I. RELATED WORK

There has been various prior works on in-memory storage and caches. Cache for distributed storage systems is different from that in traditional single-node page-based file systems. Simple cache replacement policies, like LRU, gain poor efficiency performance encountering big data scenarios.

Some framework-specific solutions include [6], [12], etc. [12] accelerates map-reduce process by caching temporary data between map and reduce processes. HDCache [6] is a distributed layered cache system built on the top of HDFS. It manages cache layer similarly to HDFS manages disk data, and organizes cache services in a P2P style using a distributed hash table. On the other hand, our approach EARNCache along with Tachyon [7] manages cache as a distributed file system, which can benefit a greater variety of computing frameworks and under file systems.

RAMCloud [13] proposes keeping information entirely in memory. Also various in-memory databases [14] [4] [5] store and process all data in memory. Spark [2] also benefits from in-memory computing. All-in-memory approach fits well in web services, while memory capacity becomes a bottleneck in big data semantics.

[9], [10] and [8] improve cache efficiency and fairness through different replacement strategies. PACMan [9] proposed LIFE and LFU-F policies. LIFE minimizes average completion time of jobs by favouring input files with smaller waves, and LFU-F maximizes cluster efficiency by favouring files that are accessed more frequently. FairRide [10] focuses on management of shared resources and extends the max-min [15][16] fairness with probabilistic blocking to avoid cheating. [8] proposed a dynamic partition strategy based on predicted utility on both fairness and performance. EARNCache also improves cache efficiency and fairness through new cache replacement policy. In comparison, EARNCache calculates base partition of memory by the access frequency information and adopts additional strategy to achieve fairness using the partition as a guide.

REFERENCES

[1] http://hadoop.apache.org/

[2] http://spark.apache.org/

[3] http://www.alluxio.org/

[4] https://memcached.org/

[5] http://redis.io/

[6] Zhang J, Wu G, Hu X, et al. A distributed cache for hadoop distributed file system in real-time cloud services. *International Conference on Grid Computing. IEEE.*, 2012: 12-21.

[7] Li H, Ghodsi A, Zaharia M, et al. Tachyon: Reliable, memory speed storage for cluster computing frameworks. *SoCC.*, 2014.

[8] Li Y, Feng D, Shi Z. Enhancing both fairness and performance using rate-aware dynamic storage cache partitioning. *International Workshop on Data-Intensive Scalable Computing Systems.* 2013.

[9] Ananthanarayanan G, Ghodsi A, Wang A, et al. PACMan: coordinated memory caching for parallel jobs. *NSDI.*, 2012.

[10] Pu Q, Li H, Zaharia M, et al. FairRide: near-optimal, fair cache sharing. *NSDI.* 2016.

[11] Zaharia M, Chowdhury M, Das T, et al. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. *NSDI.* 2012.

[12] Zhang S, Han J, Liu Z, et al. Accelerating MapReduce with distributed memory cache. *ICPADS.* 2009.

[13] Ousterhout J, Agrawal P, Erickson D, et al. The case for RAMClouds: scalable high-performance storage entirely in DRAM. *ACM SIGOPS Operating Systems Review.* 2009.

[14] Kallman R, Kimura H, Natkins J, et al. H-store: a high-performance, distributed main memory transaction processing system. *VLDB Endowment.* 2008.

[15] Ma Q, Steenkiste P, Zhang H. Routing high-bandwidth traffic in max-min fair share networks *SIGCOMM CCR.* 1996.

[16] Cao Z, Zegura E W. Utility max-min: An application-oriented bandwidth allocation scheme *INFOCOM.* 1999.