内存 HDFS 缓存技术相关工作

黄一夫

缓存替换算法：

1. 缓存性能的主要指标是延迟和命中率，各种缓存替换算法则是在两者间寻求折中[1]。Bélády's Algorithm：最优的算法，将未来最久不需要的信息替换出去，但一般无法投入实践。Least Recently Used (LRU)：将最近最少使用的信息替换出去。Most Recently Used (MRU)：将最近最常使用的信息替换出去。Random Replacement (RR)：随机替换。Least-Frequently Used (LFU)：最不常使用的信息替换出去。以上是最基本的缓存替换算法，考虑到缓存映射关系和动态可适应性，越来越多的变种缓存替换算法被提出。

2. Papaefstathiou 等人提出 explicit bulk prefether (EBP)和 epoch-based cache management (ECM)来获得更有效率的基于任务的计算[11]。EBP 是允许软件来显式地预取任务相关的内存的硬件单元。ECM 是通过任务需求来提升替换策略的硬件机制。

3. Wu 等人提出 Prefetch-Aware Cache Management (PACMan)机制，将预取与缓存管理结合，来取得更好与更可预测的性能[12]。PACMan 的目标在于：防止有害预取请求造成的缓存污染；保存不易预取的缓存行。他们将 PACMan 应用到最近的 Dynamic Re-Reference Interval Prediction (DRRIP)替换策略上，获得良好效果。

4. Levandoski 等人提出四种基于指数平滑的算法来估计内存数据库中的热数据和冷数据[18]。算法分为前向和后向扫描日志，以及对应的并行化版本。

5. Gill 等人将顺序预取与 ARC 进行了整合，提出了 SARC[19]。他们在两种流行的 LRU 变种上进行了实现，得到了良好的性能。

6. Gill 等人推出 Adaptive Multi-stream Prefetching (AMP)，该算法主要是和 SARC 配合使用[20]。AMP 管理 Sequential Read Cache，并决定什么时间来做 Prefetch，Prefetch 什么样的数据，另外，AMP 还可以动态调节 Prefetch 的量和时间点。

7. Patterson 等人提出的 Informed Prefetching and Caching 为三种需求动态地分配文件缓存[21]：预取暗示的块；缓存暗示的块来重用；缓存最近使用的数据满足未暗示的读取。

8. Megiddo 等人提出了 Adaptive Replacement Cache (ARC)，旨在根据变化的读取模式来连续地平衡最近和最常用的重要性[22]。其性能超过了 LRU。

9. PODLIPNIG 等人总结了网络缓存替换算法[24][35]。主要从时间，频率，时间/频率，功能，随机化等方面对各种网络缓存算法进行分类介绍。

10. Ambühl 等人研究了将预取和缓存整合到并行的情况[26]。他们形式化地证明了这是一个 NP 难的问题，因此该问题只能通过近似算法进行求解。

11. Cao 等人第一次将预取和缓存策略进行了结合[27]。他们提出了最优结合策略必须满足的四个条件，并且提出了两种算法 aggressive 策略和 conservative 策略。

12. Yue 等人提出能量和温度感知的缓存替换算法[29]。他们将启发式的策略整合到 LRU 中，用于降低能耗和温度。

13. Belady 最初提出了替换算法，开拓了该领域的研究[30]。

14. Jaleel 等人提出了高性能的缓存替换算法 Re-reference Interval Prediction (RRIP)[31]。具体分为静态和动态两种，算法的性能超过了常用的 LRU 和 LFU。

15. Wang 等人使用编译器来提升缓存替换策略[33]。

16. Jain 等人利用软件辅助的方式来提升嵌入式系统的缓存替换策略[34]。

17. Keramidas 等人提出基于重用距离预测的缓存替换算法[36]。

18. Kharbutli 等人提出基于计数器的缓存替换算法[37]：Access Interval Predictor (AIP) and Live-

time Predictor (LvP)。
19. Michaud 提出了数种缓存替换策略[38]。
20. Qureshi 等人提出考虑到 Memory Level Parallelism (MLP)的缓存替换算法[39]。
21. Gao 等人提出了一种高性能的缓存替换算法 Dueling Segmented LRU replacement algorithm with adaptive Bypassing (DSB)[41]。
22. Liu 等人提出新的一类坏死数据块预测，主要基于缓存块的爆发访问[42]。
23. Jimenez 提出 sampling dead block predictor 来做缓存替换[43]。
24. Chaudhuri 提出 pseudo-last-in-first-out (pseudo-LIFO)，其是用栈来管理缓存的新的算法[44]。
25. Fares 等人在大规模的的数据集上测试常用的缓存替换策略如 FIFO，LRU 和 LFU 等[45]。

内存 HDFS：
1. Zhang 等人为 HDFS 设计了一个分布式缓存系统 HDCache[2]，在于提供实时的云服务：缓存服务用 DHT 进行组织；缓存算法使用 LRU 进行替换；缓存文件有三个备份，用于提升鲁棒性和负载均衡。实验证明了 HDCache 能存储各种大小的文件，并提供良好的读取性能。
2. Singh 等人基于 Memcached 设计主动预取和缓存机制，然后整合到 Hadoop 中[3]：他们采用了两阶段贪婪缓存策略（发送和接收）；Memcached 服务器采用 LRU 进行缓存替换；全局采用多备份的 LRU 进行缓存替换；预取采用顺序策略。
3. Shrivastava 等人提出 Hadoop-Collaborative Caching 的系统架构来降低执行时间[6]。它将协同缓存，引用缓存和 Modified-ARC 算法结合了起来。Modified-ARC 算法将缓存分为两部分：已缓存对象（实际的数据）和历史对象（已换出数据的引用）。两种对象均包含最近的和最常用的。
4. Zhang 等人在 map 任务和 reduce 任务之间使用分布式缓存来提供高速的读取性能[7]。中间的分布式缓存使用 Memcached 进行实现。
5. Xiao 等人提出层次化的 MapReduce 方法 Azwraith 来最大化数据局部性和任务并行性[8]。它将数据保留在内存里，以满足结果的重用。
6. Luo 等人基于 HDFS 提出了 RAMCloud Storage System (RCSS)来获得良好的随机读性能[9]。RCSS 将 HDFS 作为备份工具，采用带有优先级的 LRU 缓存替换算法，并定义数据节点距离来进行备份。
7. Ananthanarayanan 等人考虑到并行计算所需的缓存，提出了分布式缓存服务 PACMan[10]。他们实现了两种缓存替换策略：最小化平均完成时间的 LIFE 算法；最大化集群效率的 LFU-F 算法。由于最大化缓存命中率不一定能最小化平均完成时间，LIFE 考虑更多的任务拥有完整的缓存。集群效率取决于完成任务需要的最小量的资源，LFU-F 保留常用的文件而换出少用的文件。

层次化内存架构：
1. Zhao 等人为 DFS 设计了中间件 HyCache，旨在管理异构的存储设备，将 SSD 的性能与 HDD 的容量很好地结合了起来[4]：Request Handler，处理并转发请求；File Dispatcher，提供 LRU 和 LFU 缓存替换算法；Data Manipulator，提供不同速度的访问点。
2. Zhao 等人在中间件 HyCache 的基础上扩展提出分布式存储中间件 HyCache+，支持网络存储，支持多份备份，有更强的扩展能力。提出了 2-Layer Scheduling (2LS)来优化网络传输和启发式地降低磁盘 I/O 消耗：通过形式化地定义机器，任务和文件等变量，求解任务调度的目标函数；通过制定启发式的预取和替换规则，来设计相应的算法，达到缓存性能的提高。
3. Chen 等人将 SSD 和 DRAM 结合起来构造混合内存[13]。他们采取两种方法来提高 DRAM 的命中率：使用自适应的预取机制；使用新颖的替换策略。其中，使用了 LZ 算法来挖掘文

件访问的模式。
4. Chaudhuri 等人提出基于三层缓存架构的 cache hierarchy-aware replacement (CHAR)算法[14]。其中 dead hint detector 的核心思想是通过基于距离的分类来判断将被换出的缓存是否为误判。
5. Zahran 展示了现有局部缓存替换算法的缺点，进而提出数种全局替换算法来克服这些问题[25]。他将缓存替换算法引入到更加普遍性的研究。
6. Xu 等人提出了一种新的缓存替换算法 Least Waiting Probability (LWP)[28]。主要针对于处于层次化存储里的流媒体对象。
7. Huang 等人提出一种新颖的 SSD 缓存管理算法 Lazy Adaptive Replacement Cache (LARC)[40]。

分布式内存系统：
1. RAMCloud 是一个键值对存储，它将所有的数据都保存到 DRAM 里[15]。
2. Memcached 是基于内存的键值对存储，主要针对较小的数据缓存[16][32]。
3. Redis 是含有较多抽象数据类型的键值对存储[17]。
4. GMS 利用全局的信息来达到网络带宽和替换算法的最优[23]。

参考文献：
[1] Cache algorithms – Wikipedia. http://en.wikipedia.org/wiki/Cache_algorithms.
[2] A Distributed Cache for Hadoop Distributed File System in Real-Time Cloud Services. GRID. 2012.
[3] A Dynamic Caching Mechanism for Hadoop using Memcached. Course Project. 2012.
[4] HyCache: A User-Level Caching Middleware for Distributed File Systems. IPDPS. 2013.
[5] HyCache+: Towards Scalable High-Performance Caching Middleware for Parallel File Systems. CCGrid. 2014.
[6] Hadoop-Collaborative Caching in Real Time HDFS. PDPTA. 2013.
[7] Accelerating MapReduce with distributed memory cache. ICPADS. 2009.
[8] A Hierarchical Approach to Maximizing MapReduce Efficiency. PACT. 2011.
[9] A RAMCloud Storage System based on HDFS: Architecture, implementation and evaluation. JSS. 2013.
[10] PACMan: Coordinated Memory Caching for Parallel Jobs. NSDI. 2012.
[11] Prefetching and cache management using task lifetimes. ICS. 2013.
[12] PACMan: prefetch-aware cache management for high performance caching. MICRO. 2011.
[13] A hybrid memory built by SSD and DRAM to support in-memory Big Data analytics. KIS. 2014.
[14] Introducing hierarchy-awareness in replacement and bypass algorithms for last-level caches. PACT. 2012.
[15] RAMCloud. https://ramcloud.stanford.edu/wiki/display/ramcloud/RAMCloud.
[16] Memcached. http://memcached.org.
[17] Redis. http://redis.io.
[18] Identifying hot and cold data in main-memory databases. ICDE. 2013.
[19] SARC: Sequential Prefetching in Adaptive Replacement Cache. ATEC. 2005.
[20] AMP: adaptive multi-stream prefetching in a shared cache. FAST. 2007.
[21] Informed Prefetching and Caching. SOSP. 1995.
[22] Outperforming LRU with an Adaptive Replacement Cache. Computer. 2004.
[23] The Global Memory System. http://homes.cs.washington.edu/~levy/gms.
[24] A survey of Web cache replacement strategies. CSUR. 2003.

[25] Cache Replacement Policy Revisited. WDDD. 2007.
[26] Parallel prefetching and caching is hard. STACS. 2004.
[27] A study of integrated prefetching and caching strategies. SIGMETRICS. 1995.
[28] A cache replacement algorithm in hierarchical storage of continuous media object. WAIM. 2004.
[29] Energy and thermal aware buffer cache replacement algorithm. MSST. 2010.
[30] A Study of Replacement Algorithms for a Virtual-storage Computer. IBM Systems Journal. 1966.
[31] High Performance Cache Replacement using Re-reference Interval Prediction (RRIP). ISCA. 2010.
[32] Distributed Caching with Memcached. Linux Journal. 2004.
[33] Using the compiler to improve cache replacement decisions. PACT. 2002.
[34] Software-assisted cache replacement mechanisms for embedded systems. ICCAD. 2001.
[35] Role of Aging, Frequency, and Size in Web Cache Replacement Policies. HPCN. 2001.
[36] Cache Replacement Based on Reuse Distance Prediction. ICCD. 2007.
[37] Counter-based Cache Replacement and Bypassing Algorithms. Computers. 2008.
[38] The 3P and 4P cache replacement policies. JILP. 2010.
[39] A case for MLP-aware cache replacement. ISCA. 2006.
[40] Improving flash-based disk cache with lazy adaptive replacement. MSST. 2013.
[41] A dueling segmented LRU replacement algorithm with adaptive bypassing. JWAC. 2010.
[42] Cache Bursts: A New Approach for Eliminating Dead Blocks and Increasing Cache Efficiency. MICRO. 2008.
[43] Dead Block Replacement and Bypass with a Sampling Predictor. JWAC. 2010.
[44] Pseudo-LIFO: The Foundation of a New Family of Replacement Policies for Last-level Caches. MICRO. 2009.
[45] Per-formance evaluation of traditional caching policies on a large system with petabytes of data. NAS. 2012.