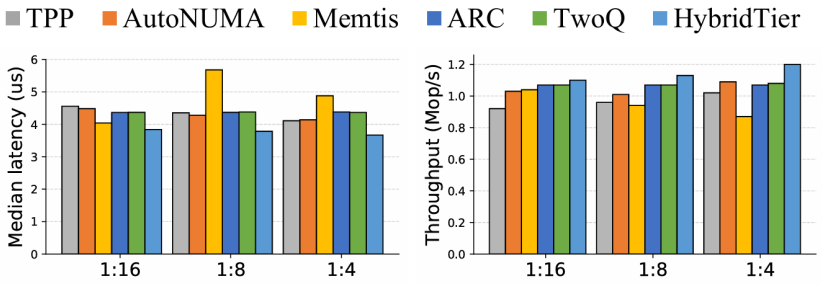
表 1-3 实验工作负载与数据规模

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

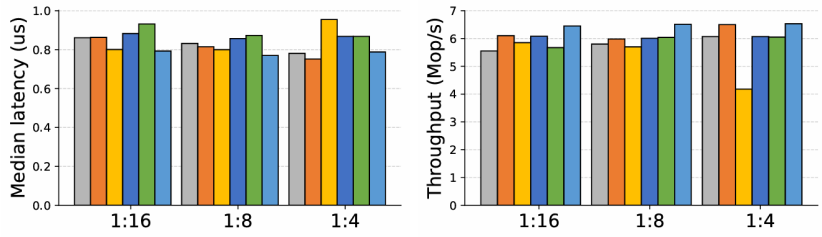
1.7 系统评估

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%。

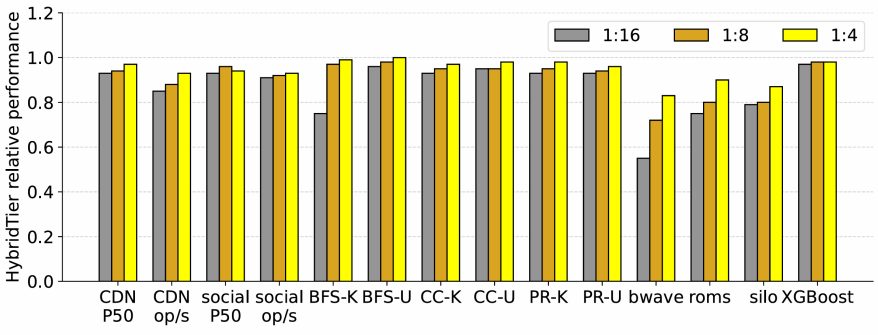


图 1-6 Regular-Page 快速层归一化性能

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。

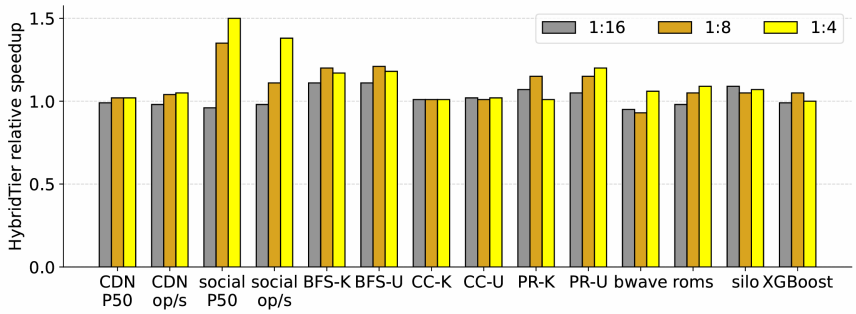


图 1-7 Huge-Page 端到端性能

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

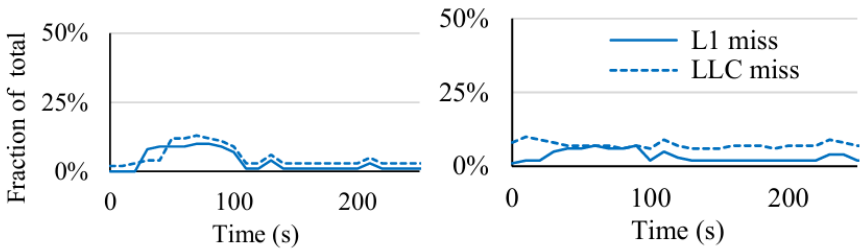
	CDN			Social-graph		
	1:16	**1:8**	**1:4**	**1:16**	**1:8**	**1:4**
Memtis	> 60	42.6	> 60	34.2	> 60	29.6
HybridTier	25.6	25.2	23.4	9.6	10.1	8.9
Relative Reduction	2.3X	1.7X	2.6X	3.6X	5.9X	3.3X

- 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

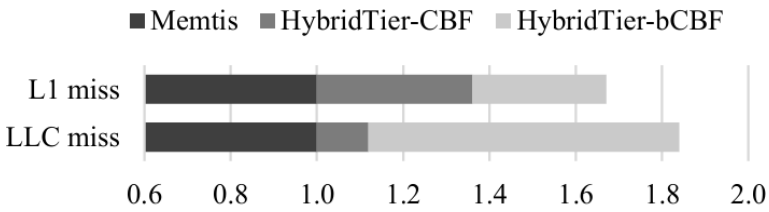


图 1-9　Cache Miss 分段延迟

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。

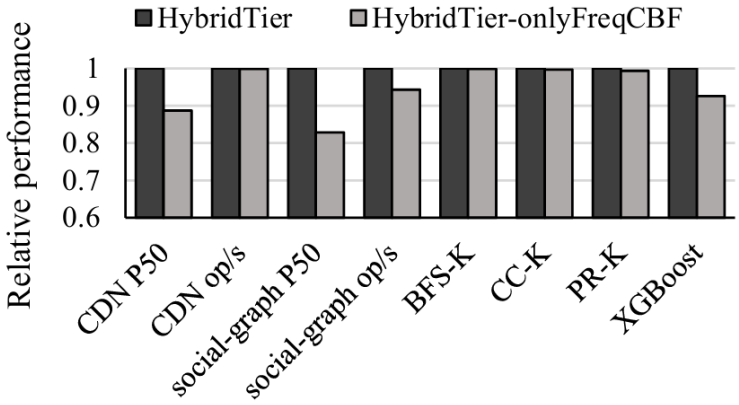


图 1-10　只用频率计数器的对比

- 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。

表 1-6　布隆过滤器准确率测试

CBF size (MB)	256	128	64	32	8
Accuracy	99.72%	99.65%	99.62%	99.42%	96.92%

- 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。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。
- 一项敏感性研究，以了解动量阈值对 HybridTier 性能的影响，相关结果如图 17 1-11所示。由于社交网络工作负载的热门数据集规模更大（见图 16），导致快速层级内存资源更为稀缺，因此其性能表现不如内容分发网络（CDN）。将动量阈值降低到 3 以下会对 HybridTier 的性能产生负面影响，因为这样

可能会将仅被访问过几次的冷页面错误地提升至快速层级。而对于我们所评估的工作负载而言，将动量阈值提高到 3 以上并不会显著改善性能。如果默认的动量阈值对于特定应用而言并非最优设置，HybridTier 允许用户调整该阈值的数值以提升性能。

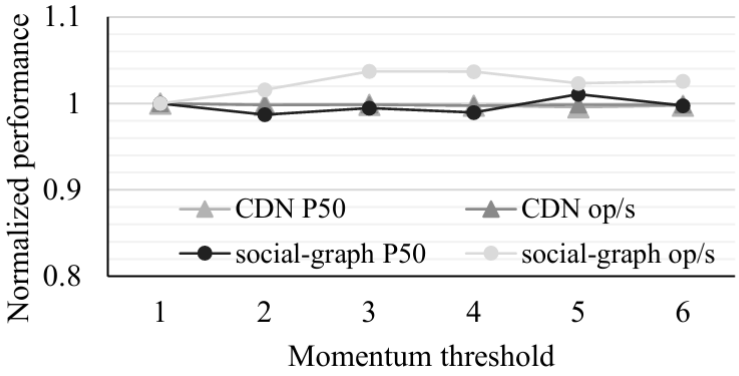


图 1-11　动量阈值敏感性测试

1.8　讨论、相关工作及结论

1.8.1　讨论（Discussions）

(1) **用户空间与内核空间分层对比**

- HybridTier 采用用户空间实现，核心优势为灵活性，该优势在现有内存分层系统（如 HeMem）及软件系统（用户空间网络、文件系统）中均获认可；
- 核心设计原则（频率-动量指标、概率性访问跟踪）具备通用性，可迁移至内核空间实现，不受当前部署模式限制。

(2) **全局内存分层支持方案**

- 架构设计：通过中央 HybridTier 控制器协调各独立实例，构建全局内存分层体系，适用于多租户虚拟机、共存应用场景；
- 数据交互：各实例向中央控制器上报本地冷热数据，由控制器统一制定全局数据晋升/降级决策，优化全局资源分配。

(3) **一次性访问模式处理**

- 针对扫描、指针追踪等一次性访问应用，可调整动量热度阈值参数；
- 权衡关系：提高阈值可减少快速层内存污染，但降低对热度变化的适应性，需结合应用场景平衡调整。

(4) **可配置参数选择**

- 频率热度阈值：借鉴 Memtis 方法，依据热度分布自动调整，无需人工干预；
- 动量热度阈值：经验性设为 3（在评估 workload 中效果优异），支持用户根据特定应用灵活修改。

1.8.2 相关工作（Related Works）

(1) **内存分层系统**

- 慢层内存类型与部署：慢层可采用固态硬盘、持久化内存；部署分本地（与快层同服务器，如 AutoNUMA、TPP）与远程（如 AIFM，需网络支持）；
- 可编程性分类：
  * 应用透明型：无需修改代码（如 AutoNUMA、TPP、Memtis、HybridTier），降低接入成本；
  * 应用依赖型：需剖析或修改代码（如 memkind、SMDK、Unimem），虽利用应用语义，但增加开发复杂度。
- 关键对比系统：
  * AutoNUMA/TPP：基于近期访问的 CXL 分层系统，存在冷热数据误分类问题；
  * Memtis/HeMem：Memtis 支持持久化内存与 CXL，HeMem 聚焦持久化内存页级频率分层，二者均有元数据开销高或适应性不足问题；
  * 其他系统：部分依赖硬件加速或聚焦特定优化（如 Memstrata 的多 VM 分配、NOMAD 的页面迁移优化），HybridTier 无需硬件/应用修改，可与其互补集成。

(2) **通用缓存算法**

- 基础算法：LRU（基于访问近期性）、CLOCK（LRU 近似），均无法兼顾频率与近期性；
- 混合缓存算法：
  * LRFU：加权平均整合近期性与频率，调整衰减因子可切换策略，但无法独立精准跟踪两类指标；
  * ARC：双 LRU 列表区分单次/多次访问项，不维护频率计数，难以区分热/温数据，资源受限场景性能受限。
- HybridTier 优势：通过独立计数器分别跟踪频率与近期性，解决现有混合算法的跟踪精度问题。

(3) **近似数据结构**

- 应用领域：概率性数据结构（如布隆过滤器）已用于数据库、内存缓存、网络通信，优化存储与查询效率；
- HybridTier 创新：首次将计数布隆过滤器（CBF）应用于内存分层，结合 CBF 特性设计高效晋升/降级机制与迁移策略，实现细粒度频率分层。

(4) **离散内存系统**

- 核心目标：通过远程服务器扩展内存容量，需将热数据放置本地快层以减少延迟；
- 与 HybridTier 关系：技术方向正交，目标服务器可在 HybridTier 管理本地 CXL 内存基础上，通过离散系统进一步扩展容量，形成多层级管理体系。

1.8.3 结论（Conclusion）

(1) **HybridTier 核心定位**：提出具备自适应性与轻量级的分层内存系统，解决现有方案在适应性、元数据开销方面的关键问题，满足现代 workload 对大内存容量与高性能的双重需求。

(2) **核心技术亮点**

- 双指标跟踪：同时跟踪长期访问频率与短期访问动量，快速适应动态访问分布，精准识别冷热数据变化，规避传统单指标局限性；

- 低开销优化：采用概率性访问跟踪（计数布隆过滤器），大幅降低元数据的内存开销与缓存开销，兼顾性能与资源效率，适合数据中心大规模部署。

(3) **性能优势总结**：在多 workload 与内存配置下，HybridTier 性能优于主流分层系统（平均提升 19%，最高达 91%），同时降低内存开销（2.0-7.8 倍）与缓存开销（1.7-3.5 倍），验证了其实用性与高效性，为 CXL 内存分层技术提供可靠解决方案。

2 论文内容分析

2.1 论文简要总结

该论文针对现代工作负载对大内存容量的需求，提出了一种基于 Compute Express Link（CXL）的自适应轻量级内存分层系统——*HybridTier*。论文首先指出传统内存扩展方案（如增加 DRAM）面临空间限制、成本高昂及密度增长放缓的问题，而 CXL 内存分层虽能通过“本地 DRAM（快层）+ CXL 内存（慢层）”平衡容量与成本，但现有分层系统存在两大核心挑战：一是难以适配倾斜且动态变化的数据热度分布，二是分层过程中元数据的内存与缓存开销过高。

为解决上述问题，HybridTier 设计了三大核心机制：

(1) **双指标热度跟踪**：同时跟踪数据的长期访问频率（分钟至小时级）与短期访问动量（秒级），通过独立的指数移动平均（EMA）计数器实现对动态热度分布的精准捕捉与快速适应；

(2) **概率性元数据优化**：采用计数布隆过滤器（CBF）替代传统精确数据结构（如哈希表），以微小且对性能无显著影响的跟踪误差，将元数据内存开销降低 2.0-7.8 倍；

(3) **缓存 locality 优化**：通过分块 CBF（blocked CBF）确保元数据访问仅触发单次缓存访问，将缓存缺失率降低 1.7-3.5 倍。

实验评估表明，在 6 类大型内存工作负载（如 CacheLib、GAP 图计算、SPEC CPU 2017 等）与不同快/慢层内存比例下，HybridTier 相较于 AutoNUMA、Memtis 等主流系统，平均性能提升 19%（最高达 91%），同时兼顾低开销与高适应性，且已开源（https://github.com/kevins981/hybridtier-asplos25-artifact）。

2.2 论文优势

2.2.1 设计创新方面

(1) **问题分析清晰准确**：首先，论文对现有问题进行了清晰准确的定位。佐以验证实验，说明了现有方案或者说需求的三个问题：（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；

(2) **双指标热度跟踪机制创新**：论文首先分析了单一热度指标（频率或近期性）在动态热度分布下的局限性，进而提出结合“频率 + 动量”双指标的跟踪机制。该设计既能利用频率指标捕获长期热点，又能借助动量指标快速响应短期访问突发，显著提升了系统对动态工作负载的适应能力。看似像是“拼好算法”，其实是独具匠心；

(3) **论文测试细致，分析到位**：论文测试列举多个对比方案，配置清晰，测试数据详实。并且对测试结果进行了细致的分析，能够从数据中提炼出有价值的信息，验证了设计的有效性。尤其是和 Memtis 的对比，体现了 HybridTier 在适应性和开销上的优势。

2.2.2 写作与内容方面

1. **问题定位精准，逻辑闭环完整**：论文以现代工作负载“大内存需求”与“传统 DRAM 局限”的核心矛盾为切入点，先通过数据量化问题紧迫性，再聚焦 CXL 内存分层的“动态热度适配”与“元数据开销”两大子问题，最终提出 HybridTier 的解决方案并通过实验验证效果，形成“问题提出-挑战分析-方案设计-实验验证”的完整逻辑链，论证层次清晰且说服力强。

2. **核心设计解释清楚，细节明了**：对核心创新点的解释兼具“原理深度”与“工程细节”：- 双指标跟踪机制中，明确频率计数器（高 EMA 冷却周期 C）与动量计数器（低 EMA 冷却周期 C）的参数设置逻辑，并用表 1 清晰呈现“频率-动量”组合下的晋升/降级策略；- 概率性元数据优化部分，不仅介绍计数布隆过滤器（CBF）的选型理由，还给出具体参数计算（如 $k = 4$ 哈希函数、$p = 0.001$ 误差率的推导）与分块 CBF 的缓存优化原理，同时通过表 5 量化不同 CBF 大小的决策准确率。

3. **实验设计严谨，结果呈现直观**：实验方案覆盖“横向对比-纵向分析-敏感性验证”多维维度，且结果呈现方式高效：- 横向对比：选取 AutoNUMA、Memtis

等 5 类主流系统作为基线，在 6 类工作负载、3 种快/慢层比例下展开性能测试，用几何均值与最高值双重指标体现优势（如平均性能提升 19%、最高达 91%）；- 纵向分析：通过图 4（热度分布变化后的适应时间）、图 13（缓存缺失率占比）等量化 HybridTier 在“适应性”“低开销”上的核心价值；- 敏感性验证：针对动量阈值、CBF 大小等关键参数做变量实验，明确“动量阈值=3”“CBF≥64MB”为最优配置，为工程落地提供明确指导。

4. **相关工作对比客观，突出自身定位**：论文将相关工作分为“内存分层系统”“通用缓存算法”“近似数据结构”“离散内存系统”四类，既肯定现有方案的贡献（如 Memtis 的频率直方图精准度、ARC 的轻量级设计），又客观指出其缺陷（如频率基系统适应慢、近期基系统识别准度低），并通过对比凸显 HybridTier“双指标 + 概率性跟踪”的创新价值，避免片面贬低他人工作，体现学术严谨性。

2.3 论文不足

(1) **全局内存分层场景支持不足**：论文仅实现单机级分层，虽提及“中央控制器 + 本地实例”的全局分层架构，但未提供具体设计（如控制器与实例的通信协议、全局热点冲突解决机制）与实验验证。在多租户虚拟机、共存应用的集群场景下，HybridTier 的全局资源调度能力、跨节点数据迁移开销尚不明确。

(2) **真实 CXL 硬件环境验证缺失**：实验采用“双 socket NUMA 节点模拟 CXL 内存”（延迟 124ns、带宽 34GB/s），而非真实 CXL 1.1/2.0 硬件设备。真实 CXL 环境中存在的链路波动、协议开销、多设备并发访问等特性，可能导致 HybridTier 的迁移决策延迟、缓存开销等指标与模拟环境存在偏差，削弱结论的硬件落地参考价值。

(3) **长期运行稳定性与容错性未提及**：论文实验周期集中在“热度分布变化后至稳态”的短期阶段（如 CacheLib 适应时间测试为 30 分钟内），未评估长期运行（如 24 小时以上）中元数据累积误差、内存泄漏等问题；同时未涉及 CXL

内存故障、页面迁移中断等容错场景的处理方案，难以支撑数据中心级的高可用需求。

2.4 论文改进建议

(1) **补充集群级设计与验证**：- 设计轻量化中央控制器：采用“本地预决策 + 全局仲裁”机制，本地实例先基于局部数据生成迁移候选，控制器结合全局快层内存利用率、跨节点带宽等信息优化决策，避免热点数据重复晋升；- 实验验证：在 Kubernetes 集群环境中部署多租户应用（如混合 CDN 缓存与图计算），量化全局分层对集群整体吞吐量、延迟标准差的提升效果。

(2) **修正模拟环境偏差**：- 搭建真实 CXL 测试床：采用支持 CXL 2.0 的内存扩展模块（如三星 512GB CXL 模块），测量真实延迟（50-100ns 额外延迟）、带宽（本地 DRAM 的 20-70%）下 HybridTier 的性能；- 优化硬件适配：针对 CXL 链路波动，增加“迁移延迟感知”的决策调整——当 CXL 带宽利用率 >80% 时，延迟迁移非核心热点页面，避免链路拥塞。

(3) **增强长期稳定性与容错性，适配数据中心需求**：- 长期运行优化：定期（如每小时）对 CBF 元数据进行“误差校准”，通过扫描部分页面的真实访问计数修正累积误差；

2.4.1 收获与总结

”Local Memory + CXL Memory” 的分层内存架构，并不是新概念，但 HybridTier 通过“双指标热度跟踪 + 概率性元数据优化”的创新设计，成功解决了动态热度适配与低开销管理的核心挑战，显著提升了分层内存系统的实用性与性能。论文在问题定位、设计创新、实验验证等方面均表现出较高水平，值得学习借鉴。

在问题分析，需求提取和动机实验的验证方面，论文做得非常到位，能够清晰地展现出研究的必要性与挑战性。在设计方案上，双指标热度跟踪机制体现了对内存访问模式深刻理解，而概率性数据结构的应用则展示了工程实践中的巧思。此外，实验设计严谨且结果解读透彻，为结论提供了有力支撑。

在论文写作方面也有许多值得借鉴的地方，如清晰的逻辑结构、详实的数据分析、客观的相关工作对比等，都体现了良好的学术写作规范。未来在类似研究中，可以参考 HybridTier 的设计思路，同时注意补充集群级支持、真实硬件验证与长期稳定性等方面的内容，以提升研究的全面性与实用价值。

3 部分实验图表分析

表格3-1系统梳理 AutoNUMA、TPP、Memtis 等 6 种主流方案的核心设计（如实现层级、跟踪策略、元数据管理方式），明确各方案的优势（如 HybridTier 轻量低耗、支持多页模式）与短板（如 Memtis 动态热度适配慢、元数据开销高），让不同方案的技术差异直观可辨，为分层系统选型提供直接参考。

表 3-1 6 种主流内存分层方案核心特点

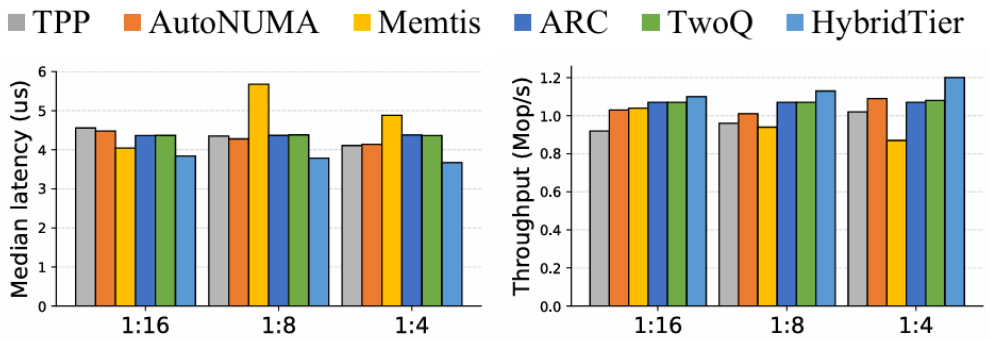
序号	方案名称及核心特点
1	**AutoNUMA**：通过 LRU 策略跟踪页面近期访问性，无频率维度识别能力，易导致冷热数据误分类。
2	**TPP**：聚焦 CXL 内存的页级迁移优化，基于访问延迟反馈调整页面位置；依赖硬件 NUMA 节点模拟 CXL 特性，迁移决策仅考虑局部节点；因内核态实现，迁移开销略高。
3	**Memtis**：支持持久化内存与 CXL 内存双层架构，通过频率直方图跟踪页面访问频率；采用哈希表存储精确计数，元数据开销高；对动态热度分布的适应速度较慢（分钟级）。
4	**ARC**：轻量级混合 LRU 列表缓存算法，可区分单次与多次访问项；因不维护频率计数，无法区分热数据与温数据；在快层占比低的资源受限场景中性能衰减明显。
5	**TwoQ**：采用 FIFO+LRU 双队列缓存策略，分离新访问与频繁访问页面；实现简单且缓存开销低，但对高倾斜度的访问分布适配性较差。
6	**HybridTier**：通过频率 + 动量双指标跟踪适配动态热度；采用计数布隆过滤器（CBF）降低元数据内存与缓存开销；支持 4KB 页与 2MB 巨页。

注：所有方案均为应用透明型，无需修改业务代码即可部署。

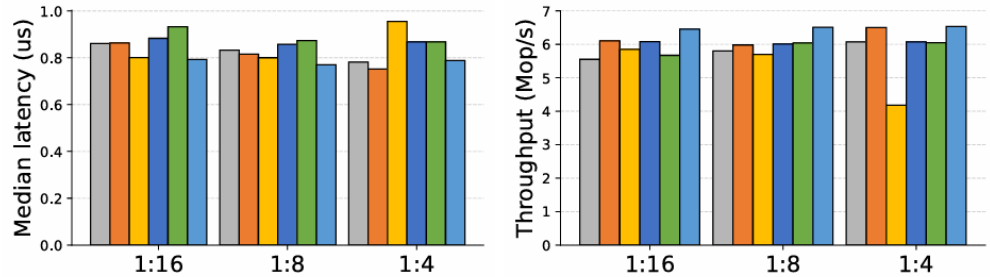
表格3-2提炼 CacheLib（CDN 缓存）、GAP（图计算）、Silo（数据库）等 6 类典型大内存负载的核心访问特征（如热点动态性、局部性、读写比例），覆盖互联网缓存、科学计算、AI 训练、数据库等关键应用场景，确保后续实验负载能代表真实业务需求，避免实验结论局限于单一场景。

表 3-2　6 类测试负载核心特点

序号	负载类型及核心特点
1	**CacheLib**：CDN 缓存，热点动态变化
2	**GAP**：图计算，随机访问局部性差
3	**SPEC CPU 2017**：访问模式相对稳定
4	**Silo**：数据库，热点集中读写混合
5	**XGBoost**：机器学习，阶段性热点显著
6	**BFS**：图遍历，热点随层级变化



(a) CacheLib CDN workload.



(b) CacheLib Social-graph workload.

图 3-1　CacheLib 工作负载实验结果分析

3.1 CacheLib 工作负载实验分析

a. 实验目标（Experiment Target）

评估不同内存分层方案在 CacheLib 两类子工作负载（CDN、社交图谱）下的性能（延迟、吞吐量），验证各方案对不同快/慢内存比例的适配能力。

b. 实验设计思想（Experiment Methodology）

选取 6 种内存分层方案（TPP、AutoNUMA、Memtis、ARC、TwoQ、HybridTier），在 CacheLib 的 CDN 与社交图谱子工作负载下开展测试；以“快层内存（DRAM）：慢层内存（CXL 内存）”的比例（1:16、1:8、1:4）为变量，分别测量各方案的中位数延迟与吞吐量指标。

c. 实验具体配置（Experiment Configuration）

- 测试工作负载：CacheLib CDN 工作负载、CacheLib 社交图谱工作负载；
- 内存分层方案：TPP、AutoNUMA、Memtis、ARC、TwoQ、HybridTier；
- 内存比例：快层内存: 慢层内存 = 1:16、1:8、1:4；
- 测量指标：中位数延迟（单位：微秒）、吞吐量（单位：Mop/s，每秒百万操作数）。

d. 图表标记含义（x-axis and y-axis）

- 横坐标：快层内存与慢层内存的比例（1:16、1:8、1:4）；
- 纵坐标：
  - 左图：中位数延迟（单位：μs，微秒）；
  - 右图：吞吐量（单位：Mop/s，每秒百万操作数）；
- 图表类型：分组柱状图；
- 系列标记：不同颜色代表不同内存分层方案（TPP：灰色，AutoNUMA：橙色，Memtis：黄色，ARC：深蓝色，TwoQ：绿色，HybridTier：浅蓝色）。

e. 实验结论（Experiment Conclusion）

在 CacheLib 两类工作负载下，HybridTier 方案的综合性能（低延迟、高吞吐量）显著优于其他 5 种方案；Memtis 的性能波动较大（在 1:4 比例下延迟

明显升高），而 HybridTier 在不同内存比例下的性能表现更稳定。

f. 结果解释（Experiment Explanation）

- HybridTier 采用“频率 + 动量”双指标跟踪，能更好适配 CacheLib 工作负载的动态热点变化，因此在不同内存比例下均保持低延迟与高吞吐量；

- Memtis 依赖频率直方图跟踪热点，对 CacheLib 的动态热点适应速度慢，当快层内存占比降低（如 1:4）时，易出现冷页误晋升导致的快层内存污染，进而推高延迟、降低吞吐量；

- 其他方案（如 AutoNUMA、ARC）因仅跟踪单一维度（近期访问性），无法精准识别动态热点，性能表现弱于 HybridTier。

3.2 动态访问分布适应时间实验分析

Table 3. Minutes required to adapt to new access distribution (reach within 1% of the steady-state median latency).

	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	2.3×	1.7×	2.6×	3.6×	5.9×	3.3×

图 3-2　动态访问分布适应时间实验结果分析

a. 实验目标（Experiment Target）

评估不同内存分层方案在访问分布发生变化后，达到稳态性能所需的适应时间，验证各方案对动态访问模式的响应能力。

b. 实验设计思想（Experiment Methodology）

选取 Memtis 与 HybridTier 两种内存分层方案，在 CDN 和 Social-graph 两类工作负载下，测试不同快/慢内存比例（1:16、1:8、1:4）下，方案延迟达到稳态中位数延迟 1

c. 实验具体配置（Experiment Configuration）

- 测试工作负载：CDN 工作负载、Social-graph 工作负载；

- 内存分层方案：Memtis、HybridTier；

- 内存比例：快层内存: 慢层内存 = 1:16、1:8、1:4；

- 评估指标：适应时间（单位：分钟，定义为延迟接近稳态的时间）。

d. 表格标记含义

- 行维度：包含 2 种内存分层方案（Memtis、HybridTier）及两者的适应时间相对减少幅度；

- 列维度：分为 CDN 与 Social-graph 两类工作负载，每类负载下包含 3 种快/慢内存比例（1:16、1:8、1:4）；

- 数据项：单元格数值为对应方案在特定负载与内存比例下的适应时间（单位：分钟），“>60”表示适应时间超过 60 分钟。

e. 实验结论（Experiment Conclusion）

HybridTier 对动态访问分布的适应速度显著优于 Memtis：在两类工作负载与各内存比例下，HybridTier 的适应时间均远短于 Memtis，且适应时间的稳定性更强；相对 Memtis，HybridTier 的适应时间最多可减少 5.9 倍。

f. 结果解释（Experiment Explanation）

- Memtis 依赖频率直方图跟踪访问模式，更新频率低、对动态变化响应慢，因此在多数场景下适应时间超过 60 分钟；

- HybridTier 采用“频率 + 动量”双指标跟踪，动量指标可快速捕捉短期访问变化，因此能在 25 分钟内（CDN 负载）或 10 分钟内（Social-graph 负载）适应新的访问分布；

- 内存比例对 Memtis 的适应时间影响极大（如 Social-graph 负载下 1:8 比例需 >60 分钟、1:4 比例仅需 29.6 分钟），而 HybridTier 的适应时间受内存比例影响较小，体现了其更强的场景适配稳定性。

4 总结与收获

论文聚焦的 CXL 内存分层技术，是当前解决“DRAM 容量瓶颈”与“存储性能需求”矛盾的核心方向之一，这与我的 SSD 软硬件专业高度关联——SSD 作为传统存储层级的“快层”，其与 CXL 内存的协同（如“SSD+CXL 内存 +DRAM”的多层级存储架构）正成为存储系统的前沿研究领域。通过阅读，我明确了内存分层的核心矛盾：动态热度适配与元数据开销的平衡，而这一矛盾同样存在于 SSD 的缓存管理中（如 SSD 的 SLC 缓存如何适配热点数据的动态变化）。同时，论文中“频率 + 动量”的双指标跟踪机制，也为 SSD 的垃圾回收（GC）策略优化提供了启发——SSD 的 GC 可结合数据的访问频率（长期热度）与近期访问动量（短期热点），优先回收冷数据块，提升 GC 效率。

此外，论文中计数布隆过滤器（CBF）的轻量级元数据设计，也补充了我对“近似数据结构在存储系统中应用”的认知：CBF 不仅可用于内存分层的热点跟踪，也可优化 SSD 的磨损均衡（Wear Leveling）中的数据热度统计，在保证精度的前提下降低 SSD 控制器的计算与存储开销。

论文的写作逻辑呈现出严谨的“问题驱动-创新突破-实验验证”闭环，这是学术论文的核心写作范式。开篇以“现代工作负载内存需求激增”与“DRAM 成本/密度瓶颈”为切入点，通过数据（如 Meta 的内存成本占比）量化问题的紧迫性；随后聚焦 CXL 内存分层的两大挑战（动态热度适配、元数据开销），明确研究目标；继而提出 HybridTier 的双指标跟踪、概率性元数据等创新方案，每一项设计均对应具体挑战；最终通过多维度实验验证方案的有效性。

这一逻辑对我的专业写作极具启发：在 SSD 相关研究中，需先明确具体场景的痛点（如 SSD 在边缘设备中的功耗与性能矛盾），再针对性提出技术方案（如自适应功耗的 NAND 闪存调度策略），最后通过端到端实验（功耗、延迟、寿命）验证价值，避免“为创新而创新”的空泛设计。

论文的内容编排采用“摘要-引言-相关工作-方案设计-实验评估-讨论-结论”的经典模块化结构，但每个模块的信息密度与衔接性极强：- 引言部分通过“问题-挑战-贡献”的三段式结构，快速传递核心价值；- 相关工作部分按“内存分层系统-缓存算法-近似数据结构”分类，既梳理领域进展，又突出自身创新；- 方

案设计部分拆解为“双指标跟踪-元数据优化-系统实现”，层层递进地解释技术细节；- 实验评估部分覆盖“性能对比-开销分析-敏感性验证”，全面支撑结论。

这种编排方式对 SSD 专业论文的写作具有指导意义：例如在“SSD 缓存管理优化”的论文中，可将方案拆解为“热点识别-缓存替换-硬件适配”，实验部分覆盖“不同负载下的缓存命中率-SSD 写放大-功耗”等多维度指标，确保内容既深入又系统。论文的实验设计体现了“全面性-严谨性-代表性”的特点，是验证技术方案的关键支撑。

这对我研究 SSD 相关实验设计的启发在于：在测试 SSD 的新调度算法时，需控制“NAND 类型（TLC/QLC）”“负载类型（随机/顺序读写）”“队列深度”等变量，选取现有主流调度算法（如 Deadline、CFQ）作为基线，同时测量“IOPS-延迟-写放大-寿命”等多维度指标，避免实验结论的局限性。

从 SSD 软硬件的专业视角看，论文的技术思路可直接迁移至以下方向：

(1) SSD 缓存管理优化：HybridTier 的“频率 + 动量”双指标跟踪，可用于 SSD 的 SLC 缓存热点识别——传统 SSD 缓存仅跟踪访问频率，易误判“一次性突发访问”为热点，而加入动量指标可快速区分短期突发与长期热点，减少缓存污染；

(2) SSD 控制器元数据优化：论文中 CBF 的轻量级设计，可用于 SSD 的“数据热度统计”与“磨损均衡”——传统 SSD 采用精确计数的哈希表存储数据访问次数，元数据开销高，而 CBF 可在精度损失可接受的前提下降低控制器的存储与计算负载；

(3) 多层级存储架构协同：CXL 内存与 SSD 的协同是未来存储系统的重要方向，HybridTier 的分层策略可拓展为“DRAM+CXL 内存 +SSD”的三层架构——通过双指标跟踪，将长期热点放在 DRAM、短期热点放在 CXL 内存、冷数据放在 SSD，实现容量与性能的平衡。

这些关联不仅拓展了我对 SSD 技术边界的认知，也为后续的专业研究提供了新的技术思路。

本次论文阅读不仅补充了 CXL 内存分层的背景知识，更在论文写作逻辑、内容编排、实验设计等方面形成了可复用的方法论，同时为 SSD 软硬件专业的研究提供了技术迁移与协同优化的方向。学术研究的核心在于“问题的精准定

位”与“方案的针对性创新”，而实验设计的全面性是结论说服力的关键——这一认知将指导我后续的专业学习与研究实践。

5 知识点与题目编写

5.1 考点

(1) 内存分层系统的延迟、吞吐量指标关联分析；

(2) 不同方案的性能优势量化（相对提升幅度计算）；

(3) 动态访问分布下的性能稳定性评估。

5.2 题目描述

在 CacheLib CDN 工作负载（快层: 慢层内存比例 =1:8）场景下，测试得到以下数据：- Memtis 的中位数延迟为 42.6 μs，吞吐量为 0.95 Mop/s；- HybridTier 的中位数延迟为 25.2 μs，吞吐量为 1.05 Mop/s。

请完成以下计算：(1) 计算 HybridTier 相对 Memtis 的延迟降低比例与吞吐量提升比例；(2) 若某 CDN 服务的业务请求量为 1.2×10^6 次/分钟，分别计算 Memtis 与 HybridTier 的请求处理耗时（单位：秒），并分析两者的服务响应能力差异。

5.3 计算公式

1. 延迟降低比例：

$$\text{延迟降低比例} = \left(1 - \frac{\text{HybridTier 延迟}}{\text{Memtis 延迟}}\right) \times 100\%$$

2. 吞吐量提升比例：

$$\text{吞吐量提升比例} = \left(\frac{\text{HybridTier 吞吐量}}{\text{Memtis 吞吐量}} - 1\right) \times 100\%$$

3. 请求处理耗时：

$$\text{耗时（秒）} = \frac{\text{总请求数}}{\text{吞吐量（次/秒）}}$$

（注：吞吐量单位转换：1 Mop/s = 1×10^6 次/秒）

5.4 解答

(1) 延迟降低比例与吞吐量提升比例计算：- 延迟降低比例：

$$\left(1 - \frac{25.2}{42.6}\right) \times 100\% \approx (1 - 0.5915) \times 100\% \approx 40.85\%$$

- 吞吐量提升比例：

$$\left(\frac{1.05}{0.95} - 1\right) \times 100\% \approx (1.1053 - 1) \times 100\% \approx 10.53\%$$

(2) 请求处理耗时计算：总请求数为 1.2×10^6 次/分钟，即 2×10^4 次/秒。

- Memtis 的处理耗时：

$$\frac{2 \times 10^4}{0.95 \times 10^6} \approx 0.0211 \text{ 秒}$$

- HybridTier 的处理耗时：

$$\frac{2 \times 10^4}{1.05 \times 10^6} \approx 0.0190 \text{ 秒}$$

5.5 分析

HybridTier 的延迟较 Memtis 降低约 40.85%，吞吐量提升约 10.53%；在相同请求量下，HybridTier 的处理耗时更短（减少约 0.0021 秒）。这表明在 CDN 场景中，HybridTier 的服务响应速度更快、处理能力更强，更能适配 CDN 的高并发、低延迟需求。