Cost-Driven Data Caching with Predictions

TIANYU ZUO, Nanyang Technological University, Singapore
XUEYAN TANG, Nanyang Technological University, Singapore
BU SUNG LEE, Nanyang Technological University, Singapore

ACM Reference Format:
Tianyu Zuo, Xueyan Tang, and Bu Sung Lee. 2025. Cost-Driven Data Caching with Predictions. *LATA 2025* 1, 1 (May 2025), 2 pages. https://doi.org/XX.XXXX/XXXXXXX.XXXXXXX

1 Problem Definition

We study an online caching problem for distributed data access. Consider a system including n geo-distributed servers (or sites) $s_1, s_2, \ldots, s_n$. A data object is hosted in the system and copies of this object can be created and stored in any servers. Storing a data copy in a server incurs a cost of 1 per time unit. The data object can also be transferred between servers when needed. The transfer cost of the object between any two servers is λ ($\lambda > 0$).

Requests to access the data arise at the servers over time (e.g., due to the computational jobs running at the servers or user requests). When a request arises at a server s_j, if s_j holds a data copy, the request is served locally. Otherwise, the object has to be transferred from a server holding a copy to s_j in order to serve the request. s_j may, as a result of the transfer, create a data copy and hold it for some period of time. We assume only one data copy placed in server s_1 initially. We seek to develop a *caching strategy* that determines the data copies to create and hold as well as the transfers to carry out in an *online* manner to serve all requests arising at various servers. There must be at least one data copy stored at all times to preserve the data. Our objective is to minimize the total storage and transfer cost of serving a request sequence. We focus on the problem in the learning-augmented setting and assume there are predictions of inter-request times at individual servers (e.g., based on the request history or other features). Specifically, we assume that after a request arises at a server, a simple binary prediction is available about whether the next request at the same server will arise within or beyond a period of λ time units from the current request. We would like to make sensible use of the predictions to reduce the cost for serving requests.

2 Main Results

Observe that there is a trade-off between holding and not holding a data copy at each individual server. If no copy is held by a server, we have to pay the transfer cost for serving each local request. If a copy is held by a server, we have to pay the storage cost which is proportional to the duration of storage. Such a trade-off appears to resemble that between rent and buy in the classical ski-rental problem. Without predictions of inter-request times, to achieve decent competitiveness, an intuitive

This extended abstract summarizes the paper [1]. This research is supported by the Ministry of Education, Singapore, under its Academic Research Fund Tier 2 (Award MOE-T2EP20121-0005) and Tier 1 (Award RG23/23).
Authors' Contact Information: Tianyu Zuo, Nanyang Technological University, Singapore, zuot0001@e.ntu.edu.sg; Xueyan Tang, Nanyang Technological University, Singapore, asxytang@ntu.edu.sg; Bu Sung Lee, Nanyang Technological University, Singapore, ebslee@ntu.edu.sg.

ACM XXXX-XXXX/2025/5-ART
https://doi.org/XX.XXXX/XXXXXXX.XXXXXXX

idea is to let a server s_j hold a copy for a period of λ time units after serving every local request. If the next request arises at s_j before this period ends, the request will be served by the local copy and we pay the storage cost which is optimal. If no request arises at s_j during the period of λ, s_j will delete the copy and an incoming transfer will then be required to serve the next request at s_j. In this case, the total cost paid is $2 \cdot \lambda$, while the optimal cost is a transfer cost λ (by not holding a copy in s_j). On the other hand, with perfect predictions of inter-request times, we can always achieve the optimal cost. That is, if the inter-request time between two consecutive requests at a server s_j is no more than λ, we keep a data copy in s_j between the requests and pay the storage cost which is optimal. Otherwise, we do not keep a copy in s_j and pay the transfer cost for serving the latter request which is optimal. Nevertheless, this solution does not have bounded robustness, because (1) if the inter-request time is predicted to be longer than λ but is actually shorter than λ, the transfer cost to pay is fixed while the optimal cost can be close to 0; and (2) if the inter-request time is predicted to be shorter than λ but is actually longer than λ, the storage cost can go towards infinity in the worst case. Furthermore, if we simply apply the above ideas (with or without predictions) to every server, it may not meet the requirement of maintaining at least one data copy at any time, because all copies will be deleted after a sufficiently long silent period without any request.

Our approach to algorithm design is to integrate and balance between following predictions and not using predictions. We introduce a hyper-parameter $\alpha \in (0, 1]$ to indicate the level of distrust in the predictions. $\alpha \to 0$ indicates nearly full trust and reliance on the predictions, while $\alpha \to 1$ indicates almost no trust and not using the predictions. If the next request at a server is predicted to arise beyond a period of λ from the current request, we let the server hold a copy for a period of $\alpha \cdot \lambda$ instead of deleting the copy immediately. Then, even if the prediction is not correct, the next request still has some chance to be served by the local copy, avoiding the transfer. This will save the cost when the next request actually arises quite soon and thus improve robustness while keeping the loss in consistency under control. On the other hand, if the next request at a server is predicted to arise within a period of λ from the current request, we let the server hold a copy for a period of λ rather than up to the next request. Then, even if the prediction is not correct, the storage cost will not increase infinitely, thereby enhancing robustness while still ensuring consistency. In addition, to meet the at-least-one-copy requirement, the copy whose period ends the latest according to the above strategies will continue to be kept beyond its period until the next request in the system.

The at-least-one-copy requirement presents considerable challenges to the algorithm analysis. We cannot simply adapt the ski-rental analysis by treating the costs at individual servers independently. Our general approach to competitive analysis is to first divide a request sequence into partitions based on the characteristics of an optimal offline strategy, and then study the costs of the proposed algorithm and the optimal strategy for each partition separately. Our main results include:

- We show that the proposed algorithm is $\left(\frac{5+\alpha}{3}\right)$-consistent and $\left(1 + \frac{1}{\alpha}\right)$-robust. We construct examples to show that our consistency analysis and robustness analysis are both tight.
- We establish a lower bound of $\frac{3}{2}$ on the consistency of any deterministic learning-augmented algorithm. This implies that it is not possible to achieve a consistency approaching 1 in our problem, and demonstrates a critical difference of our problem from classical ski-rental.
- We analyze the impact of mispredictions on the total cost and leverage the analysis to monitor (an upper bound of) the online-to-optimal cost ratio as time goes on and adapt our algorithm to achieve an improved robustness while retaining its consistency.

We also empirically evaluate our algorithms using real data access traces.

References

[1] Tianyu Zuo, Xueyan Tang, and Bu Sung Lee. 2024. Cost-Driven Data Replication with Predictions. In *Proceedings of the 36th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*. 309–320.