I. Experiment

EARN is evaluated on a 5-node cluster on Amazon EC2, one as master and four as worker. Each node is of m4.2xlarge type with 8 cores and 32GB of memory. 20GB of memory is used for cache in worker nodes, which summarized to 80GB of cache in total.

We measured the performance of EARN under different circumstances in comparison to other different cache eviction strategies including LRU, LFU and MAXMIN. One experiment is composed of scanning three files, which is denoted as FILE-1, FILE-2 and FILE-3. There are three frequency patterns, meaning ROUND, ONE and TWO ass described below.

- ROUND: Three files are accessed in the same frequency.
- ONE: One of the three files is accessed more frequently.
- TWO: Two of the three files are accessed more frequently.

EARN uses the window size of 1000GB in default.

Fig. 1 shows the average processing time of different strategies in different frequency patterns. Each file is 40GB in size. It can be seen that EARN is the fastest of all. Fig. 2 shows similar results where files are 70GB, 40GB and 10GB in size respectively.

More specifically, we analyzed the distribution of block accessing. As shown in Fig. 3, EARN has the largest number of blocks accessed in local or remote memory. It means that EARN has the highest memory efficiency compared to other strategies. This is also the reason EARN strategy is the fastest.

As we can see in the time figures, MAXMIN is close in time to our EARN strategy. Yet essentially, MAXMIN helps maintain the same amount of memory for all files or users, which can be a problem when some files stop been accessed or some users become inactive. To illustrate this, we simulated a scenario where first all three files are accessed in the same frequency and later one or two files stops been accessed. Fig. 4 shows changes in amount of in-memory data after FILE-3 is not accessed. The X-axis is the number of experiments, which can be viewd as the dimension of time. Apparently, the remaining active files gains more memory resource with time going in EARN strategy, while FILE-3 still holds nearly 1/3 of total memory even it's not accessed. That is quite a waste of limited memory resource.

Fig. 5 shows similar results when both FILE-2 and FILE-3 are not accessed any more. The gain in memory of FILE-1 is more remarkable.

Finally, we also analyzed the impact of window size. As shown in Fig. 6, when windown size is too small, the time performance will degrade.

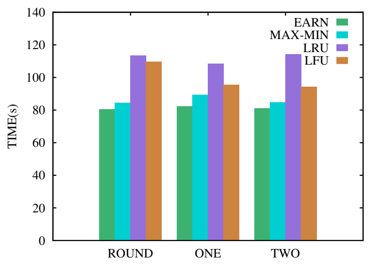


Fig. 1. Access Time. Three files are equal in size (40GB).

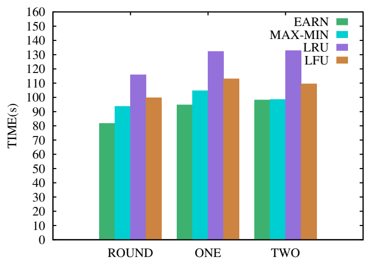
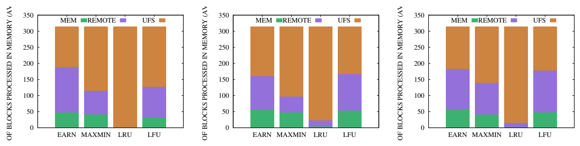
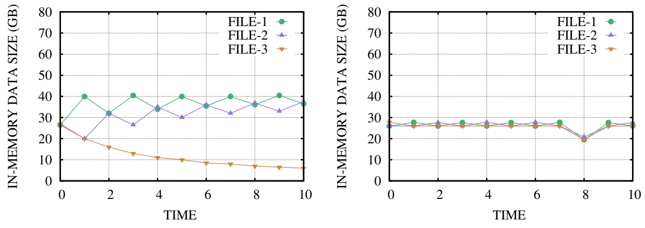


Fig. 2. Access Time. The size of files is 70GB, 40GB and 10GB respectively.



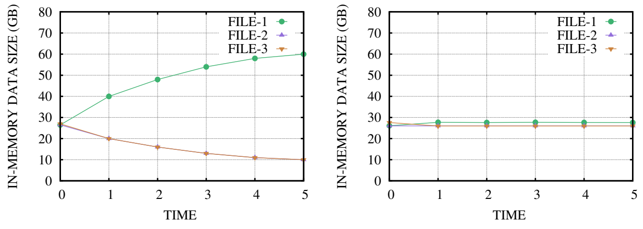
(a) ROUND (b) ONE (c) TWO

Fig. 3. Distribution of blocks accessed in memory, remote memory and under file system.



(a) EARN (b) MAXMIN

Fig. 4. In-memory data size of each file after File-3 stops been visited.



(a) EARN (b) MAXMIN

Fig. 5. In-memory data size of each file after File-3 and File-2 stop been visited.

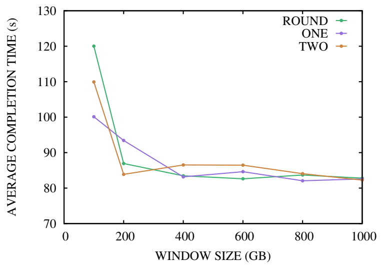


Fig. 6. Access Time. Each with different window size.