Attribute-Based Partial Geo-Replication System

Hobin Yoon, Ada Gavrilovska, and Karsten Schwan
College of Computing, Georgia Institute of Technology, Atlanta, GA, USA

***Abstract*—Existing partial geo-replication systems do not always provide optimal cost or latency, because their replication decisions are based on statically established data access popularity metrics, regardless of the application types. We demonstrate that additional reduction in cost and latency can be achieved by 1) using the right object attributes for making replication decisions for each type of application, 2) using multi-attribute-based replications, and 3) combining the popularity-based but reactive approach with the more random but proactive approach to data replication. Toward this end, we propose Acorn, an Attribute-based COntinuous partial geo-ReplicatioN system, and its prototype implementation based on Apache Cassandra. Experiments with two types of global-scale, data-sharing applications demonstrate up to 54% and 90% cost overhead reduction over existing systems or 38% and 91% latency overhead reduction.**

I. Introduction

Many partial geo-replication systems have been recently introduced, including [1]–[5], to address the cost of geo-replication, which increases as the volume of data and the number of datacenters grow. They lower cost by replicating data only to where it is likely to be consumed, and their performance depends on the quality of the future data access predictions, typically based on the history of data accesses. They use access statistics of a specific data attribute like "user" for making replication decisions; some systems use the attribute "user" directly [1]–[3], while others use it indirectly [4], [5]. The latter systems use access statistics of tablets, which horizontally partition objects by a range of keyspace – most often by users.

They work well with private data-sharing applications like social network applications; however, not with public data-sharing applications such as YouTube and Flickr, where data is accessed through various channels, not just through users (friends). Another limitation of existing solutions is that they do not explore the potential for compound, multi-attribute metrics to drive their replication actions. Lastly, not all future accesses can be predicted with the attribute-popularity-based replication systems; for example, an object with a new attribute that has never happened before cannot be proactively replicated to datacenters until the first request to those attributes is made. To address the shortcomings of the existing systems, we propose Acorn, an Attribute-based COntinuous partial geo-ReplicatioN system, which achieves lower cost and latency than existing systems with three design principles; a) Use the right attribute for each application, b) Use multiple attributes, and c) Use continuous replications.

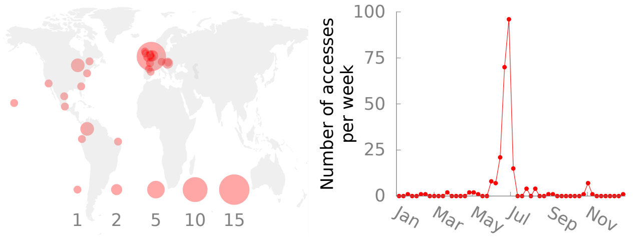
This paper makes the following contributions. (1) We describe Acorn – a partial geo-replication solution for large-scale, data-sharing applications. Acorn monitors object attribute access popularity for making replication decisions, using simple design principles that benefit both public and private data-sharing applications, thus lowering cost and latency. (2) We implement Acorn by extending Apache Cassandra [6]. (3) We evaluate Acorn with two types of data-sharing applications: a public data-sharing application with YouTube access traces and a private data-sharing application generated from a Yelp dataset [7]. This demonstrates up to 54.28% and 89.91% cost overhead reduction over existing, user-based systems or 37.73% and 90.90% latency overhead reduction.

II. Attribute-based data replication

To explore the design space of geo-replication solutions, we classify replication models as follows; a) Full replication: makes replicas to all datacenters. This is what most replicated systems do, including Cassandra, HBase, and MySQL Cluster. It has the highest replication cost and the lowest data access latency. b) No-active replication: stores data only in local datacenters. When a requested object is not in the local datacenter, it is fetched from a remote datacenter, and a local replica is made. Caching systems, including CDN systems, follow this model. It has the lowest cost and the highest latency. c) Partial replication: selectively makes remote replicas based on the access statistics of objects or metadata. d) Partial replication with future knowledge: replicates data only to where it will be accessed in the future. It is a theoretic model with both the lowest cost and the lowest latency.

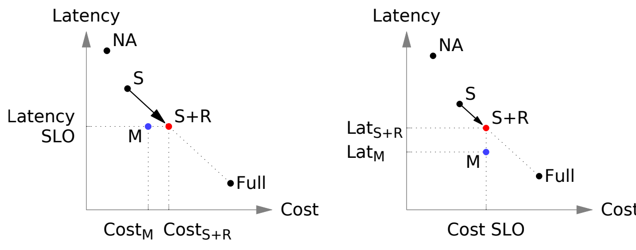
We use the following three design principles to achieve low-cost geo-replication and provide low-latency data accesses. **A. Use the right attributes for each type of application**. Different types of applications have different data access patterns. In private data-sharing applications, such as Facebook or Snapchat, objects are accessed mostly through friends, thus making "user" the best attribute to monitor and predict future accesses. On the other hand, public data-sharing applications, such as YouTube or Flickr, have diverse sources of accesses. For example, YouTube has the majority (63%) of its accesses from non-user-based (non-social) channels, like search and in-application navigation [8]. For these applications, "topic" is a better attribute for making replication decisions than "user". Figure 1 shows an example of topic's strong data access locality both geographically and temporally. Videos with the topic "Wimbledon" are mostly accessed from North and South America and Europe from Jan. 1st to June 7th, so they don't need to be replicated to Africa, Asia, or Australia.

B. Use multiple attributes. Multi-attribute-based replications have lower cost under a latency SLO (service level objective), or lower latency under a cost SLO, compared to



(a) Geographic locality (b) Temporal locality

Fig. 1: (a) Access locations of YouTube videos with the topic "Wimbledon" from Jan. 1st to June 7th, 2014. It is from our YouTube access dataset, which is explained in more detail in Section V. The size of each circle represents the number of accesses. "Wimbledon" becomes globally popular after June 7th, because the event starts at the end of June. (b) The number of accesses to the same videos per week in 2014.



(a) Cost comparison (b) Latency comparison

Fig. 2: Cost and latency comparison of single- vs. multi-attribute-based replications. S, M, S+R, and NA represent single-attribute based replication, multi-attribute based replication, single-attribute based replication plus extra random replicas, no-active replication, and full replication, respectively.

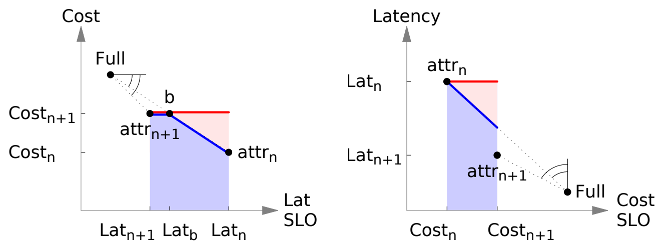
single-attribute-based ones, even when additional extra random replicas are used to remedy any limitations from the predictive use of the single attribute. Using a data attribute, you can make a better educated guess as to where a replica should be placed than placing replicas at random places.[1] In the same manner, using two attributes will most likely give you better replication decisions than using one attribute and adding extra random replicas; in Figure 2(a), the multi-attribute replication M has lower cost than the single-attribute with random replication solution ($S+R$). For example, replicating YouTube videos to Atlanta, Georgia, when either (a) they have the topic "tennis"[2] or (b) they are uploaded by the user "John Isner"[3], has a lower cost than replicating them to Atlanta when (a) they have the topic "tennis" or (c) with an extra 15% probability. In the same manner, using multiple attributes lowers latency, as shown in Figure 2(b). In general, we expect that M_n, a partial replication system making replication decisions with n attributes, has lower cost and latency than M_{n-1} plus extra random replicas.

C. Use continuous replications. To achieve further cost

[1]This is the rationale behind all history-based prediction systems, including the existing user-based partial replication systems. We acknowledge that anti-patterns can happen, where the geographic and temporal locality doesn't hold. An example is a repeating pattern of an attribute being popular briefly and not accessed at all for the duration of popularity monitor window. However, we believe those artificial patterns rarely happen with user-generated data.

[2]Atlanta has the highest number of USTA members per capita in the US.

[3]Isner played for University of Georgia and is a top-ranked tennis player in the US as of Dec. 2015.



(a) Cost under latency SLO (b) Latency under cost SLO

Fig. 3: Cost and latency reduction of continuous replications under SLO constraints. Attributes are labeled with $attr_1$ to $attr_n$ by their angles from the line extending the full replication. In Figure (a), the blue and red lines represent cost of continuous and non-continuous replication, respectively. The blue and the (red+blue) areas represent the sum of all costs of continuous and non-continuous replications, respectively. In Figure (b), the lines and areas represent latencies.

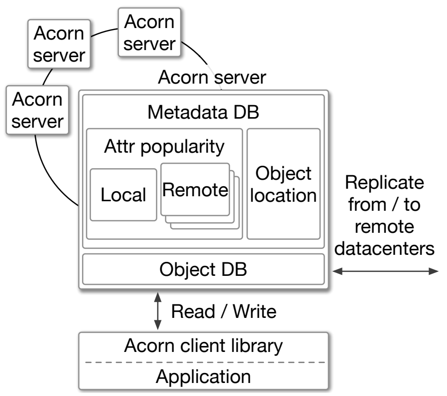


Fig. 4: Acorn system architecture

or latency reduction SLO constraints on top of the attribute-popularity-based access predictions, you can add as many extra random replicas as needed to meet your SLO constraints. This "continuous" replication allows a replication system to meet the SLOs without having to settle for suboptimal ones. In Figure 3(a), when you have a latency SLO Lat_n, you can use attribute $attr_n$ with cost $Cost_n$. As the latency SLO becomes smaller (as users want faster responses), you can add as many extra replicas as needed to meet the new SLO, as in the blue line of the figure until the cost becomes $Cost_{n+1}$, at which point you can use the next attribute $attr_{n+1}$. Without continuous replication, you would have to use attribute $attr_{n+1}$, resulting in a higher cost depicted in the red area in the figure. In the same manner, continuous replication brings latency reduction as depicted in the red area in Figure 3(b).

III. System Design

Acorn consists of Acorn servers and a client library that applications link against, as shown in Figure 4. Metadata DB consists of attribute popularity tables and object location tables.

Attribute popularity monitor. Acorn servers in each datacenter monitor popularity of each attribute independently from other datacenters using a sliding time window per each attribute, as shown in Figure 5. Popularity of an attribute item

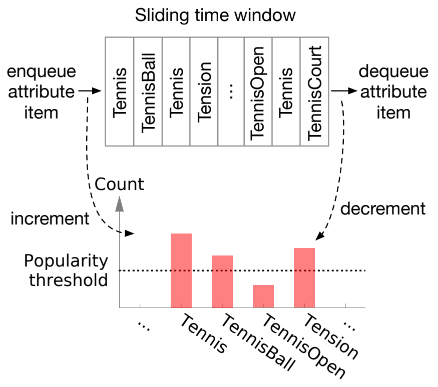


Fig. 5: Attribute popularity monitor

increases as it enters the sliding time window and decreases as it exits the window. Popularity counters are shared by all Acorn servers in a datacenter. A popularity threshold per each attribute determines whether an attribute item is popular or unpopular.

Popularity synchronization between datacenters. Acorn synchronizes attribute popularity metadata periodically with remote datacenters. An attribute popularity synchronizer node, selected from Acorn servers in a datacenter, proxies the synchronization, which is similar to what Cassandra's coordinator node does with the inter-datacenter communication. At every synchronization epoch, a synchronizer node calculates changes in popularity items since the previous epoch, i.e., newly popular items and items that are not no longer popular, and broadcasts them to remote datacenters.

Partial replications. Upon a write request, Acorn makes replication decisions independently in each datacenter based on its attribute popularity snapshots of remote datacenters. Unlike existing work (placement manager of SpanStore [2] or configuration service of Tuba [5]), Acorn does not involve any global component when making replication decisions, enabling low latency and high availability. After writing an object in a datacenter, Acorn updates the object location metadata, which is looked up when an Acorn server misses the object in the local datacenter and needs to fetch it from a remote datacenter.

A read request first looks for an object in a local datacenter; then, if it misses, it then fetches the object from the closest (the one with the lowest network latency) remote datacenter.

IV. Implementation

We implement Acorn by modifying Apache Cassandra [6]. We modify the write path so that the StorageProxy module writes objects to the local datacenter and to a subset of remote datacenters by checking their attributes against attribute popularity snapshots of remote datacenters. Since the number of replicas could be smaller than what full replication would do, *CL* (consistency level) is internally modified so that the number of acknowledgements for which a client waits does not exceed the number of replicas. StorageProxy, after writing an object to local object DB, updates the object location table for future on-demand fetch operations. Read serves a requested object from a local datacenter first, and, when it misses the object, it fetches the object from the closest remote datacenter. The request is notified to attribute popularity monitor asynchronously.

Acorn's popularity monitor uses Cassandra counters [9] internally to monitor the popularity of each attribute item. The counters are "eventually" consistent, just like any other writes and reads with their *CL*s less than *Quorum*. They serve Acorn's purpose very well for the two reasons; a) We prefer performance and scalabiltity over strong consistency. b) Within a local datacenter, which is the scope of the monitoring, inconsistencies requiring conflict resolutions rarely occur.

V. Evaluation

We evaluate Acorn by comparing it with existing, user-based systems in terms of cost and latency using two types of applications: a public and a private data-sharing application.

Experimental setup. The public data-sharing application workload was gathered from an extensive crawling of Twitter; we crawled tweets that have both YouTube links and coordinates using the Snowball crawling method [10] with periodic random seedings to prevent the crawler from being biased towards nodes with a high number of edges. The workload had 2.3 M YouTube videos, 833 K users, and 7.2 M accesses to the videos. For the private data-sharing application, we used a Yelp dataset [7], which had 1.1 M reviews, 253 K users, and a 956 K-edge social network graph. From the dataset, we built social network application requests; users check the latest reviews from their friends, just like Facebook users check the status updates of their friends [3]. We set up a simulated multi-datacenter testbed at Georgia Institute of Technology with real-world network latencies among datacenters.

Evaluation metrics. To compare replication models independently from application sizes or object sizes, we use cost and latency overhead as evaluation metrics. We define cost overhead CO of a replication model as a relative cost to the minimum cost, which is the cost of the no-active replication model (NA): $CO = (Cost - Cost_{NA})/Cost_{NA}$. In the same manner, we define latency overhead LO as a relative latency to the minimum latency: $LO = (Lat - Lat_{Full})/Lat_{Full}$. The partial replication with future knowledge model has a LO 0 and a CO 0, which is every partial geo-replication system's ultimate goal.

The right attributes for each application. Figure 6 demonstrates different types of applications have different attributes that replication decisions can be made best out of. In the public data-sharing application, data is accessed through various channels, and the "user"-based channel is not the best one for making replication decisions. In this case, "topic" captures the popularity of objects better than "user". In the private data-sharing application, where data is shared among friends, "user" is a natural choice. A natural follow-up question is how do you know which attributes are the best for your application? Sometimes they are trivial for application designers; for example, in social network applications, where checking status updates of friends is the most-used feature, it

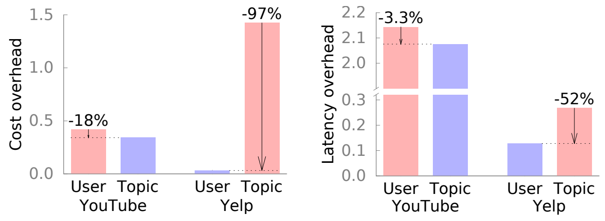


Fig. 6: Cost and latency overhead by attributes and application types

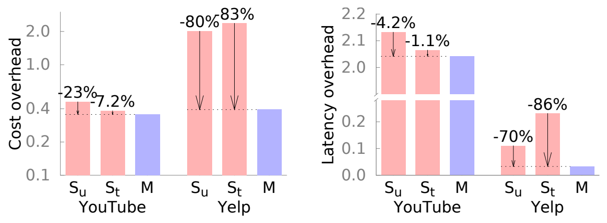
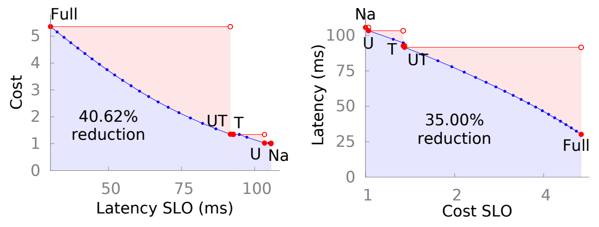


Fig. 7: Cost and latency overhead comparison of single- vs. multi-attribute-based replications. S_u and S_t use a single-attribute "user" and "topic", respectively, for making replication decisions. M uses both of them.



(a) Cost under latency SLOs (b) Latency under cost SLOs

Fig. 8: (a) Cost of the continuous and non-continuous replication system under latency SLO constraints using the public data-sharing application workload. Each of the blue and the red line represents the cost of continuous and non-continuous replication system under latency SLOs. Cost in the y-axis is plotted in the relative cost to the no-active replication system for an easy comparison. UT, T, U, and Na represent (user+topic)-based, topic-based, user-based, and no-active replication, respectively. The blue and the (red+blue) area represent the sum of all costs of the continuous and non-continuous replication system, respectively. (b) Latency of the continuous and non-continuous replication system under cost SLO constraints.

is trivial that "user" is the best attribute. Other applications can find their attributes with trial runs of Acorn with some initial part of the data. For example, with a trial run of only 3% of the public data-sharing application workload, we identified that "topic" was a better attribute than "user".

Multi-attribute based replication. In both public and private data-sharing applications, multi-attribute based replications outperform single-attribute based ones under SLO constraints, as shown in Figure 7: up to 23% and 83% cost overhead reductions in the public and private data-sharing applications, respectively, and up to 4.2% and 86% latency overhead reductions.

Continuous replication. Figure 8(a) compares the cost of a continuous replication system with the cost of a non-continuous replication system under latency SLO constraints. When the latency SLO is high, i.e., when the storage system has a good latency budget, the system can be configured to no-active replication model and run with the lowest cost. As the latency SLO decreases, the system needs to move from a no-active replication model to user-based to topic-based to (user+topic)-based, and finally to full replication model to meet the decreased latency SLO. A non-continuous replication system needs to follow the red line to meet the new latency SLO, resulting in a bigger cost jump than a continuous replication system, which needs only a small amount of extra replicas to meet the same SLO. Our experiments with the public data-sharing application workload demonstrate that the continuous replication system has an average cost reduction of 40.62% over the non-continuous replication system. In the same manner, the continuous replication system has an average latency reduction of 35.00% over the non-continuous replication system under cost SLO constraints as shown in Figure 8(b).

Overall, Acorn achieves up to 54.28% and 89.91% cost overhead reduction and 37.73% and 90.90% latency overhead reduction for the public and private data-sharing applications over existing partial replication systems.

VI. Conclusions and Future Work

We explore the attribute-based partial geo-replications in multi-datacenter environment with the three key ideas: a) Use the right attributes for each application, b) Use multiple attributes, and c) Use continuous replications. We evaluate the idea with our prototype implementation of Acorn and demonstrate considerable reductions in cost and latency with two types of data-sharing applications. Future work will investigate fault-tolerance issues arising from a reduced number of object replicas, deletion or offloading unpopular objects to cold storage, and partial replication under client-defined constraints, e.g., favoring private datacenters over public datacenters due to capital vs. operational expenditure.

References

[1] S. Kadambi *et al.*, "Where in the world is my data," in *Proceedings International Conference on Very Large Data Bases (VLDB)*, 2011.

[2] Z. Wu, M. Butkiewicz, D. Perkins, E. Katz-Bassett, and H. V. Madhyastha, "Spanstore: Cost-effective geo-replicated storage spanning multiple cloud services," in *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*. ACM, 2013, pp. 292–308.

[3] S. Traverso *et al.*, "Social-aware replication in geo-diverse online systems," *Parallel and Distributed Systems, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2014.

[4] D. B. Terry *et al.*, "Consistency-based service level agreements for cloud storage," in *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*. ACM, 2013, pp. 309–324.

[5] M. S. Ardekani and D. B. Terry, "A self-configurable geo-replicated cloud storage system," in *Symp. on Op. Sys. Design and Implementation (OSDI)*, 2014, pp. 367–381.

[6] A. Lakshman and P. Malik, "Cassandra: a decentralized structured storage system," *ACM SIGOPS Operating Systems Review*, vol. 44, no. 2, pp. 35–40, 2010.

[7] Yelp, "Yelp Dataset Challenge," 2015. [Online]. Available: http://www.yelp.com/dataset_challenge

[8] A. Brodersen, S. Scellato, and M. Wattenhofer, "Youtube around the world: geographic popularity of videos," in *Proceedings of the 21st international conference on World Wide Web*. ACM, 2012, pp. 241–250.

[9] Datastax, "Whats New in Cassandra 2.1: Better Implementation of Counters," 2014. [Online]. Available: http://www.datastax.com/dev/blog/whats-new-in-cassandra-2-1-a-better-implementation-of-counters

[10] S. Scellato, C. Mascolo, M. Musolesi, and V. Latora, "Distance matters: geo-social metrics for online social networks," in *Proceedings of the 3rd conference on Online social networks*, 2010, pp. 8–8.