Niklas Semmler, Matthias Rost, Georgios Smaragdakis, Anja Feldmann

Edge Replication Strategies for Wide-Area Distributed Processing

Semmler, N., Rost, M., Smaragdakis, G., & Feldmann, A. (2020). Edge replication strategies for wide-area distributed processing. Proceedings of the Third ACM International Workshop on Edge Systems, Analytics and Networking. Presented at the EuroSys '20: Fifteenth EuroSys Conference 2020. https://doi.org/10.1145/3378679.3394532

Edge Replication Strategies for Wide-Area Distributed Processing

Niklas Semmler
niklas.semmler@sap.com
SAP SE
Walldorf, Germany

Matthias Rost
mrost@inet.tu-berlin.de
TU Berlin
SAP SE

Georgios Smaragdakis
TU Berlin
Max Planck Institute for Informatics

Anja Feldmann
anja@mpi-inf.mpg.de
Max Planck Institute for Informatics
Saarbrücken, Germany

ABSTRACT

The rapid digitalization across industries comes with many challenges. One key problem is how the ever-growing and volatile data generated at distributed locations can be efficiently processed to inform decision making and improve products. Unfortunately, wide-area network capacity cannot cope with the growth of the data at the network edges. Thus, it is imperative to decide which data should be processed in-situ at the edge and which should be transferred and analyzed in data centers.

In this paper, we study two families of proactive online data replication strategies, namely ski-rental and machine learning algorithms, to decide which data is processed at the edge, close to where it is generated, and which is transferred to a data center. Our analysis using real query traces from a Global 2000 company shows that such online replication strategies can significantly reduce data transfer volume—in many cases up to 50% compared to naive approaches—and achieve close to optimal performance. After analyzing their shortcomings for ease of use and performance, we propose a hybrid strategy that combines the advantages of both competitive and machine learning algorithms.

KEYWORDS

data replication, distributed systems, edge computing

1 INTRODUCTION

Data is the new oil. Many successful applications, such as online social networks [6], online tracking and advertisements [19], location-based marketing and navigation [18], rely on data generated at the edge of the network. We expect that this trend will continue, e.g., with the Internet of Things (IoT). Today, there are about four billion online users, i.e., half the world population [17]. Yet, the estimated number of connected devices already exceeds twenty billion [7] and they generate traffic around the clock [13]. With the increasing penetration of IoT, e.g., in all areas of the fourth industrial revolution including production lines and self-driving cars, a dramatic increase of both the number of connected devices and the produced data at the network edge is forecasted [7]. Processing of such voluminous data is necessary, e.g., to inform production-related decision making, investments, and improve products [14].

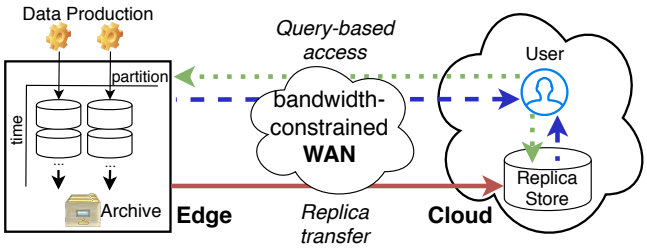

Figure 1: Edge-to-cloud transfers face a trade-off between transferring individual query results and replicating entire partitions.

Figure 1 illustrates a typical setting of wide-area distributed data processing to enable data-driven applications. The data arrives asynchronously as a stream of data partitions at the network edge. Data partitions aggregate thousands of rows or columns, e.g., to speed up lookups [16], enable parallel data processing [3], and reduce storage [11]. Middleboxes at the network edge collect, store, and analyze these partitions. These middleboxes must decide which raw data resp. summaries to forwarded to remote data centers, i.e., cloud infrastructures, based on application requirements and data access patterns. Ideally, all data partitions should be forwarded to data centers for central processing. However, this is undesirable or even infeasible due to limited wide-area bandwidth, high latency to the remote data center, data privacy regulations, or simply because only a small fraction of the data will ever be accessed by applications. Nevertheless,

data that is frequently accessed should be *replicated* to the data centers to reduce application response time.

Recent studies have shown that online replication strategies can reduce the cost of replicating data when data is immutable [15]. However, for many applications data ages quickly, requiring to regularly invalidate already replicated partitions [4]. To model this requirement, we, in this paper, consider data partitions to be immutable only within (short) time windows. Hence, the task is to minimize replication costs over *several* time windows. Still, historical information from preceding time windows can and should be used to inform replication decisions. Yet, even access patterns and popularities of partitions are subject to *volatility*.

In this paper, we apply the ski-rental algorithm [2, 8, 9] to decide which data within a relatively short time window should be proactively forwarded from the network edge to the cloud. Previous works have shown the potential of Machine Learning for efficiently replicating data, e.g., in the case of Content Delivery Network caching [1]. To that end, we propose a Machine Learning (ML) based strategy for replicating data. We compare the performance of both strategies, namely the ski-rental and the ML-based ones, against naïve strategies, i.e., replicate all and replicate nothing, as well as the optimal offline one.

Our contributions can be summarized as follows.

- Analysis of the potential benefit of reactive online replication strategies in time windows at the network edge.
- Evaluation of multiple online edge replication strategies, including competitive ratio-based and ML-based strategies, by applying them to two real datasets that span four days.
- Introduction of hybrid online edge replication strategies that combine the benefits of both the competitive ratio-based and ML-based strategies.

2 DATASET

To evaluate our proposed online replication strategies we use a set of Enterprise resource planning (ERP) database traces of a Global 2000 company [4]. After giving an overview of the traces we detail how we cater it to our use case.

2.1 Raw Traces

The traces, see [4], record queries to various tables of an ERP database of a Global 2000 company. For each table and each query, the time of execution, as well as the accessed rows, are recorded. The traces span less than three days. We focus on production queries and, hence, exclude backup transfers taking place around 1 am each day. We focus on the two tables that where accessed most frequently and, hence, obtain two (sub-)traces. The first trace contains roughly 100 Million rows and records about 2.5 Million queries. The second trace has 24 Million rows and roughly 1.3 Million queries. Table 1 gives an overview of both traces.

Table 1: Trace statistics for the two largest tables [4]

Name	Trace 1	Trace 2
Table size in rows [million]	100	24
Number queries [million]	2.49	1.28
Duration [days]	≈ 3	≈ 3
Accesses in rows [million]	137	34
Avg. rows per query	55	26

2.2 Data Cleaning

In the following, we detail our modifications to the raw data to cater them to our use case.

- For performance reasons, the original trace only contains queries for the first 2 minutes for each 10 minute interval. To remove these gaps, we scale the captured queries to the full 10 minute interval.
- We introduce data partitions by aggregating 10k adjacent rows into a single partition[1], yielding roughly 10k and 2.4k partitions, respectively.
- We focus on two (full) consecutive days and aggregate accesses to partitions in 100-second intervals, yielding 864 data points per partition, trace, and day.

2.3 Time Windows

For our analysis, we focus on time windows of one day which results in two time windows per trace. Figure 2 depicts the richness of our datasets. Clearly, the accesses recorded in both traces vary over time and are (temporally) correlated both within and across time windows (days): Periods with a large number of accesses are interleaved with periods of few to no accesses. Data locality correlations between adjacent partitions can also be observed: For example in trace 2 for partitions 1720 through 2440 around the 5h mark. Additionally, we note that row accesses per partition are highly skewed. For roughly 75% of the partitions less than 1k row accesses are recorded while for more than 1% of the partitions far more than 10k accesses are recorded. Notably, these heavy-hitters are observed at different times and they can change from day to day: The common peaks at 15h on day 2 for trace 2 do not exist in the previous day. Thus, the training process of learning from previous windows may be challenging.

3 EDGE REPLICATION: CHALLENGES AND OPPORTUNITIES

Next, we introduce the general system model, introduce and discuss naïve replication schemes, and analyze potential cost reductions.

[1]Partition sizes between 1k [16] and 1,000k [11] are common.

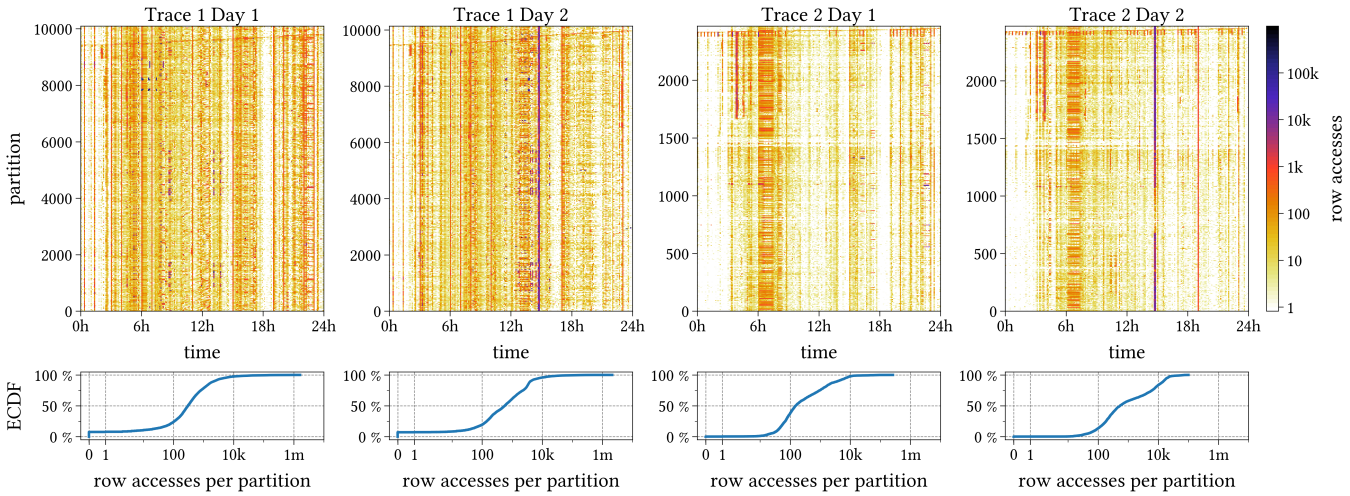

Figure 2: Visualization of obtained traces. Top: row accesses aggregated over 500s intervals and 20 (trace 1) or 5 partitions (trace 2). Bottom: ECDF of cumulative row accesses per partition. Note the logarithmic x-axes.

3.1 Setting

As discussed in Section 1, we assume that data is stored locally at the network edge while queries are processed in the cloud—remote data centers. The main challenge is to decide which partitions to replicate to the cloud and which data partitions to keep at the edge. In this paper, our primary concern is the reduction of the transferred data volume. We leave the inclusion of secondary cost factors, e.g., storage and processing cost, to future work.

While replicating a partition incurs a transfer cost proportional to the size of the partition (in our case 10k rows, cf. Section 2), queries to non-replicated partitions are processed at the edge and incur transfer costs proportional to the number of accessed rows. We refer to the former cost type as the *replication cost* and the latter type as the *transfer cost*. We furthermore denote by *partition cost* the cumulative costs for serving all queries of a partition from the edge, i.e., the cumulative shipping cost. Clearly, replicating a partition to the cloud only yields a benefit if the *remaining partition cost* is at least as high as the replication cost itself.

3.2 Naïve Replication Strategies

A replication strategy is an algorithm that decides whether and when to replicate each partition. Its cost is the total transfer costs incurred within a time window across all partitions. The naïve replication strategies are either: *replicate nothing* or *replicate all*. These strategies do not differentiate between individual partitions. Hence, their performance depends on the ratio of partitions whose partition cost exceeds their replication cost. This may not only depend on the raw access volume but also system assumptions. For example, shipping individual rows is likely to incur a higher overhead than

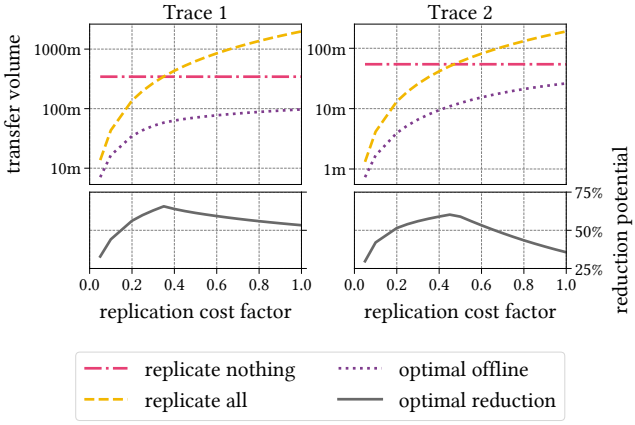

Figure 3: Costs of the naïve and optimal offline strategies and comparison of naïve baseline to optimum.

replicating a partition as a whole. Furthermore, data partitions can often be compressed significantly [11] such that the replication cost may be a fraction of the cost of transferring all rows individually. We refer to this ratio of replication cost to transfer cost as the *replication cost factor*, whereby a factor of, e.g., 0.5 implies that replicating a partition only incurs half the costs of transferring all rows individually. To broaden the scope of our analysis we study 10 replication cost factors, namely $\{0.1, 0.2, \ldots, 1.0\}$, albeit assuming the same replication cost factor for all partitions.

The optimal replication strategy is to replicate an individual partition only if its remaining partition cost is higher than its replication cost. Note, this optimal strategy can only be computed if all accesses are known in advance. Hence,

we refer to it as the *optimal offline* strategy. While its performance is unattainable for all practical purposes, it serves as a lower bound for other replication strategies and quantifies the replication cost reduction potential.

Figure 3 (top) depicts the cost of both naïve strategies and the optimal offline strategy for both traces. For smaller replication cost factors the replicate all strategy yields the best results while for larger ones the replicate nothing one does. Note, both are far from optimal. Referring to the minimum cost over both strategies as the naïve baseline, the optimality gap of this naïve baseline ranges from 30% to 66% (cf. bottom of Figure 3). Moreover, a single strategy may not always yield the best results: For a replication cost factor of 0.4 neither the replicate all nor the replicate nothing strategy consistently yield the best performance.

4 ONLINE REPLICATION STRATEGIES

Replication strategies can decide at any point in time to replicate a particular partition. Moreover, as future accesses and pattern shifts are often not known, these strategies are inherently online. In the following, we introduce several strategies including some that offer *competitiveness* guarantees. A replication strategy is called c-competitive if for any sequence of time windows and any access pattern the strategy's cost is at most c-times that of the cost of the optimal (offline) strategy. Next, we first discuss our competitive strategies and then propose several heuristics, including machine learning based ones. See Table 2 for a summary of all strategies studied in this paper.

4.1 Competitive Strategies

When competitive analysis first arose, Karlin et al. [8, 9] presented a very simple but effective competitive online algorithm for the so-called *ski-rental problem*: a skier may either rent skis or buy them but does not know the length of her vacation. Karlin et al. proved that the best 2-competitive strategy is to buy the skis once the cumulative daily rental fees would exceed the price of purchase.

This *ski-rental* strategy is applicable in the context of replication schemes (cf. [15]): once the cumulative transfer costs exceed the replication cost, the partition is replicated. Clearly, this strategy is also 2-competitive and the result by Karlin et al. [9] also implies that no c-competitive strategy can exist for $c < 2$. However, empirically tuning the *threshold* which is used to decide when a partition is replicated might be beneficial in practice. In particular, one may replicate a partition once the transfer costs exceed t-times the replication cost. Notably, for any constant $t > 0$, the respective strategy is $\max\{1 + f/t, 1 + t/f\}$-competitive, where f denotes the replication cost factor.

Table 2: Summary of studied replication strategies

		Strategy	Description
naïve		replicate all	replicates partitions immediately when a new time window starts
		replicate nothing	always answers queries directly
online	heuristic	last-partition	replicates partitions that previously exceeded its replication cost
		classifier	uses random forest classifier trained on previous time window
	competitive	hybrid	replicates if ski-rental or the classifier strategy would do so
		ski-rental	replicates once replication cost has been exceeded (threshold=1)
		last-threshold	sets ski-rental threshold to optimal one of previous window
		optimal offline	uses knowledge about future to inform replication decisions

Harnessing information on the previous time window, we propose the *last-threshold* strategy: (i) for the previous time window the *optimal* threshold t is computed and (ii) for the current time window the t-threshold strategy is executed.

4.2 Heuristic Strategies

Next, we propose several heuristics, i.e., strategies not providing performance guarantees.

Last-Partition Strategy. Given the accesses from the previous time window one can compute a posteriori optimal replication decisions. Accordingly, a simple strategy is to immediately replicate partitions which should have been replicated using the threshold from the previous time window. Interestingly, if access patterns are invariant over time, this strategy is *optimal*.

Machine Learning Strategies. All of the above-discussed approaches handle partitions equally: upon meeting a certain common criterion, replication is performed. This is not necessary and my give rise to improvements. To motivate *fine-granular* machine learning algorithms, consider the following (cf. [10]). First, access patterns are (highly) correlated across both: the temporal and the data-locality dimension (cf. Section 2) and the above approaches are agnostic to this *seasonality*. Second, the above approaches are agnostic to common (sub-)patterns shared by heavy hitters and which may hence be harnessed to perform replications early on. Third, learning-based approaches may be robust towards linear transformations of access patterns recorded in previous windows, e.g., due to increased demand.

To evaluate the potential of *learning* which partitions to replicate, we cast the problem as a *classification* problem. Specifically, we view partition accesses as time series and

create for each point a feature vector together with the (a posteriori known) classification decision whether the replication *at this point* would have been beneficial. We generate features by using a set of well-studied aggregation methods (e.g., sum, mean, max, etc.) over varying numbers of preceding points.

To perform the actual classification, we propose to use random forest classifiers [5] because these scale well even for large training sets, are implemented in several frameworks, and allow for human interpretation. Specifically, we propose to train the classifier over the preceding time window and, then, apply the obtained classification model to the current time window. The classifier then returns a *classification probability* in the range $[0, 1]$. We (by default) interpreted it as Boolean replication decision by checking whether the probability lies above 0.5. Besides deciding on a classification probability threshold, various classification model parameters need to be calibrated. We defer this discussion to the evaluation.

Besides the above classifier strategy, we also propose to bridge machine learning and competitive strategies to preserve performance guarantees (to some extent) while harnessing *fine-granular* trace histories. In particular, we propose the *hybrid* strategy that executes both the ski-rental and the classifier strategy: if either strategy decides to replicate a partition, the hybrid strategy also performs the replication. Intuitively, by using the disjunction of both, we aim at replicating heavy-hitters early on while preserving the 2-competitiveness for partitions whose transfer cost eventually exceeds the replication cost. We note that *calibrating* even such simple classification models requires substantial effort and defer implementation details to the evaluation section (cf. Section 5.1).

5 EVALUATION

We now evaluate the performance of the diverse replication strategies (cf. Table 2) on the dataset introduced in Section 2. Before reporting on the results, we give some details on the calibration of the machine learning-based strategies.

5.1 Classifier Calibration

Accurately calibrating classifiers while not over-training is challenging. First, due to the skewed partition cost distribution, partitions that should not be replicated dominate. We adjust the model to this imbalance by weighting feature vectors according to their class frequency. Second, feature vectors differ in importance: correctly recognizing partitions whose partition cost is either very low or very high is of vital importance. Hence, we again adjust the weights accordingly. Third, to further increase the accuracy of the classifier we combine our random forest classifiers with an isotonic regression model [12].

Being interested in the off-the-shelf performance, we employ the default classification probability threshold of 0.5 for the basic classifier strategy. For the hybrid strategy, we increase this threshold to 0.8 to minimize incorrect replication decisions but also discuss various other thresholds below.

5.2 Results

Figure 4 shows the performance of the strategies relative to the naïve baseline for the 2nd day for both traces. A number lower/higher than one reflects a reduction/increase in the relative total cost. Notably, most strategies reduce the costs for a wide range of replication cost factors. The last-partition strategy is an exception: its potential reductions are outweighed by its additional costs for replication factors below 0.4. The competitive ski-rental strategy overall outperforms the basic classifier approach (especially on trace 1) while the last-partition's performance is less consistent for trace 2. From the "pure" strategies ski rental achieves the largest reduction with an average performance improvement of 22% and a maximum performance of 50%. Over all strategies, the hybrid strategy performs best with an average performance of 25% and a maximum performance of 51%. It significantly improves upon the standalone performance for both the ski-rental and the classifier strategies in trace 2. Specifically, the hybrid strategy improves upon ski-rental's cost by 28% for a replication factor of 0.1 while yielding the same performance as ski-rental for a replication factor of 1.0 even though the classifier alone performs 13% worse. Overall, the hybrid strategy improves the ski-rental approach by 3% on average.

5.3 Hybrid Strategy: In-Depth Analysis

The hybrid strategy yields the best performance when manually selecting a probability classification threshold of 0.8. While this *a posteriori* choice highlights the potential benefits of such hybrid strategies, it also highlights the challenges of robustly calibrating machine learning models. Figure 5 depicts the performance as a function of the classification threshold over both traces. While choosing a threshold greater than or equal to 0.75 improves performance, choosing a replication threshold of 0.6 yields improvements for trace 2 while consistently worsening performance for trace 1.

To gain insights into the effectiveness of the hybrid strategy, consider Figure 6. It depicts the precision[2], recall[3], and the mean replication time for the hybrid strategy and its

[2]The number of partitions being correctly identified to be replicated over the total number of replicated partitions.

[3]The number of partitions being correctly identified to be replicated over the total number of partitions that were to be replicated.

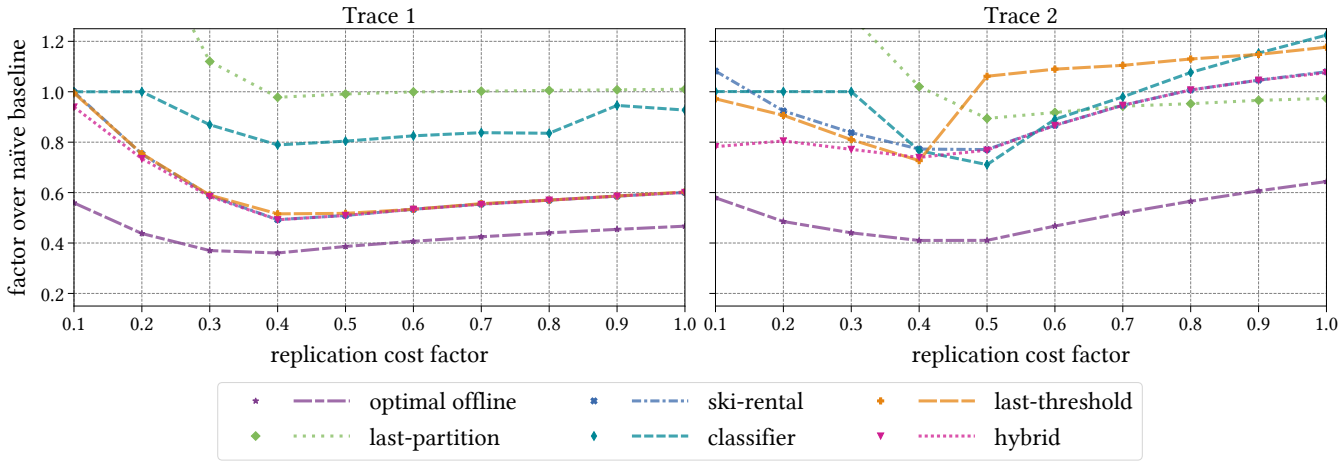

Figure 4: Performance relative to the naïve baseline, i.e., the minimal cost of the naïve approaches.

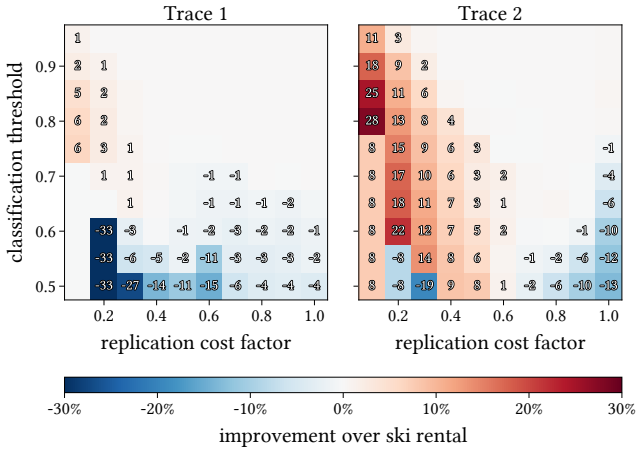

Figure 5: Sensitivity analysis of the hybrid strategy's performance relative to ski-rental's performance.

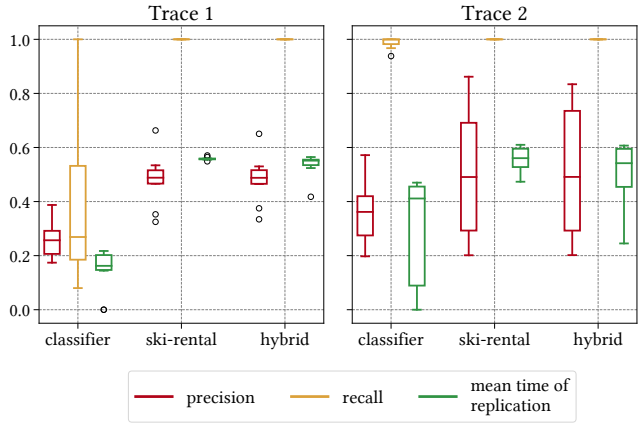

Figure 6: Analysis of hybrid strategy improvement for each sub-strategy.

sub-strategies. As ski-rental replicates any partition that exceeds the replication cost, its recall is 1.0. This carries over to the hybrid strategies which, thus, also have a recall of 1.0. However, by using the classifier the time at which partitions are replicated decreases significantly, thereby saving transfer costs. Furthermore, the classifier's slightly worse precision only marginally reduces hybrid's precision. Therefore, it does not add excessive replication cost. This explains the improvement of hybrid over ski-rental.

6 CONCLUSION

As the traffic at the network edge continues to grow at an unprecedented pace, it is imperative to decide which data should be processed in-situ at the edge and which data should be forwarded to the cloud. In this work, we observe that such decisions have to be reactive to volatile accesses. We study two families of online algorithms, namely, competitive (ski-rental) and machine learning algorithms, to inform such decisions at the edge of the network. These algorithms proactively decide which data will be replicated to the remote cloud, based on the recent access activity. Our results show that ski-rental not only yields significant cost reductions (22% on average up to 50%) compared to naïve strategies, but is also easy to use at the edge even when resources may be limited. Moreover, the best online strategy may depend on the scenario. To address this, we introduce a hybrid strategy that combines the advantages of both families of strategies. Thus, it yields the best results which are close to the offline optimal. As part of our future research agenda, we will investigate the learning curve of machine learning-based algorithms in this setting. We argue that more extended training periods may lead to better results, and thus, further improve replication at the edge. Finally, we want to extend this work to address

other cost factors, e.g., storage, processing and training time costs.

ACKNOWLEDGMENTS

This work was partially supported by the European Research Council (ERC) Starting Grant ResolutioNet (ERC-StG-679158) and by the German Ministry for Education and Research as BIFOLD - Berlin Institute for the Foundations of Learning and Data (ref. 01IS18025A and ref. 01IS18037A).

REFERENCES

[1] Daniel S. Berger. 2018. Towards Lightweight and Robust Machine Learningfor CDN Caching. In *Proceedings of the 17th ACM Workshop on Hot Topics in Networks*. https://doi.org/10.1145/3286062.3286082

[2] Marcin Bienkowski. 2009. Price Fluctuations: To Buy or To Rent. In *International Workshop on Approximation and Online Algorithms*. Springer, 25–36. https://doi.org/10.1007/978-3-642-12450-1_3

[3] Martin Boissier and Kurzynski Daniel. 2018. Workload-Driven Horizontal Partitioning and Pruning for Large HTAP Systems. In *2018 IEEE 34th International Conference on Data Engineering Workshops (ICDEW)*. 116–121. https://doi.org/10.1109/ICDEW.2018.00026

[4] Martin Boissier, Carsten Alexander Meyer, Timo Djürken, Jan Lindemann, Kathrin Mao, Pascal Reinhardt, Tim Specht, Tim Zimmermann, and Matthias Uflacker. 2016. Analyzing Data Relevance and Access Patterns of Live Production Database Systems. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. 2473–2475. https://doi.org/10.1145/2983323.2983336

[5] Leo Breiman. 2001. Random forests. *Machine learning* 45, 1 (2001), 5–32. https://doi.org/10.1023/A:1010933404324

[6] Guoqiang Jerry Chen, Janet L. Wiener, Shridhar Iyer, Anshul Jaiswal, Ran Lei, Nikhil Simha, Wei Wang, Kevin Wilfong, Tim Williamson, and Serhat Yilmaz. 2016. Realtime Data Processing at Facebook. In *Proceedings of the 2016 International Conference on Management of Data (SIGMOD '16)*. Association for Computing Machinery, New York, NY, USA, 1087–1098. https://doi.org/10.1145/2882903.2904441

[7] Cisco. 2020. Cisco Annual Internet Report (2018–2023) White Paper.

[8] Anna R Karlin, Kai Li, Mark S Manasse, and Susan Owicki. 1991. Empirical Studies of Competitve Spinning for a Shared-Memory Multiprocessor. In *ACM SIGOPS Operating Systems Review*, Vol. 25. 41–55. https://doi.org/10.1145/121133.286599

[9] Anna R Karlin, Mark S Manasse, Larry Rudolph, and Daniel D Sleator. 1988. Competitive Snoopy Caching. *Algorithmica* 3, 1-4 (1988), 79–119. https://doi.org/doi.org/10.1007/BF01762111

[10] Tim Kraska, Mohammad Alizadeh, Alex Beutel, Ed H Chi, Jialin Ding, Ani Kristo, Guillaume Leclerc, Samuel Madden, Hongzi Mao, and Vikram Nathan. 2019. SageDB: A Learned Database System. In *CIDR*.

[11] Per-Åke Larson, Cipri Clinciu, Eric N. Hanson, Artem Oks, Susan L. Price, Srikumar Rangarajan, Aleksandras Surna, and Qingqing Zhou. 2011. SQL Server Column Store Indexes. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data (SIGMOD '11)*. Association for Computing Machinery, New York, NY, USA, 1177–1184. https://doi.org/10.1145/1989323.1989448

[12] Alexandru Niculescu-Mizil and Rich Caruana. 2005. Predicting good probabilities with supervised learning. In *Proceedings of the 22nd international conference on Machine learning*. 625–632. https://doi.org/10.1145/1102351.1102430

[13] Jingjing Ren, Daniel J Dubois, David Choffnes, Anna Maria Mandalari, Roman Kolcun, and Hamed Haddadi. 2019. Information Exposure From Consumer IoT Devices: A Multidimensional, Network-Informed Measurement Approach. In *Proceedings of the Internet Measurement Confernece*. https://doi.org/10.1145/3355369.3355577

[14] Niklas Semmler, Georgios Smaragdakis, and Anja Feldmann. 2019. Distributed Mega-Datasets: The Need for Novel Computing Primitives. In *39th IEEE International Conference on Distributed Computing Systems, ICDCS 2019, Dallax, TX, USA, July 7-9, 2010*. https://doi.org/10.1109/ICDCS.2019.00167

[15] Niklas Semmler, Georgios Smaragdakis, and Anja Feldmann. 2019. Online Replication Strategies for Distributed Data Stores. *Open Journal of Internet Of Things (OJIOT)* 5, 1 (2019), 47–57.

[16] Liwen Sun, Michael J. Franklin, Sanjay Krishnan, and Reynold S. Xin. 2014. Fine-Grained Partitioning for Aggressive Data Skipping. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data (SIGMOD '14)*. Association for Computing Machinery, New York, NY, USA, 1115–1126. https://doi.org/10.1145/2588555.2610515

[17] International Telecommunication Union. 2019. Individuals Using the Internet Statistics. https://www.itu.int/en/ITU-D/Statistics/Pages/stat/default.aspx. Accessed: 2020-04-18.

[18] Jia-Dong Zhang and Chi-Yin Chow. 2015. GeoSoCa: Exploiting Geographical, Social and Categorical Correlations for Point-of-Interest Recommendations. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '15)*. Association for Computing Machinery, New York, NY, USA, 443–452. https://doi.org/10.1145/2766462.2767711

[19] Weinan Zhang, Shuai Yuan, and Jun Wang. 2014. Optimal Real-Time Bidding for Display Advertising. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '14)*. Association for Computing Machinery, New York, NY, USA, 1077–1086. https://doi.org/10.1145/2623330.2623633