重庆大学《计算机系统结构》
2022-2023 学年第一学期期末考核

姓名：_____陈鹏宇____　　学号：___20204227____

CPU 缓存技术研究综述

摘　要：随着中央处理器（CPU）的迅速发展，CPU 处理数据的速率要远高于主存系统。为调和 CPU 和内存之间速度的差异，CPU 缓存（以下简称缓存）技术得以提出，且在利用局部性原理的基础上，使内存层次结构高效工作。近年来，在命中率、延迟、速度、替换策略和能量消耗等方面上，已经取得了各种进步来提高缓存的性能。本文主要从 CPU 缓存技术的研究背景和研究意义出发，提出其面临的挑战，并以减少 AMAT 为目的提出缓存技术优化的方法，最后进行总结和展望。

关键词：缓存技术；CPU 缓存；存储；优化；

A Survey of CPU Cache Technology Research

Abstract: With the rapid development of the central processing unit (CPU), the rate at which the CPU processes data is much higher than that of the main memory system. In order to reconcile the difference in speed between CPU and memory, CPU cache (hereinafter referred to as cache) technology is proposed, and based on the principle of locality, the memory hierarchy can work efficiently. In recent years, various advances have been made to improve the performance of caches in terms of hit rate, latency, speed, replacement strategy, and energy consumption. This paper mainly starts from the research background and research significance of CPU cache technology, puts forward the challenges it faces, and proposes the method of caching technology optimization for the purpose of reducing AMAT, and finally summarizes and looks forward to it.

Keyword: Cache technology; CPU cache; storage; optimization;

目　录

1　CPU 缓存技术研究背景

1.1 缓存技术研究背景

CPU 缓存的概念由来已久，可以追溯到计算的早期。缓存的第一个例子是在 1950 年代开发的 UNIVAC 1103 中使用的缓冲存储器。该系统使用少量快速内存来存储在主内存和 CPU 之间传输的数据。

20 世纪 60 年代，研究人员开始更系统地研究缓存的使用，作为提高计算机性能的一种方式。Edson de Castro 是这方面的早期研究人员之一，他在 DEC PDP-6[1] 计算机的设计中提出了缓存的使用。De Castro 认为，通过将频繁访问的数据存储在少量快速内存中，CPU 可以更快地访问这些数据并更有效地处理它。

20 世纪 70 年代，随着计算机系统变得越来越复杂，CPU 的性能越来越受到内存系统速度的限制，缓存的使用变得更加广泛。研究人员开始更详细地研究缓存的设计，重点关注缓存的大小、缓存的关联性和缓存替换策略等因素。

20 世纪 80 年代，静态随机存取存储器(SRAM)和动态随机存取存储器(DRAM)等高速内存技术的发展使得构建更大、更复杂的缓存系统成为可能。研究人员继续研究缓存的设计以及设计缓存所涉及的权衡，例如性能与功耗之间的权衡或缓存大小与缓存关联性之间的权衡。

20 世纪 90 年代，缓存的使用变得更加重要，因为 CPU 速度的增长开始快于主内存的速度。这导致了多级缓存系统的发展，该系统使用多级缓存以不同的速度和大小存储数据。研究人员继续研究缓存的设计和设计缓存所涉及的权衡，以及缓存对计算机整体架构的影响。

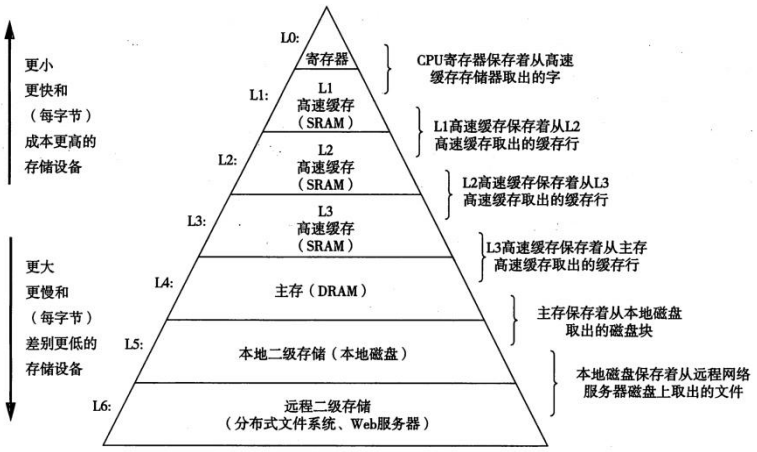

图 1 存储器层次结构图

21 世纪，缓存的使用在广泛的应用程序中变得越来越重要，研究人员继续研究缓存的设计和设计缓存所涉及的权衡，以及缓存对计算机整体架构的影响。此外，随着计算机的能耗成为越来越重要的关注点，研究人员开始研究缓存对计算机能效的影响。

随着计算机系统变得越来越复杂，CPU 的性能仍然受到内存系统速度的限制[2]，对 CPU 缓存的研究仍然是一个活跃的研究领域。研究人员继续研究缓存的设计和设计缓存所涉及的权衡[3]，以及缓存对计算机整体架构和计算机能效的影响。

1.2 缓存技术相关概念

本节主要介绍缓存技术方向相关的机制和术语，如局部性原理，cache 的基本单位，映射机制，性能评估等。

1.2.1 局部性原理

局部性原理作为 Cache 原理的基础，即处理器在访问数据过程中、在短暂的时间内访问重复，基于本质，一些数据或位置具有比较大的访问几率；可将其划分成以下类型：

- 时间局部性：如此次访问某数据，不久将存在再访问的可能性。
- 空间局部性：如访问某位置数据，则存在访问相邻数据的可能性。

1.2.2 Cache 的基本单位--cache line

Cache line 即最小缓存单位；以局部空间思想为基础，内存与缓存间数据的移动并非逐字节实现。当一个指令或数据被访问后，与它相邻地址的数据有很大概率也会被访问，所以缓存在读取或写回内存数据时通常是一块一块地进行读写，可以提高缓存命中率。

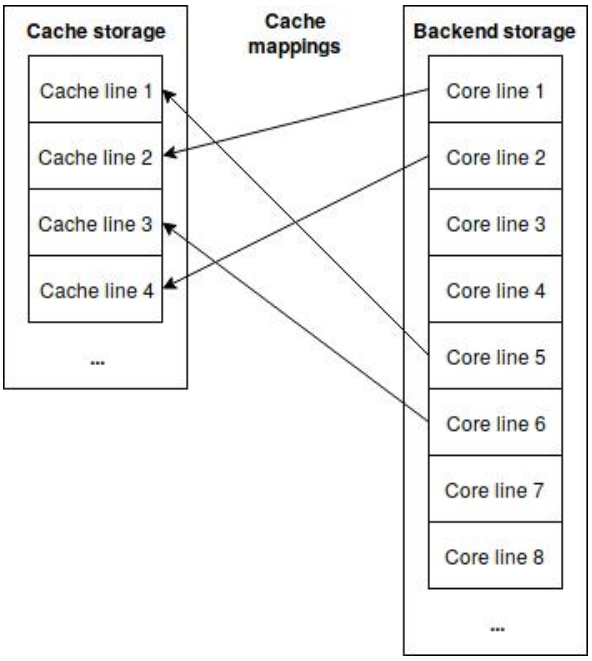

图 2 Cache line 示意图

1.2.3 映射机制

读或写缓存数据时，CPU 需知晓所访问内存数据对应的 Cache 位置，于是牵涉到主存与 Cache 地址的映射，包括以下类型：

- 全相联映射：许可内存块映射至任一 Cache line 上

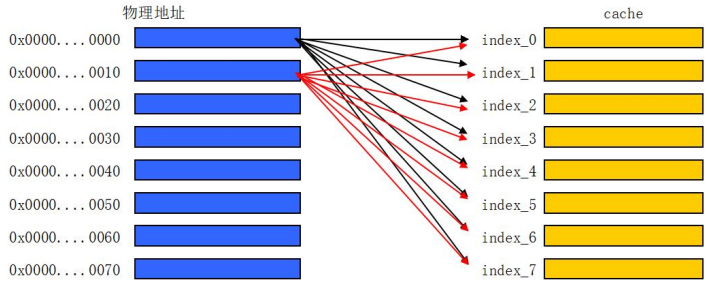

图 3 全相联

- 直接映射：内存块和 Cache line 之间建立固定的映射关系，一个内存块仅能映射到同一个 Cache line 上

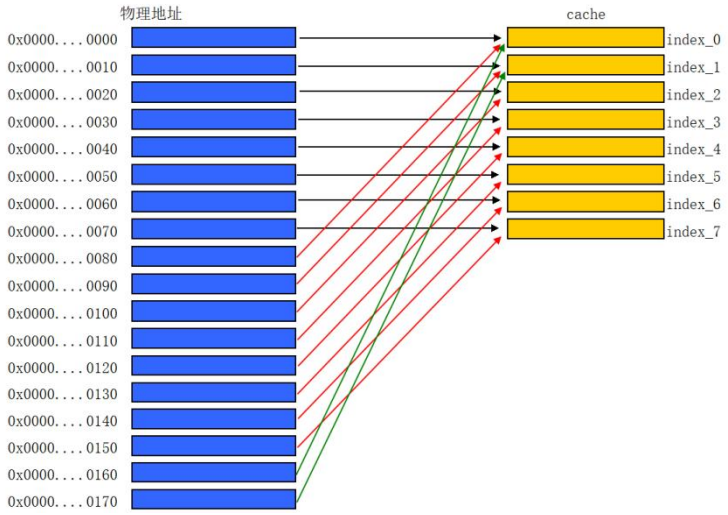

图 4 直接映射

- 组相联映射：上述两种方法的折中，可划分成多组，各内存块和某分组间构建固定映射的关系，许可内存块映射向组中任意一个 Cache line 内存贮缓存。

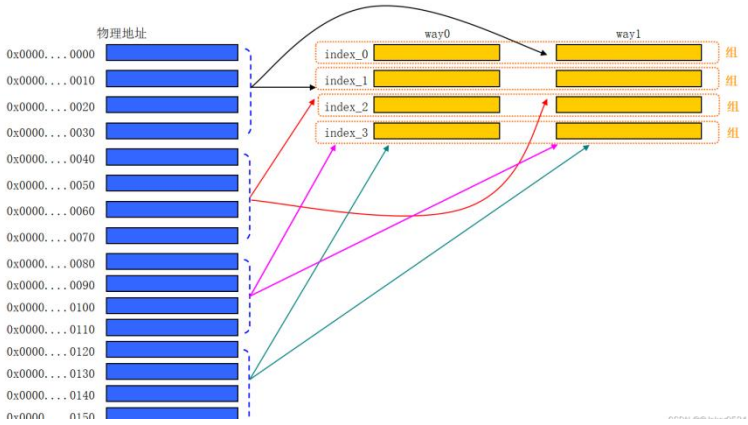

图 5 两路组相联

1.2.4 性能评估

评估缓存的性能主要采用 AMAT（平均内存访问时间）这一指标。其计算公式为：

$$\mathrm{AMAT} = \mathrm{T_{hit}} + \mathrm{MR} \times \mathrm{MP}$$

其中，T_{hit}为命中时延，从缓冲块定位，标签对比，直到数据传回 CPU 所需的时间；MR 为缺失率，即 Cache 访问缺失次数占总访问次数的比例；MP 为缺失代价，判定 Cache 缺失后，从内存中定位数据并更新缓存所需的时间。

为更好理解缺失率的原因，从而更好设计缓存，通过 3C 模式把上述场景划分成以下类型：

- 强制（Compulsory）：首次访问某数据块，缓存不包含此块，向缓存中调入；此缺失不能规避。
- 容量（Capacity）：如缓存没有涵盖程序运行的全部数据块，由于一些块被摒弃后再进行调入，于是缺少容量。
- 冲突（Conflict）：假如块放置方案并非全相联，多个块映射于某块组，针对不同块访问混合，将有可能摒弃某块且再被调入其中，于是导致缺失冲突现象发生。

本节仅介绍上述我认为比较重要的概念，由于篇幅有限，其余较细节部分就不在赘述。

1.3 缓存技术研究意义

结合研究背景和相关概念，我总结出以下缓存技术的研究意义：

- 提高计算机的能效：研究人员研究了通过使用低功耗内存技术或使用更节能的缓存设计来降低缓存功耗的方法。
- 对计算机内存层次的影响：缓存是内存层次结构中不可或缺的一部分，它决定了数据在计算机中的存储和访问方式。研究人员研究了缓存对计算机整体架构的影响，并探索了一系列提高缓存性能[4]和整体内存层次结构的方法。
- 专用应用程序：研究人员研究了缓存在专用应用程序中的使用，例如嵌入式系统[5]或分布式系统[6]，其中缓存对系统的性能和可靠性至关重要。
- 设计和优化挑战：研究人员研究了为现代计算机系统设计和优化缓存的挑战，包括使用机器学习[7]和优化技术为特定应用程序或工作负载设计和优化缓存。

随着计算机系统的不断发展，CPU 的性能仍然受到内存系统速度的限制，对 CPU 缓存的研究会继续成为一个活跃而重要的研究领域。

2 CPU 缓存技术的挑战

2.1 性能和功耗之间的权衡

CPU 缓存设计的主要挑战之一是找到提高缓存性能同时最大限度降低功耗[8]的方法。这在现代计算机系统中尤为重要，因为缓存的能耗会对系统的整体能效产生重大影响。研究人员探索了一系列解决这一挑战的方法，包括使用低功耗内存技术[9]和开发更节能的缓存设计。

2.2 性能和成本之间的权衡

CPU 缓存设计的另一个挑战是找到提高缓存性能[10]的方法，同时最大限度地降低缓存的总体成本。缓存的成本会对计算机系统的总体成本产生较大影响。研究人员探索了一系列解决这一挑战的方法，包括使用低成本内存技术和开发更具成本效益的缓存设计。

2.3 现代计算机系统发展

现代计算机系统越来越复杂，这使得为这些系统设计和优化缓存变得困难。研究人员已经探索了一系列方法[11]来应对这一挑战，包括使用机器学习和优化技术来设计和优化特定应用程序或工作负载的缓存。

2.4 缓存一致性

缓存一致性是系统的属性，其中系统中的所有缓存都包含一致的数据。在具有多个缓存的系统或具有共享数据的处理器的系统中，保持缓存一致性可能是一个挑战[12]。研究人员研究了高速缓存一致性，并开发了在这些系统中保持高速缓存一致性的技术。

3 CPU 缓存技术的优化

由于篇幅有限，优化方法没有针对上一节所提及的挑战。本节主要介绍了以减少 AMAT 为目的的缓存技术优化方法。由 AMAT 的计算公式可知，可基于以下三个层面优化缓存；但由于三者间的内在性质，某些技术可能无法同时对三者进行优化，也因此造成了性能、功耗和成本之间的权衡[13]。

3.1 降低缺失率

针对缺失率的降低，有许多比较简单的方法，如使用更大的数据块、使用更大的缓存或者使用多路组相联，但副作用也比较明显，如更高的缺失代价、更复杂的电路逻辑、更长的命中时延、更高的成本和功耗等。

为更好的权衡性能和成本，以 FIFO（先入先出算法）、LRU（最近最久未使用算法）、LFU（最近最少使用算法）为基础的缓存替换算法成为降低缺失率的主要途径。

Aamer Jaleel 等[14]提出了 LRU 插入策略（LIP），从而降低缺失率方向，使缓存不被抖动得以保障，同时为具备循环引用模式的程序提高命中率且维系适宜值；提出双峰插入策略（BIP）且增强 LIP，使工作集的改变得以适应；提出动态插入策略（DIP）。

Kathlene Morales 等[15]对分段 LRU 进行完善改进，根据有效分割率固定受保护段的数量，并增加受保护段，以及实施选择性缓存，实现更多有效驱逐，防止死块进入缓存。

Gerhard Hasslinger 等[16]将 LRU 和 LFU 算法结合起来，该策略可实现的命中率显示随着对请求模式的利用增加而提高，但同时计算量也在增加。

Mazen Kharbutli 等[17]提出一种新的基于计数器的方法来处理高度关联的缓存中 LRU 性能低的问题。而 Yee Ming Chung 等[18]在前者基础上，引入重用频率（MRU）提出一种末级缓存（LLC）替换策略，相较 LRU 有更好的性能。

Gurjit Kaur 等提出基于 ML 的缓存替换算法[19]，他们考虑到机器学习和缓存替换算法都是基于过去数据的，具有相似的预测性质。通过不断的参数训练，不断逼近 OPT 算法性能，从而提高命中率。

3.2 缩短命中时延

针对命中时延的缩短，最直接的方法是使用小而简单的缓存。一方面来说，电路延迟很大程度取决于存储芯片的大小，较小容量能保证最短的访问周期，另一方面，采取直接映射等简单缓存，对比于组相联映射，能够使命中时延得以缩减。但是，代价是命中率的降低。

20 世纪 90 年代，主要的解决办法是路预测和追踪缓存。

Bryan Black 等[20]提出基于块的追踪缓存，可以通过更有效的跟踪存储实现更高的 IPC 性能。基于块的追踪缓存在基本块级别重命名提取地址，并将对其的块存储在块缓存中，追踪是通过追踪表中的块指针访问复制块缓存来构建的。

Koji Inoue 等[21]提出了一种使用路径预测来实现集合关联缓存，通过仅访问预测的单个缓存路而不是访问一组中的所有路，可以降低时延和能量消耗。

随着硬件技术的不断发展，现如今通常采取改变缓存架构来实现命中时延的降低。

Pai Chen 等[22]观察到两路组相联提供了最大的命中率增加，提出了一种同时优化命中率和命中时延的两路组相联缓存（SODA-cache）。为减少命中时延，他们提出了 SRAM 中的 WayLocator 缓存，以快速确定 DRAM 缓存中请求的数据位置，而无需访问 DRAM 缓存中的标签。有效地利用了 LH-Cache 和 Ally-Cache 的最佳方面，避免了它们的局限性，并实现了最佳平均访问延迟和加权 IPC。

Nagendra Gulur 等[23]提出双模式 DRAM 缓存，不仅能提高缓存命中率，通过利用堆叠式 DRAM 组织提供的巨大内部带宽，可以实现对标签和数据的高效并发访问，以减少命中时延。

3.3 降低缺失代价

采用多级缓存[24]不仅能减少命中时延，也能降低缺失代价。常见的关键字优先和提前重启动也能很好在最小化 CPU 停顿的情况下，达到降低缺失代价的效果。合并写缓存区通过在写入内存时采用缓冲多字写入，提高写效率从而降低缺失代价。

肖侬等[25]观察到替换算法忽视角缓存失效开销的减少环节，提出一种基于顺序检测的双队列缓存替换算法，能优先淘汰缓存中的顺序页面，保留随机页面。根据结果显示，能减少访存次数，显著降低缺失代价。

4　总结与展望

本节归纳优化技术，同时展望发展 CPU 缓存技术的美好前景。

4.1 总结

在降低缺失率方向，国内外学者在缓存替换策略设计上进行了详细分析。整体上都通过改良或结合 LRU 算法来达到更好的命中率，间接上说明了 LRU 是最具有拓展性的缓存替换策略[26]。但相关研究表明，缓存替换算法都将重点放于提高命中率上，甚至会对缓存时延造成负面影响。随着机器学习领域的不断扩展，将其与缓存替换机制结合可能是未来发展的趋势。但对于目标的选择，不能仅以高命中率为目的，更应该是三者的权衡，以此追求更高的缓存性能。

对于降低缺失代价和缩短命中时延，本质上都属于缓存的延迟，无论是读写命中还是缺失。国内外学者主要通过改变缓存架构和配置达到降低延迟的效果。对于缓存配置参数的设定，可以通过数学建模的方法，对程序进行追踪和静态分析，找出不同行为应用下最佳性能的缓存配置。对缓存进行可重构设计，在运行过程中根据不同的行为模式动态选择配置。

以上所有优化技术都没有绝对的优劣之分，在优化项中，相互之间可能存在制约关系，但又相互联系。在对缓存进行设计时需从多个方面权衡考虑，而不仅仅是追求某一方向的极致优化。

4.2 展望

Phil Karlton 曾说过*"There are only two hard things in Computer Science: cache invalidation and naming things"*。缓存技术的创新和发展是 CPU 高性能工作基础，解决缓存带来的问题和挑战会为计算机系统发展提供强有力的支持，基于本文的分析，我们对 CPU 缓存优化技术的发展前景进行展望：

持续关注性能、功耗、成本三者的权衡关系，找到提高缓存性能同时最大限度降低功耗和成本的方法。随着计算机系统变得更加节能和功率受限，研究人员可能会继续专注于开发低功率缓存技术和设计。

从多个新角度切入对缓存进行优化，如安全性[27]、可靠性[28]以及缓存一致性[29]等。

受新兴技术的影响，非易失性存储器、3D 存储器和神经形态计算等新技术的出现，很可能对缓存的设计和运行产生重大影响。研究人员可能会研究这些技术在缓存中的使用，并探索它们的潜在优势和挑战。

选择更合理的动态缓存配置选择机制[30]，针对不同的应用程序，选择具有针对性的缓存配置。

参 考 文 献

[1] Bell et al. 1978, pp. 487, 489: "The project from which the PDP-6, DECsystem-10, and DECSYSTEM-20 series of scientific, timeshared computers evolved began in the spring of 1963 and continued with the delivery of a PDP-6 in the summer of 1964.".

[2] R. Stacpoole and T. Jamil, "Cache memories," in IEEE Potentials, vol. 19, no. 2, pp. 24-29, April-May 2000, doi: 10.1109/45.839642.

[3] P. Chen, J. Yue, X. Liao and H. Jin, "Trade-off Between Hit Rate and Hit Latency for Optimizing DRAM Cache," in IEEE Transactions on Emerging Topics in Computing, vol. 9, no. 1, pp. 55-64, 1 Jan.-March 2021, doi: 10.1109/TETC.2018.2800721.

[4] S. Prybylski, M. Horowitz, and J. Hennessy. 1988. Performance tradeoffs in cache design. SIGARCH Comput. Archit. News 16, 2 (May 1988), 290–298. https://doi.org/10.1145/633625.52433

[5] 黄锦灏,丁钰真,肖亮,沈志荣,朱珍民.一种基于强化学习的嵌入式系统抗拒绝服务攻击的缓存调度方案[J].计算机科学,2020,47(07):282-286.

[6] 李宁,张轶昀.一种分布式微服务架构系统缓存解决方案[J].电脑知识与技术,2020,16(36):73-74.DOI:10.14004/j.cnki.ckt.2020.3687.

[7] Zhan Shi, Xiangru Huang, Akanksha Jain, and Calvin Lin. 2019. Applying Deep Learning to the Cache Replacement Problem. In Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO '52). Association for Computing Machinery, New York, NY, USA, 413–425. https://doi.org/10.1145/3352460.3358319

[8] Mittal S. A survey of architectural techniques for improving cache power efficiency[J]. Sustainable Computing: Informatics and Systems, 2014, 4(1): 33-43.

[9] Mittal S, Verma G, Kaushik B, et al. A survey of SRAM-based in-memory computing techniques and applications[J]. Journal of Systems Architecture, 2021, 119: 102276.

[10] Mittal S, Umesh S. A survey on hardware accelerators and optimization techniques for RNNs[J]. Journal of Systems Architecture, 2021, 112: 101839.

[11] Verges H T, Nikolos D. Efficient fault tolerant cache memory design[J]. Microprocessing and microprogramming, 1995, 41(2): 153-169.

[12]杨涛,郑烃,徐正欢,施钱宝,彭思伟.通用缓存替换策略下的缓存强一致性研究[J].计算机工程,2022,48(12):180-188+195.DOI:10.19678/j.issn.1000-3428.0063848.

[13] Aleksey Pesterev, Nickolai Zeldovich, and Robert T. Morris. 2010. Locating cache performance bottlenecks using data profiling. In Proceedings of the 5th European conference on Computer systems (EuroSys '10). Association for Computing Machinery, New York, NY, USA, 335–348. https://doi.org/10.1145/1755913.1755947

[14] Moinuddin K. Qureshi, Aamer Jaleel, Yale N. Patt, Simon C. Steely, and Joel Emer. 2007. Adaptive insertion policies for high performance caching. SIGARCH Comput. Archit. News 35, 2 (May 2007), 381–391. https://doi.org/10.1145/1273440.1250709

[15] K. Morales and B. K. Lee, "Fixed Segmented LRU cache replacement scheme with selective caching," 2012 IEEE 31st International Performance Computing and Communications Conference (IPCCC), 2012, pp. 199-200, doi: 10.1109/PCCC.2012.6407712.

[16] G. Hasslinger, J. Heikkinen, K. Ntougias, F. Hasslinger and O. Hohlfeld, "Optimum caching versus LRU and LFU: Comparison and combined limited look-ahead strategies," 2018 16th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt), 2018, pp. 1-6, doi: 10.23919/WIOPT.2018.8362880.

[17] Mazen Kharbutli and Yan Solihin. 2008. Counter-Based Cache Replacement and Bypassing Algorithms. IEEE Trans. Comput. 57, 4 (April 2008), 433–447. https://doi.org/10.1109/TC.2007.70816

[18] Y. M. Chung and Z. A. Halim, "Combining Reused Frequency, Most Recently Used and

Program Counter Predictor as Last Level Cache Replacement Policy," 2018 IEEE Student Conference on Research and Development (SCOReD), 2018, pp. 1-6, doi: 10.1109/SCORED.2018.8711066.
[19] G. Kaur, R. R. Maiya and R. Bharti, "Ml-Powered Cache Replacement Algorithm," 2022 IEEE 7th International conference for Convergence in Technology (I2CT), 2022, pp. 1-5, doi: 10.1109/I2CT54291.2022.9824712.
[20] Bryan Black, Bohuslav Rychlik, and John Paul Shen. 1999. The block-based trace cache. In Proceedings of the 26th annual international symposium on Computer architecture (ISCA '99). IEEE Computer Society, USA, 196–207. https://doi.org/10.1145/300979.300996
[21] Inoue, K., Ishihara, T., & Murakami, K. (1999). Way-predicting set-associative cache for high performance and low energy consumption. 273-275. Paper presented at Proceedings of the 1999 International Conference on Low Power Electronics and Design (ISLPED), San Diego, CA, USA. https://doi.org/10.1145/313817.313948
[22] P. Chen, J. Yue, X. Liao and H. Jin, "Trade-off Between Hit Rate and Hit Latency for Optimizing DRAM Cache," in IEEE Transactions on Emerging Topics in Computing, vol. 9, no. 1, pp. 55-64, 1 Jan.-March 2021, doi: 10.1109/TETC.2018.2800721.
[23] N. Gulur, M. Mehendale, R. Manikantan and R. Govindarajan, "Bi-Modal DRAM Cache: Improving Hit Rate, Hit Latency and Bandwidth," 2014 47th Annual IEEE/ACM International Symposium on Microarchitecture, 2014, pp. 38-50, doi: 10.1109/MICRO.2014.36.
[24] Przybylski S, Horowitz M, Hennessy J. Characteristics of performance-optimal multi-level cache hierarchies[J]. ACM SIGARCH Computer Architecture News, 1989, 17(3): 114-121.
[25] 肖侬,赵英杰,刘芳,陈志广.基于顺序检测的双队列缓存替换算法[J].中国科学：信息科学,2011(4).
[26] S. Kumar and P. K. Singh, "An overview of modern cache memory and performance analysis of replacement policies," 2016 IEEE International Conference on Engineering and Technology (ICETECH), 2016, pp. 210-214, doi: 10.1109/ICETECH.2016.7569243.
[27] I. Lokegaonkar, D. Nair and V. Kulkarni, "Enhancement of Cache Memory Performance," 2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N), 2021, pp. 1490-1492, doi: 10.1109/ICAC3N53548.2021.9725639.
[28] F. C. Silva and I. S. Silva, "A Redundant Approach to Increase Reliability of Data Cache Memories," 2021 XLVII Latin American Computing Conference (CLEI), 2021, pp. 1-7, doi: 10.1109/CLEI53233.2021.9640087.
[29] A. D. Joshi and N. Ramasubramanian, "Comparison of significant issues in multicore cache coherence," 2015 International Conference on Green Computing and Internet of Things (ICGCIoT), 2015, pp. 108-112, doi: 10.1109/ICGCIoT.2015.7380439.
[30] 闵庆豪,张为华.多核缓存优化技术研究综述[J].计算机系统应用,2015,24(01):1-8.