并发平均内存访问时间（C-AMAT）与分层性能匹配（LPM）的研究现状

赵晓菲
兰州石化职业技术大学、兰州交通大学

2025 年 4 月 15 日

目录

1 引言

随着处理器速度的飞速发展，处理器与内存之间的速度差距日益扩大，这一问题通常被称为“内存墙”[1]。这种速度差异严重制约了计算机系统的整体性能，使得高效的内存层次结构管理成为弥合这一性能鸿沟的关键。为了应对这一挑战，研究人员不断探索新的方法来分析和优化内存系统的性能。并发平均内存访问时间（Concurrent Average Memory Access Time, C-AMAT）模型和分层性能匹配（Layered Performance Matching, LPM）算法应运而生，旨在更精确地评估现代内存系统的性能，并优化内存层次结构中不同层级之间的性能匹配，从而减少内存访问延迟。C-AMAT 模型通过统一考虑数据局部性、并发性和重叠性对内存访问时间的影响，为更准确地评估现代内存系统提供了理论基础。基于 C-AMAT，LPM 算法旨在通过匹配内存层次结构中相邻层级之间的数据请求速率和数据供给速率，实现整体内存系统性能的优化。本报告旨在对 C-AMAT 模型和 LPM 算法的研究现状进行全面的分析和总结，涵盖其定义、原理、在统一数据局部性和并发性影响方面的作用、优化内存层次结构性能匹配的具体方法、实验验证、应用案例、最新研究进展、与其他类似模型和算法的比较以及社区评价和开源实现情况。通过对这些方面的深入探讨，本报告旨在为相关领域的研究人员和工程师提供一个详尽的参考。

2 并发平均内存访问时间（C-AMAT）模型

2.1 C-AMAT 的定义与原理

并发平均内存访问时间（C-AMAT）模型是一种用于分析和优化现代分层内存系统性能的通用模型 [2]。它通过递归定义内存层次结构中每一层的内存访问延迟来工作 [2]。与传统的平均内存访问时间（Average Memory Access Time, AMAT）模型不同，C-AMAT 显式地考虑了现代内存系统支持并发数据访问的特性，即在同一个内存周期内可以处理多个内存访问请求 [3]。AMAT 模型主要关注数据局部性对内存性能的影响，假设内存访问是串行的 [4]。而 C-AMAT 通过引入并发性的概念，能够更准确地评估现代内存系统的性能，尤其是在大数据应用等高并发场景下 [3]。

C-AMAT 模型基于以下关键原理 [3]：

- **并发性**：C-AMAT 明确考虑了并发数据访问，这是现代内存系统的一个重要特性，与 AMAT 假设的串行访问形成对比 [3]。
- **重叠模式**：C-AMAT 以重叠模式计算总内存访问周期。这意味着如果在同一周期内发生多个内存访问，周期计数仅增加一次，而内存访问总数则增加实际发生的访问次数 [3]。
- **内存活跃周期**：C-AMAT 仅包括有内存访问活动的时钟周期，空闲周期不计入计算 [3]。这与每周期访问次数（Access Per Cycle, APC）的概念相关，APC 是 C-AMAT 的倒数 [3]。
- **纯缺失**：C-AMAT 引入了“纯缺失”的概念，指至少包含一个“纯缺失周期”的缺失。纯缺失周期是指不与任何命中周期重叠的缺失周期 [5]。C-AMAT 基于纯缺失重新定义了缺失率和平均缺

失代价，得到“纯缺失率”（pure Miss Rate, pMR）和“纯平均缺失代价”（pure Average Miss Penalty, pAMP）[3]。只有纯缺失周期才直接导致处理器停顿 [6]。

- **平衡优化**：C-AMAT 揭示，减少传统缓存缺失的数量不如减少纯缺失的数量对性能提升更重要。它还表明，仅提高数据局部性可能并不总能改善性能，优化应侧重于在数据局部性和并发性之间取得平衡 [3]。

C-AMAT 的基本公式为 [3]：

$$\text{C-AMAT} = \frac{\text{总内存访问周期}(T_{\text{MemCycle}})}{\text{总内存访问次数}(C_{\text{MemAcc}})}$$

扩展的参数化 C-AMAT 公式为 [3]：

$$\text{C-AMAT} = \frac{H}{C_h} + \frac{\text{pMR} \times \text{pAMP}}{C_{pm}}$$

其中，H 为命中延迟（Hit Latency），C_h 为命中并发度（Hit Concurrency），pMR 为纯缺失率（Pure Miss Rate），pAMP 为纯平均缺失代价（Pure Average Miss Penalty），C_{pm}（或 C_M）为纯缺失并发度（Pure Miss Concurrency）[3]。

C-AMAT 模型的核心创新在于它从以 CPU 为中心的串行内存访问视角（如 AMAT）转变为以内存为中心的视角，并考虑了现代内存系统中固有的并行性。引入“纯缺失”的概念是一个关键的区分点，因为它关注的是真正导致处理器停顿的缺失 [3]。参数化公式详细分解了影响并发环境下内存访问时间的各种因素，为有针对性的分析和优化提供了基础。

2.2 C-AMAT 在统一数据局部性、并发性和重叠性影响方面的作用

C-AMAT 模型通过将数据并发性整合到传统的 AMAT 框架中，实现了对数据局部性和并发性的统一 [5]。AMAT 主要通过命中延迟、缺失率和缺失代价来衡量数据局部性的影响，并假设内存访问是串行的 [4]。C-AMAT 通过包含命中并发度（C_h）和纯缺失并发度（C_{pm}）等参数扩展了这一模型，直接量化了并行访问的影响 [3]。命中并发度（C_h）可以由诸如多端口、多体或流水线缓存等架构特性贡献 [7]。纯缺失并发度（C_{pm}）可以通过非阻塞缓存和预取逻辑等机制来增强 [5]。“纯缺失”概念的重要性在于它能更准确地反映性能，因为它关注的是未被并发命中活动隐藏的缺失周期 [5]。只有纯缺失才导致内存停顿，因此减少纯缺失对于性能提升至关重要 [3]。C-AMAT 可以递归地应用于内存层次结构的多个层级，以分析每一层数据局部性和并发性的综合影响 [3]。一个层级的 C-AMAT 可以从其下一层级的 C-AMAT 推导出来 [7]。重叠性在 C-AMAT 的计算中被隐式地捕获，尤其是在纯缺失的定义以及以重叠模式计算 C-AMAT 的方式中 [3]。C-AMAT 与 AMAT 的差异突显了并发数据访问的贡献 [8]。C-AMAT 通过直接将并发性纳入内存访问时间计算中，实现了显著的统一。在 AMAT 中，内存访问时间主要受局部性因素影响。纯缺失的概念是这种统一的核心，因为它将并发性与缺失所造成的实际性能损失联系起来。C-AMAT 的递归特性允许对整个内存层次结构进行整体分析，捕捉每一层局部性和并发性之间的相互作用。

3 基于 C-AMAT 的分层性能匹配（LPM）算法

3.1 LPM 的定义与原理

分层性能匹配（LPM）是一种旨在通过匹配内存层次结构中相邻层级之间的数据请求速率和数据供给速率来优化整体内存系统性能并解决内存墙问题的方法和算法 [7]。其核心思想是优化每一层的性能，使其与直接位于其上层的请求紧密匹配 [8]。

LPM 的基本原理是通过平衡每一层的数据局部性、数据并发性和数据流的延迟隐藏来减少整体数据访问延迟 [8]。通过提高较高层级中命中和缺失之间的有效重叠，可以减轻较低层级对性能的影响 [8]。

LPM 的关键原理包括 [8]：

- **性能匹配**：优化每一层的性能以满足其上层的需求。
- **并发性与局部性**：在匹配过程中同时考虑数据访问并发性和局部性 [8]。
- **延迟隐藏**：利用并发性来增加命中和缺失之间的有效重叠，从而隐藏缺失的延迟 [8]。
- **纯缺失区分**：区分一般缺失和纯缺失，以便将优化工作集中在直接导致处理器停顿的缺失上 [8]。

分层性能匹配比率（Layered Performance Matching Ratios, LPMRs）作为量化相邻层级之间匹配程度的指标被引入 [8]。LPMR 是上层的数据请求速率与下层的数据供给速率之比 [8]。

LPM 提供了一种结构化的方法来优化整个内存层次结构，其重点在于层级之间的接口。与侧重于单个缓存级别的优化方法相比，这种方法能够实现更全面的优化。匹配数据请求和供给速率，并在并发性和局部性考虑的指导下进行，是其关键特点。

3.2 LPM 优化存储层次间性能匹配的具体方法

LPM 算法利用 C-AMAT 作为关键的分析工具，来量化相邻内存层级之间的性能不匹配 [7]。C-AMAT 提供了一种衡量每一层性能（考虑并发性的平均内存访问时间）的方法 [8]。分层性能匹配比率（LPMRs）的概念及其与 C-AMAT 和其他系统参数（如内存频率（f_{mem}）、完美缓存下的每指令周期数（CPI_{exe}）以及缺失率（MR））的直接关系被解释 [8]。对于一个三级层次结构，LPMR 可以用 C-AMAT 和这些参数来表示，量化一个层级的请求速率与下一层级的供给速率之间的不匹配 [8]。C-AMAT 的参数（命中时间、命中并发度、纯缺失率、纯缺失代价、纯缺失并发度）在每一层都提供了指导 LPM 内部优化策略的洞察 [8]。LPM 旨在优化这些参数以降低 C-AMAT 并改善匹配比率。LPM 使用从目标最小数据停顿时间和计算与内存访问重叠比率导出的 LPMR 阈值（T1 和 T2）来确定哪些层级需要优化 [8]。LPM 算法的迭代特性包括初始 LPMR 测量、与阈值比较、应用优化技术（硬件或软件）、更新指标以及重复该过程直到实现匹配 [3]。LPM 可以指导各种硬件（重配置）和软件（代码修改、调度）优化技术的选择和应用，以基于 C-AMAT 分析来提高识别出的瓶颈层级的数据局部性和并发性 [8]。LPM 的有效性通过其对 C-AMAT 的依赖性显著增强。C-AMAT 为识别性能瓶颈和衡量内存层级之间不匹配程度提供了定量的基础。C-AMAT 的参数为优化提供了具体的方向，而 LPMR 则为评估匹配过程的成功与否提供了明确的指标。这种集成使得 LPM 成为一种应用感知且动态适应的优化技术。

4 实验显示内存停顿时间减少 150 倍的研究

4.1 对声称 150 倍减少的研究进行分析

关于内存停顿时间减少 150 倍的说法，主要来源于对 Xian-He Sun 及其同事工作的分析 [7]。这一说法在 Sun 博士的一次讲座的摘要中明确提及 [9]，并在一个案例研究中进行了详细说明 [10]。具体实验细节在 [10] 和 [10] 中有所描述，该实验使用了 SPEC CPU 2006 套件中的 410.bwaves 基准程序。该基准程序以其内存密集型特性而闻名。仿真环境涉及 GEM5 和 DRAMSim2 仿真器的集成，

并增强了 C-AMAT 组件，以准确模拟并发内存访问行为。在 LPM 优化之前，初始内存停顿时间大于 60% [10]。在对可重构架构应用 LPM 算法并设置特定阈值（T1 = 1.52，T2 = 2.14）后，停顿时间显著降低至小于 1% [10]。这种从 >60% 到 <1% 的停顿时间减少意味着内存性能提高了 150 倍以上。据报告，整体执行时间加速为 2.5 倍（计算为 100/40，假设停顿时间是导致执行时间降至其潜在值的 40% 的主要因素）[10]。根据初始内存停顿百分比的不同，如果初始停顿为 70% 或 90%，则潜在的加速可能更高（高达 230 倍或 900 倍）[10]。这突出了 LPM 在高内存受限场景中的潜力。停顿时间的减少取决于具体的应用程序 [10]，这意味着 LPM 的有效性可能因应用程序的内存访问模式和并发特性而异。这种显著的减少归因于 LPM 实现的整体步调匹配优化，其中 LPM 控制每个内存层级的数据传输以匹配请求和供给速率，从而最大限度地减少内存墙效应 [10]。声称内存停顿时间减少 150 倍的实验结果为 LPM 算法的潜在有效性提供了有力的证据，尤其是在内存密集型应用中。使用广泛认可的基准程序和详细的仿真环境增加了这一说法的可信度。然而，重要的是要认识到，这是在特定条件下实现的，可能并非普遍适用。较高的初始停顿时间表明，该基准程序特别容易受到内存瓶颈的影响，这使得 LPM 的干预措施非常有效。

5 C-AMAT 模型和 LPM 算法的应用案例

5.1 计算机体系结构

在计算机体系结构领域，C-AMAT 作为一种精确的指标和现代内存系统的设计与优化工具发挥着重要作用 [3]。C-AMAT 有助于理解并发性和局部性对内存性能的影响。它可以用于评估不同的缓存设计（例如，多端口、多体、流水线、非阻塞）及其对命中和缺失并发度的贡献 [5]。LPM 可用于指导内存层次结构的设计，通过识别层级之间的性能不匹配并提出优化建议 [7]。LPM 可以帮助确定最佳的缓存大小、关联度和实现平衡性能的其他参数。[7] 提到 C-AMAT 可用于检测系统结构，以识别缓存命中并发度和缓存缺失并发度。C-AMAT 作为开发新的基于并发性的优化技术和利用现有基于局部性的优化技术的基础而受到重视 [7]。C-AMAT 和 LPM 为计算机架构师提供了强大的框架，用于设计能够有效处理现代应用程序日益增长的需求的内存系统。通过量化并发性的影响并提供一种平衡内存层级性能的方法，它们有助于创建更高效和可扩展的架构。

5.2 操作系统

C-AMAT 和 LPM 在操作系统层面的内存管理和调度方面也存在潜在的应用 [3]。操作系统可以利用 C-AMAT 来监控运行进程的内存访问模式和并发级别，从而做出更明智的内存分配和进程调度决策。LPM 原理可以指导操作系统管理跨越层次结构不同级别的内存资源，确保应用程序的数据访问需求与底层硬件能力相匹配。[7] 提到基于 C-AMAT 的在线重配置和智能调度是潜在的应用，这表明操作系统可以动态调整内存系统参数或调度策略。虽然片段中没有详细说明操作系统应用，但 C-AMAT 的并发感知特性和 LPM 的分层优化重点表明，它们在管理复杂内存层次结构和调度并发工作负载方面具有潜在的优势。未来在这一领域的研究可以探索如何将这些概念整合到操作系统内存管理策略中。

5.3 其他相关领域

C-AMAT 和 LPM 的适用性超出了传统的以 CPU 为中心的系统，扩展到了 GPGPU 内存性能评估领域，其中 GPGPU/C-AMAT 模型被提出，通过考虑 warp 级数据访问作为基本操作单元，更好地理解和量化 GPU 中的内存性能 [11]。该模型同时考虑了 warp 级的数据访问并发性和局部

性。C-AMAT 还用于高性能计算中分离内存系统（Disaggregated Memory Systems, DMS）的性能建模，其中 C-AMAT 被扩展到多核架构以测量 DMS 性能并识别应用程序-DMS 特性 [12]。[3] 提到 C-AMAT 在 FPGA 利用率和任务调度方面也具有潜在的应用。C-AMAT 在内存计算（Processing-in-Memory, PIM）架构的工作负载卸载决策中也发挥作用，其中纯缺失周期率（源于 C-AMAT 原理）被用作判断哪些代码块应在 PIM 端执行以获得更好性能（通过减少数据移动）的标准 [13]。C-AMAT 和 LPM 的多功能性使其能够应用于各种计算范式，有助于分析和优化 GPU、分离内存系统、FPGA 和 PIM 架构中的内存性能，这突显了它们在应对各种新兴计算模式下的内存墙挑战方面的相关性。

6 近年来关于 C-AMAT 和 LPM 的最新研究进展

6.1 C-AMAT 模型的改进与扩展

近年来，研究人员持续致力于增强和完善 C-AMAT 模型，以提高其准确性并扩展其在现代架构中的适用性 [5]。一种增强的 C-AMAT 模型被提出，该模型考虑了各个内存单元及其在内存架构设计中的拓扑结构，解决了原始模型自顶向下计算内存访问时间而不考虑单个单元贡献的局限性 [6]。FC-AMAT（Factor-Based C-AMAT）作为一种基于 C-AMAT 的分析模型被提出，该模型将内存系统划分为多个因素，并使用因素优先的 C-AMAT 来评估优化效果，旨在减少测量硬件开销，同时保持可接受的准确性 [6]。BC-AMAT（Blocked C-AMAT）被引入，它考虑了阻塞周期以提高内存评估的准确性，解决了原始模型中潜在的不准确性 [6]。C-AMAT 被扩展到多核架构，并应用于测量诸如 Argonne Cooley 计算机集群等平台上的 DMS 性能，探索分离内存的影响 [12]。一个旨在开发基于 C-AMAT 的先进内存架构性能建模和优化框架的持续项目，考虑了分层内存系统中特定缓存内存设备上的分裂和合并 [14]。最近的研究表明，增强 C-AMAT 模型以更好地捕捉现代复杂内存架构的细微之处是一个明显的趋势。这些改进侧重于解决与粒度、异构性、测量复杂性和准确性相关的局限性，表明持续努力使 C-AMAT 成为更强大和实用的内存系统分析工具。

6.2 LPM 算法的优化与新应用

近年来，关于增强 LPM 算法或识别新的应用场景的研究不断涌现 [6]。LPM 优化算法的开发旨在减少每个内存层级之间请求和供给速度的不匹配，从而缓解由此产生的数据停顿时间。该算法被设计为应用感知的，利用所需参数的在线测量 [8]。LPM 方法已通过代码修改（例如，提高缓存破坏性基准程序的性能）、重新配置系统硬件（在更少的步骤中为基准程序找到完美匹配）以及在异构平台上进行调度（在多项式时间内实现半最优调度）等方式得到应用 [8]。[10] 提到 LPM 在异构环境中的任务调度以及确定内存层次结构中的最佳层数方面具有潜在用途。并发感知数据访问局部性（Concurrency-aware data access Locality, CaL）作为局部性的扩展被引入，它考虑了并发性，并且其与 LPM 结合用于优化，在大数据基准程序中实现了显著的性能提升 [15]。LPM 研究的最新进展侧重于展示其在各种优化场景和计算平台上的实际适用性。算法本身的开发及其通过不同方法（软件、硬件、调度）的成功部署突显了其作为一种用于增强内存系统性能的多功能工具的潜力。与 CaL 等概念的联系进一步扩展了优化的可能性。

7 C-AMAT 和 LPM 与其他类似模型和算法的比较分析

7.1 C-AMAT 与传统内存访问时间模型（如 AMAT）的比较

C-AMAT 相对于 AMAT 等传统模型的关键区别和优势在于其能够处理并发内存访问，而 AMAT 则假设内存访问是串行的 [1]。C-AMAT 通过引入命中并发度（C_h）和缺失并发度（C_m 或 C_{pm}）参数以及纯缺失的概念，能够更细致、更准确地表示内存访问时间，因为它区分了导致停顿的缺失和那些被重叠的缺失 [3]。当没有数据并发性时，C-AMAT 会简化为传统的 AMAT 公式 [3]，这表明了它的通用性。

表 1: AMAT 与 C-AMAT 模型比较

参数	AMAT	C-AMAT
命中时间（延迟）	H	H
缺失率	MR	–
缺失代价	AMP	–
命中并发度	隐式为 1（串行）	C_h
缺失并发度	隐式为 1（串行）	C_m 或 C_{pm}
纯缺失率	不考虑	pMR
纯平均缺失代价	不考虑	pAMP
公式	$H + (\text{MR} \times \text{AMP})$	$H/C_h + (\text{pMR} \times \text{pAMP}/C_{pm})$

7.2 LPM 与其他内存层次结构优化算法的比较

LPM 与其他内存层次结构优化技术（如缓存替换策略（例如，LRU、FIFO [16, 17]）、预取策略（[18]）和内存分区（[14]））进行了比较。LPM 的独特之处在于其关注于基于数据局部性和关键的数据访问并发性来匹配内存层次结构相邻层级之间的性能，并使用 C-AMAT 作为指导指标 [7]。与许多侧重于优化单个层级或方面的其他技术不同，LPM 采取了更全面的端到端视角。LPM 可以根据识别出的性能不匹配，潜在地指导其他优化技术在特定层级的应用，充当更高级别的控制机制 [8]。

7.3 C-AMAT 与 MLP（内存级并行）的比较

C-AMAT 和 MLP（内存级并行）作为性能指标的关系进行了分析 [7]。MLP 衡量的是当至少存在一个未完成的长延迟主内存访问时，未完成的长延迟主内存访问的平均数量 [7]。它量化了在访问主内存时实现的并发程度。C-AMAT 是一个分析模型和性能指标，它考虑了整个内存层次结构中的局部性和并发性 [7]。MLP 主要关注主内存级别的并发性。MLP 在其基本定义中没有明确考虑缓存层次结构中的数据局部性，而 C-AMAT 则将局部性和并发性都融入其公式中 [7]。每周期访问次数（APC）是 C-AMAT 的倒数，可以被认为比 MLP 更全面的指标，因为它反映了在每个级别考虑局部性和并发性的整体内存系统吞吐量 [3]。

8 研究社区对 C-AMAT 模型和 LPM 算法的评价和反馈

8.1 C-AMAT 的优势与局限性

研究文献普遍认为 C-AMAT 模型具有显著的优势 [5]，包括其在建模具有并发性的现代内存系统方面的准确性，将数据局部性和并发性的影响统一在一个公式中，以及作为开发新的基于并发性的优化技术的基础。它还可以递归地应用于内存层次结构的各个层级。然而，C-AMAT 也存在一些局限性 [6]。原始模型可能缺乏处理异构内存技术和不同系统拓扑所需的粒度和灵活性。获得计算所需参数（如纯缺失周期）的难度，以及直接测量带来的潜在硬件开销也是一个问题。此外，一些扩展模型可能没有充分考虑较低级别缓存的串行访问特性。

8.2 LPM 的优势与局限性

据报告，LPM 算法具有多项优势 [8]，包括其优化内存层次结构性能的系统方法，同时考虑数据局部性和并发性，显著减少数据停顿时间的潜力，以及通过代码修改、硬件重新配置和调度等多种方式的应用。虽然提供的片段主要强调了 LPM 在内存系统方面的优势，但值得注意的是，“LPM”一词也在其他领域使用（例如，统计学中的线性概率模型 [19, 20]，网络中的最长前缀匹配 [21]，工业控制中的回路性能管理器），这些其他的“LPM”有其自身的局限性 [19]。这些片段没有提供对 LPM 算法在 C-AMAT 驱动的内存优化方面的广泛批评。

8.3 总体社区评价和未来发展方向

研究社区普遍认为 C-AMAT 和 LPM 是理解和优化并发计算时代内存性能的重要工具 [7]。未来的研究可能侧重于完善这些模型，开发实用的测量技术，探索在新兴架构（如 PIM 和异构系统）中的应用，并可能将它们与机器学习相结合以实现自适应优化 [22]。

9 开源的 C-AMAT 模型或 LPM 算法的实现

9.1 开源 C-AMAT 实现的可用性

虽然片段中提到在仿真器中添加了 C-AMAT 组件（[10]），但在提供的材料中没有明确提到专门针对 C-AMAT 的独立开源实现（[23]）。然而，C-AMAT 的概念很可能在内存系统仿真器中有所实现。[24] 展示了一个开源缓存仿真器，该仿真器允许配置缓存级别和预取机制，这可以作为实现和评估 C-AMAT 原理的平台。[25] 介绍了 gem5，这是一个广泛使用的开源计算机架构仿真器，能够进行详细的内存系统建模，这表明 C-AMAT 可以在此框架内实现和使用。基于这些片段，一个专门的、独立的 C-AMAT 开源实现可能并不作为一个单独的项目广泛存在。然而，在诸如 gem5 等开源仿真器中建模并发内存访问的能力为研究人员使用和评估 C-AMAT 提供了必要的工具。

9.2 开源 LPM 算法实现的可用性

搜索开源 LPM 算法的实现。这些片段没有明确提到针对内存层次结构优化的 LPM 算法的特定开源实现。[26], [27], [21] 指的是不同上下文中的“LPM”（网络中的最长前缀匹配，图形学中的 HDR 映射库）。基于提供的片段，专门用于 C-AMAT 驱动的内存优化的 LPM 算法的开源实现似乎并不存在。研究人员可能在其仿真环境或特定研究项目中实现该算法。

9.3 相关的性能评估工具和数据集

识别可用于评估 C-AMAT 和 LPM 的开源仿真器（例如，GEM5 [10]，DRAMSim2 [10]）和性能分析工具（[28]）。GEM5 是一个被广泛采用的开源仿真器，支持多种指令集架构，并提供 CPU、GPU 和内存系统的详细建模，使其适用于 C-AMAT 和 LPM 的评估 [29]。DRAMSim2 是一个周期精确的 DRAM 仿真器，可以与 GEM5 等仿真器集成，以提供详细的片外内存行为，这对于全面的内存层次结构分析至关重要 [10]。各种性能分析工具，如 gprof、Valgrind、TAU[30, 31] 和特定于平台的工具（例如，AMD uProf [28]），可用于收集性能数据（例如，缓存未命中、内存访问时间），这些数据可以使用 C-AMAT 原理进行分析，或用于评估 LPM 实现的有效性。[24] 突出显示了一个开源缓存仿真器，该仿真器具有可配置的参数和高级预取功能，这对于研究并发性对 C-AMAT 建模的缓存性能的影响可能很有价值。研究社区可以使用多种强大的开源工具和仿真器，这些工具可以用于实现、评估和进一步开发 C-AMAT 和 LPM。这些工具为在并发环境中进行深入的内存系统性能研究提供了必要的灵活性和细节。

10 结论

并发平均内存访问时间（C-AMAT）模型代表了内存性能建模方面的一个重要进步，它通过纳入并发性和重叠性来解决传统指标（如 AMAT）在现代内存系统中的局限性。分层性能匹配（LPM）算法作为一种系统性的方法，通过匹配内存层次结构中每一层的请求和供给速率，为优化内存性能提供了一个有前景的途径，并由并发感知的 C-AMAT 指标指导。实验验证，特别是报告的 150 倍内存停顿时间减少，突显了 LPM 在显著提高内存受限应用程序性能方面的潜力。C-AMAT 和 LPM 已证明在各种计算领域（包括 CPU、GPU、分离内存、FPGA 和 PIM 架构）具有适用性。持续的研究致力于完善这些概念，解决其局限性，并探索新的应用，这表明这是一个充满活力和积极的研究领域。这些概念对于解决持续存在的内存墙问题至关重要，内存墙仍然是实现现代计算系统更高性能的关键瓶颈。通过有效管理并发性和优化内存层次结构，C-AMAT 和 LPM 有助于开发更高效和可扩展的系统。未来的研究方向和未解决的挑战包括：开发更高效、更准确的硬件和软件机制来测量实际系统中的 C-AMAT 参数；进一步探索 LPM 在新兴内存技术（例如，非易失性内存、高带宽内存）和日益异构的计算架构中的适用性和有效性；研究使用机器学习和人工智能技术基于 C-AMAT 和 LPM 原理动态地适应和优化内存系统性能；开发更标准化的基准和评估方法，专门用于评估并发感知内存优化技术（如 C-AMAT 和 LPM）的性能优势；创建更用户友好和易于访问的工具和框架，使研究人员和工程师能够轻松地在其工作中实现和使用 C-AMAT 和 LPM。C-AMAT 和 LPM 代表着朝着更全面地理解和有效优化面对日益增长的并发需求而产生的内存系统性能迈出的关键步骤。虽然已经取得了显著的进展，但持续的研究和开发对于克服剩余的挑战并充分实现这些概念在解决内存墙和推动下一代高性能计算系统方面的潜力至关重要。

References

[1] ResearchGate. *An example of C-AMAT and pure miss | Download Scientific Diagram*. https://www.researchgate.net/figure/An-example-of-C-AMAT-and-pure-miss_fig1_325254598. Accessed: April 15, 2025.

[2] B. Feng et al. *Proceedings of the 2022 Winter Simulation Conference*. Tech. rep. 10423827. Accessed: April 15, 2025. Also cited as ref 3. National Science Foundation (NSF), 2022.

[3] Computer Science Department, Illinois Institute of Technology. *Concurrent Average Memory Access Time (C-AMAT)*. http://cs.iit.edu/~scs/research/c-amat/c-amat.html. Accessed: April 15, 2025. Also cited as ref 4.

[4] Wikipedia. *Average memory access time*. https://en.wikipedia.org/wiki/Average_memory_access_time. Accessed: April 15, 2025. Also cited as ref 5.

[5] ResearchGate. *Illustration of C-AMAT model and Pure Miss*. https://www.researchgate.net/figure/llustration-of-C-AMAT-model-and-Pure-Miss_fig1_369520504. Accessed: April 15, 2025.

[6] B. Feng et al. *Proceedings of the 2022 Winter Simulation Conference*. https://par.nsf.gov/servlets/purl/10423827. Accessed: April 15, 2025. Also cited as ref 8. 2022.

[7] SlideServe. *Optimizing Data Access Speed with C-AMAT Metric*. https://www.slideserve.com/plorna/c-amat-concurrent-average-memory-access-time-powerpoint-ppt-presentation. Accessed: April 15, 2025. Likely based on work by X.-H. Sun et al. Also cited as ref 7.

[8] Yuhang Liu and Xian-He Sun. "LPM: Concurrency-Driven Layered Performance Matching". In: *Proceedings of the ACM International Conference on Supercomputing (ICS '24)*. Accessed: April 15, 2025. Also cited as ref 11. 2024.

[9] 东莞理工学院 (Dongguan University of Technology). 美国伊利诺伊理工大学杰出教授、*IEEE Fellow* 孙贤和特邀讲座. https://dgut.edu.cn/info/1028/17619.htm. Accessed: April 15, 2025. Mentions 150x reduction. Also cited as ref 24. 2024.

[10] Xian-He Sun. *Memory Wall Problem and Optimization*. http://www.cs.iit.edu/~scs/assets/files/isc-memory.pdf. Accessed: April 15, 2025. Likely presentation slides. Also cited as ref 21.

[11] Z. Zhang, B. Feng, and X.-H. Sun. "Evaluating GPGPU Memory Performance Through the C-AMAT Model". In: *Proceedings of the Workshop on Memory Centric High Performance Computing (MCHPC'17)*. Accessed: April 15, 2025. Also cited as ref 9. 2017.

[12] Hao Sun et al. "Performance Modeling and Evaluation of a Production Disaggregated Memory System". In: *Proceedings of the 11th International Workshop on Memory Systems (MEMSYS '20)*. Accessed: April 15, 2025. Also cited as ref 6. 2020, pp. 235–246.

[13] Yijia Wang, Kai Lu, and Xian-He Sun. "CoPIM: A Concurrency-aware PIM Workload Offloading Architecture for Graph Applications". In: *Proceedings of the 50th International Conference on Parallel Processing (ICPP '21)*. Accessed: April 15, 2025. Also cited as ref 29. 2021.

[14] Gnosis Research Center, Illinois Institute of Technology. *Optimization of Memory Architectures: A Foundation Approach*. https://grc.iit.edu/research/projects/optmem/. Accessed: April 15, 2025. Also cited as ref 16.

[15] Yuhang Liu and Xian-He Sun. "CaL: Extending Data Locality to Consider Concurrency for Performance Optimization". In: *Proceedings of the IEEE International Parallel and Distributed Processing Symposium (IPDPS '24)*. Accessed: April 15, 2025. Also cited as ref 15. 2024.

[16] Wikipedia. *Cache replacement policies*. https://en.wikipedia.org/wiki/Cache_replacement_policies. Accessed: April 15, 2025. Also cited as ref 33.

[17] Fiveable Library. *Cache Replacement and Write Policies | Advanced Computer Architecture Class Notes*. https://library.fiveable.me/advanced-computer-architecture/unit-7/cache-replacement-write-policies/study-guide/OYpe2aD5JCvYaxRh. Accessed: April 15, 2025. Also cited as ref 32.

[18] Author(s) Unknown. "Memory Hierarchy Optimization Strategies for High- Performance Computing Architectures". In: *ResearchGate Publication* (2024). Accessed: April 15, 2025. Also cited as ref 27. DOI: 10.13140/RG.2.2.13868.23684.

[19] Dummies.com. *3 Main Linear Probability Model (LPM) Problems*. https://www.dummies.com/article/business-careers-money/business/economics/3-main-linear-probability-model-lpm-problems-156466/. Accessed: April 15, 2025. Different LPM context. Also cited as ref 44.

[20] Wikipedia. *Linear probability model*. https://en.wikipedia.org/wiki/Linear_probability_model. Accessed: April 15, 2025. Different LPM context. Also cited as ref 46.

[21] Mindaugas Rasiukevicius (rmind). *rmind/liblpm: Longest Prefix Match (LPM) library*. https://github.com/rmind/liblpm. Accessed: April 15, 2025. Different LPM (networking). Also cited as ref 61.

[22] MIT Media Lab. *Overview ‹ Large Population Models*. https://www.media.mit.edu/projects/ai-lpm/overview/. Accessed: April 15, 2025. Different LPM context (Large Population Models). Also cited as ref 50.

[23] SourceForge. *Best Open Source C Software*. https://sourceforge.net/directory/c/. Accessed: April 15, 2025. General open source listing. Also cited as ref 56.

[24] Mudit Bhargava. *muditbhargava66/CacheSimulator: A high-performance cache and memory hierarchy simulator...* https://github.com/muditbhargava66/CacheSimulator. Accessed: April 15, 2025. Also cited as ref 38.

[25] Vayavya Labs. *Introducing gem5 : An Open-Source Computer Architecture Simulator*. https://vayavyalabs.com/blogs/introducing-gem5-an-open-source-computer-architecture-simulator/. Accessed: April 15, 2025. Also cited as ref 58.

[26] Cookiengineer. *lpm package - github.com/cookiengineer/lpm*. https://pkg.go.dev/github.com/cookiengineer/lpm. Accessed: April 15, 2025. Different LPM (networking). Also cited as ref 59.

[27] AMD GPUOpen. *AMD FidelityFX™ Luminance Preserving Mapper (HDR Mapper)*. https://gpuopen.com/fidelityfx-lpm/. Accessed: April 15, 2025. Different LPM (graphics). Also cited as ref 60.

[28] Rajesh S. *Tools and Utilities | Computer Engineering Resources*. https://rajesh-s.gitbook.io/compengg/area_specific/tools_and_utils. Accessed: April 15, 2025. Lists performance tools. Also cited as ref 63.

[29] Luca Nardi et al. "HERMES: High-Performance RISC-V Memory Hierarchy for ML Workloads". Version v2. In: *arXiv preprint arXiv:2503.13064* (2025). Accessed: April 15, 2025. Also cited as ref 37. arXiv: 2503.13064v2.

[30] Wikipedia. *List of performance analysis tools*. https://en.wikipedia.org/wiki/List_of_performance_analysis_tools. Accessed: April 15, 2025. Also cited as ref 64.

[31] Fiveable Library. *Performance Profiling and Analysis Tools | Parallel and Distributed Computing Class Notes*. https://library.fiveable.me/parallel-and-distributed-computing/unit-8/performance-profiling-analysis-tools/study-guide/uhgn9Yt1Be8zOugr. Accessed: April 15, 2025. Also cited as ref 65.