Proceedings of 2013 IEEE 2nd International Conference on Cloud Networking (CloudNet), 11-13 November 2013, San Francisco, CA, USA, which should be cited to refer to this work.
DOI: 10.1109/CloudNet.2013.6710573

A Cloud Data Center Optimization Approach Using Dynamic Data Interchanges

Efstratios Rappos
Institute for Information and Communication Technologies,
Haute Ecole d'Ingénierie et de Geston du Canton de Vaud,
University of Applied Sciences of Western Switzerland,
Route de Cheseaux 1
CH-1401 Yverdon-les-Bains
Switzerland
E-mail: efstratios.rappos@heig-vd.ch

Stephan Robert
Institute for Information and Communication Technologies,
Haute Ecole d'Ingénierie et de Geston du Canton de Vaud,
University of Applied Sciences of Western Switzerland,
Route de Cheseaux 1
CH-1401 Yverdon-les-Bains
Switzerland
E-mail: stephan.robert@heig-vd.ch

Rudolf H. Riedi
Department of Telecommunications,
Ecole d'ingénieurs et d'architectes de Fribourg,
University of Applied Sciences of Western Switzerland,
Bd de Pérolles 80, CP 32
CH-1705 Fribourg
Switzerland
E-mail: rudolf.riedi@hefr.ch

***Abstract*—Distributed data center architectures have been recently developed for a more efficient and economical storage of data. In many models of distributed storage, the aim is to store the data in such a way so that the storage costs are minimized and increased redundancy requirements are maintained. However, many approaches do not fully consider issues relating to delivering the data to the end user and the associated costs that this creates. We present an integer programming optimization model for determining the optimal allocation of data components among a network of Cloud data servers in such a way that the total costs of additional storage, estimated data retrieval costs and network delay penalties is minimized. The method is suitable for periodic dynamic reconfiguration of the Cloud data servers, so that the when localized data request spikes occur the data can be moved to a closer or cheaper data server for cost reduction and increased efficiency.**

***Index Terms*—cache storage; grid computing; mathematical programming.**

I. Introduction

Distributed data centers in the Cloud constitute a recent development aimed at increasing data availability, reducing costs and taking advantage of economies of scale they offer. The economics of Cloud data centers have been extensively studied [1], [2], [3], [4] with the aim to reduce the infrastructure costs via economies of scale or efficient use of the hardware infrastructure.

At the same time, mathematical modeling methods aiming to make better use of distributed data centers have been developed, such as [5], [6], [7] who use combinatorial optimization methods to determine the best allocation of virtual severs to physical target servers, or virtual resources to actual physical resources, all in an effort to reduce costs. This has been followed by techniques targeting geo-optimization [8] where the geographical location of the servers or customers are taken into consideration when trying to optimize distributed Cloud services.

We consider the problem of geo-distributed users accessing data also located on geo-distributed Cloud data servers. We provide a mathematical programming approach which determines whether data from one server should be temporarily duplicated on another server which is located nearer to the bulk of the customers currently requesting this data. In doing so we can achieve better performance, indicated by a lower 'cost' in the model. This cost can represent many possible metrics, such as low latency or genuine data transfer costs.

The inspiration for this approach comes from the area of predictive caching [9], [10], [11] and its applicability to help correct imbalances between where the data is stored and where the users are located.

We begin by describing the underlying mathematical optimization model used to determine if cost savings can be achieved by temporarily moving objects from one server to another. This is followed by a demonstration of the potential benefits using a synthetic workload simulation.

II. A mathematical modeling approach for optimized data delivery

In this section we present the mathematical model for the data delivery and server configuration problem. The setup of the model, described by an integer program, is as follows.

We consider D geographically dispersed data centers and N also geographically dispersed users who are accessing data from the servers. For simplicity, we assume that a total of K data objects are held in the servers (for example, these can be the K most frequently requested objects). Each object exists in only one copy which is located in only one of the servers, but the objects can have different sizes.

When a user requests a particular object, it will be fetched from the sever which holds it at a cost of c units (for example, the cost of latency or delay in retrieving the object, but this cost can also vary for each server). In our experimentation we assume that the main factor affecting the cost is the distance between user and data center, although the method works for any cost metric.

Our optimized data delivery framework operates as follows: we assume that each data server has a small area of additional storage where temporary copies of objects can be stored (in essence, a cache storage space). The server can then decide, for example, depending on the access pattern of the received requests and/or the location of the users and in order to achieve increased performance, to keep a local copy of some objects which are located in other servers. So, if for example many users in one geographical location frequently request objects that are located far away, a local server may temporarily duplicate these objects, so future requests for these objects will be carried out at a smaller cost.

As with all predictive caching and prefetching models, a key component is an estimation of the pattern of future user requests so that the caching can be designed accordingly. In our mathematical model the predictive element is described by the probabilities p_{uj} which represent the probability of a data object i will be requested by a user u in the near future. In our experimentation we use a Zipf distribution fit to estimate these probabilities from historic data. The choice of the Zipf distribution is justified by the fact that this heavy tail distribution is often present in the access patterns of real-life user data requests [12], [13] and is also commonly used for the generation of synthetic workload traces [9].

The objective of the mathematical model is to determine the objects, if any, that need to be copied from one server to the cache area of another, in such a way that the total expected cost of delivering the data to the users is minimized. This set of objects is determined by:

1) the probabilities of an object being requested by users in a particular geographic location;
2) where the data are currently stored;
3) the relative cost difference of retrieving the objects from their current location versus a closer location; and
4) the relative cost to copy the objects from one server to another.

The integer programming model to optimize this operation is shown below:

$$\min \left\{ \sum_{u=1}^{N} \sum_{d=1}^{D} \sum_{i=1}^{K} p_{iu} c_{ud} x_{id} + \sum_{d=1}^{D} \sum_{i=1}^{K} C_{id} x_{id} \right\} \quad (1)$$

subject to

$$\sum_{\substack{i=1 \\ i \neq i^*}}^{K} s_i x_{id} \leq Z, \text{ for all } 1 \leq d \leq D \quad (2)$$

$$\sum_{d=1}^{D} x_{id} \geq 1, \text{ for all } 1 \leq i \leq K \quad (3)$$

$$x_{id} \in \{0, 1\} \quad (4)$$

The binary decision variables x_{id} will take the value 1 if the object i should be obtained from data center d, copying it to the cache of d if necessary, and 0 otherwise. The cost

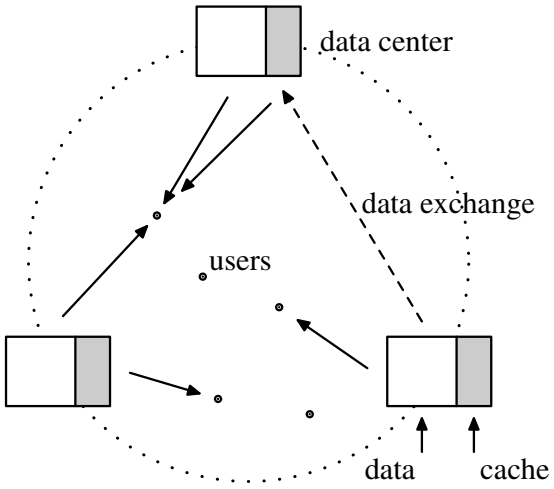

Fig. 1. Simulation setup.

c_{ud} represents the cost of obtaining an object from data center d (e.g., the distance between the user u and the data center d that holds i or a copy of i). The cost C_{id} denotes the cost of copying data object i from its default server to data server d; this cost will be zero if object i already exists on that server. The first summation in (1) relates to the expected (multiplied by the probability of object i being requested) cost of retrieving the object from data center d. The second summation relates to the cost of moving the objects between data centers as necessary.

The constraint (2) specifies that the combined size s_i of all the objects copied to data center d cannot exceed the capacity Z of the server cache area. In the summation, however, we must exclude objects i^* that are physically located in data center d, as it is not necessary to copy an object residing in a data center to the cache of the same data center. The second constraint (3) ensures that each object is available from at least one data center.

The values of x_{id} in the solution of this model will determine the data objects to duplicate across servers. We note that the mathematical model is NP-hard and as a result the solution time is expected to grow exponentially with the input parameters; therefore for arbitrarily large parameter values its use would become impractical. Nonetheless, it can be of practical use because (i) for reasonable values of the parameters (1000s) the solution is obtained very quickly, generally in under a second; (ii) the model input size can be kept small without loss of applicability, for instance the number of users N can represent groups of nearby users, or the data objects K can denote the K most frequently requested objects; and (iii) the model does not need to be solved live where a quick solution is essential, but instead solved offline using recent access logs to calculate a cost-reducing data exchange strategy to be applied subsequently.

III. Experimentation

A series of simulations were carried out to assess the benefits of the proposed framework and the resulting cost savings, where we compared the optimized data-exchange method to retrieve the requested data objects against simply obtaining the data from the server which owns them.

In the experimental testbed, we placed D data centers at equal distances along the circumference of a circle, and N users were placed randomly inside the circle. The costs were then determined by the euclidean distance between the user and each data center and between the data centers themselves (Fig. 1). A set of K objects was generated and assigned at random to one of the data centers.

The ranges of the parameter values used were as follows: we used 3, 5 or 10 data centers; 20, 100, 500 or 1000 data users; 100, 500 or 1000 data objects; and 1500 or 3000 for the capacity of the cache for each server. The sizes of the individual data objects were integers chosen uniformly in the range 50–150.

A sequence of 2000 data requests was then generated independently for each user. This was done using a Zipf distribution, but for each user the order of the objects was randomly permuted. So, for example, for user 1 object 1 may be the most popular, followed by objects 2, 3, whereas for user 2, object 14 may be the most popular, followed by objects 6 and 35. A random Zipf distribution parameter between 1.0 and 3.0 was used for each user.

The algorithm first proceeds to fit a Zipf distribution, for each user, based on a sequence of their first 1000 data requests (representing the historic access pattern) using a maximum likelihood estimation. This is done so that the Zipf parameter is derived from the data, simulating a real-life implementation scenario. We then use the probability distribution function to calculate the values of the probabilities p_{iu} per user.

The simulation is carried out on the remaining 1000 data requests per user. Each iteration begins with the optimization step where the integer program is solved to determine the data interchanges and then calculate the data retrieval costs in two ways: directly from the original data server, or from a local cache if the object has been duplicated. The cost of moving data between servers is added on, regardless if the local copy was eventually needed or not. Once a local cached copy was created it can remain there at no additional cost, although in practice, as with all caches, a mechanism to flag it as expired would be required should the original object is modified.

We note that for single user it is always better to obtain a data object directly rather than copying it to a local cache and retrieving it from there (triangle inequality), but in the presence of many users this is not true because the data is only copied once but will benefit several users.

One could, if needed, adjust the probabilities p_{iu} during the course of the simulation by recalculating them using a more recent data request series. This was not needed as the access pattern of our generated data did not vary with time.

IV. Results

The simulation was implemented in Java using the IBM ILOG CPLEX 12.4 library to solve the optimization model.

Tables I, II and III present the results of the computation experimentation for 3, 5 and 10 data centers respectively. The table columns denote, respectively, the number of users N, the number of data objects K, the capacity of the cache Z, the total cost if data are retrieved from the default server and the corresponding cost if data exchanges between servers are allowed. The percentage difference between the two costs is also reported. The last column denotes the total number of times objects were moved between servers during the simulation.

We note that the optimized method always yields a lower cost than the default method, with a reduction varying from 54% to 98%, and the average percentage for 3, 5 and 10 data centers being 80.6%, 79.8% and 80.8% respectively.

The instances with the smaller number of users showed the largest variations in performance. This can be explained by the fact that the user locations, data sequences and data-to-server allocations were random and therefore, with a small number of users, their location with respect to the servers containing the objects they require can impact more on the data retrieval costs and savings. When the number of users is large there is less chance for extreme variations, but the cost savings are still significant: a consistent cost reduction to 75–80% (a saving of 20–25%) is observed.

Overall, using the proposed method results in an average reduction of the cost of around 20% compared to the default data retrieval method.

V. Future work

The optimization model for Cloud data center performance improvement by dynamic data exchanges between the servers presented is very flexible and can be used in a variety of setups. However, more experimentation is required to determine what the percent improvement will be on a real system. In our analysis we used the distance between the user and server as a measure of cost of the transfer, but in reality the cost structure is more complex. Similarly, the user access patterns may be more varied.

In the formulation of the mathematical model the decision variables x_{id} do not correspond to individual users. This means that the optimal server to be used to retrieve a particular object is determined by the location and costs of the users as a whole, adjusted by the likelihood that each user will need this specific data object.

Although it is possible to formulate the model—and obtain better results—in a way that the decision-making is applied to each user individually, this this would be impractical because the number of decision variables would increase dramatically resulting in a model that cannot be solved in reasonable time. This will be possible however for small problem sizes.

TABLE I
NUMERICAL EXPERIMENTATION, 3 DATA CENTERS

users N	*data* K	*cache* Z	*cost default*	*cost optimized*	*objects moved*
20	100	1500	2226	2159 (97%)	1012
20	100	3000	2354	1740 (74%)	7933
20	500	1500	2290	1870 (82%)	7621
20	500	3000	2271	1634 (72%)	7732
20	1000	1500	2129	2012 (94%)	1447
20	1000	3000	2245	1558 (69%)	9014
100	100	1500	11267	10821 (96%)	6280
100	100	3000	11542	8392 (73%)	49190
100	500	1500	11143	8189 (73%)	44805
100	500	3000	11122	10348 (93%)	12835
100	1000	1500	11227	10999 (98%)	3325
100	1000	3000	11380	9170 (81%)	32062
500	100	1500	57746	41270 (71%)	241202
500	100	3000	57412	40832 (71%)	255437
500	500	1500	57017	47850 (84%)	139755
500	500	3000	58036	40264 (69%)	255339
500	1000	1500	57522	46847 (81%)	152667
500	1000	3000	56802	43545 (77%)	195552
1000	100	1500	112522	103347 (92%)	136965
1000	100	3000	112747	87779 (78%)	372329
1000	500	1500	113374	82852 (73%)	464904
1000	500	3000	114190	80713 (71%)	502346
1000	1000	1500	113463	79480 (70%)	497532
1000	1000	3000	112049	105924 (95%)	94292
			overall average	*80.6%*	

TABLE II
NUMERICAL EXPERIMENTATION, 5 DATA CENTERS

users N	*data* K	*cache* Z	*cost default*	*cost optimized*	*objects moved*
20	100	1500	2197	2083 (95%)	2197
20	100	3000	1983	1851 (93%)	2824
20	500	1500	2407	1592 (66%)	9481
20	500	3000	2259	2196 (97%)	1315
20	1000	1500	2279	1983 (87%)	3547
20	1000	3000	2336	1837 (79%)	6835
100	100	1500	11334	9229 (81%)	36244
100	100	3000	10414	9784 (94%)	8304
100	500	1500	10997	10439 (95%)	7678
100	500	3000	11021	8991 (82%)	39602
100	1000	1500	11392	9594 (84%)	25487
100	1000	3000	12160	7433 (61%)	60848
500	100	1500	57345	44437 (77%)	254412
500	100	3000	56853	43380 (76%)	245749
500	500	1500	57375	44136 (77%)	192894
500	500	3000	58244	39778 (68%)	247032
500	1000	1500	56908	41019 (72%)	212399
500	1000	3000	56823	42269 (74%)	206193
1000	100	1500	112847	93451 (83%)	377702
1000	100	3000	115646	80146 (69%)	495847
1000	500	1500	113487	78272 (69%)	487868
1000	500	3000	113136	79748 (70%)	477817
1000	1000	1500	112873	96975 (86%)	242417
1000	1000	3000	113494	87506 (77%)	344443
			overall average	*79.8%*	

TABLE III
NUMERICAL EXPERIMENTATION, 10 DATA CENTERS

users N	*data* K	*cache* Z	*cost default*	*cost optimized*	*objects moved*
20	100	1500	2395	1291 (54%)	11995
20	100	3000	2215	1863 (84%)	9345
20	500	1500	2286	1747 (76%)	9612
20	500	3000	2145	1868 (87%)	9996
20	1000	1500	2308	2039 (88%)	3426
20	1000	3000	2266	1832 (81%)	8011
100	100	1500	11351	10776 (95%)	10625
100	100	3000	11116	10171 (91%)	14480
100	500	1500	11533	8791 (76%)	43167
100	500	3000	11298	8455 (75%)	42271
100	1000	1500	11147	9348 (84%)	49430
100	1000	3000	11119	9203 (83%)	48526
500	100	1500	56884	41757 (73%)	223788
500	100	3000	56079	47955 (86%)	221087
500	500	1500	56332	49817 (88%)	136452
500	500	3000	56961	43951 (77%)	241274
500	1000	1500	57777	41914 (73%)	237158
500	1000	3000	56511	51161 (91%)	96551
1000	100	1500	114072	79322 (70%)	494210
1000	100	3000	112755	88594 (79%)	462783
1000	500	1500	112548	98379 (87%)	379342
1000	500	3000	112609	97364 (86%)	367133
1000	1000	1500	113065	89588 (79%)	486833
1000	1000	3000	113574	85424 (75%)	424589
			overall average	*80.8%*	

VI. CONCLUSION

An optimization approach for cost minimization when a number of geo-distributed users retrieve data objects stored in geo-distributed Cloud servers was presented. The mathematical model determines if data objects residing in one server need to be duplicated temporarily onto a temporary cache area of another server which is located closer to the users currently requesting the data. This is influenced by a set of data transfer costs determined by the distance between users and servers and between servers themselves, and a predictive component which uses historic data access information to estimate the likelihood that particular data objects will be needed in the near future.

The performance of the optimized method was evaluated in a series of simulations across a range of parameters using synthetic workloads. The use of the optimized data retrieval model resulted in an average cost improvement of around 20% which is very promising, and further work is underway to evaluate its merits in several real-life scenarios.

The proposed model is very flexible with a number of adjustable parameters and can provide valuable information about the locality of data requests and simple temporary data duplication or caching measures that can improve performance and reduce data transfer costs.

REFERENCES

[1] L. Ganesh, H. Weatherspoon, T. Marian, and K. Birman, "Integrated approach to data center power management," *IEEE Transactions on Computers*, vol. 62, no. 6, pp. 1086–1096, June 2013.

[2] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: research problems in data center networks," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 1, pp. 68–73, January 2009.

[3] L. Rao, X. Liu, L. Xie, and W. Liu, "Minimizing electricity cost: Optimization of distributed internet data centers in a multi-electricity-market environment," in *IEEE INFOCOM 2010 proceedings*, 2010, pp. 1–9.

[4] D. Kondo, B. Javadi, P. Malecot, F. Capello, and D. P. Anderson, "Cost-benefit analysis of cloud computing versus desktop grids," in *IEEE International Symposium on Parallel and Distributed Processing, IPDPS*, 2009, pp. 1–12.

[5] B. Speitkamp and M. Bichler, "A mathematical programming approach

for server consolidation problems in virtualized data centers," *IEEE Transactions on Services Computing*, vol. 3, no. 4, pp. 266–278, October-December 2010.

[6] A. Beloglazov and R. Buyya, "Energy efficient resource management in virtualized cloud data centers," in *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, ser. CCGrid 2010. IEEE Computer Society, 2010, pp. 826–831.

[7] C. Papagianni, A. Leivadeas, S. Papavassiliou, V. Maglaris, C. Cervelló-Pastor, and Á. Monje, "On the optimal allocation of virtual resources in cloud computing networks," *IEEE Transactions on Computers*, vol. 62, no. 6, pp. 1060–1071, June 2013.

[8] L. Jiao, J. Li, T. Xu, and X. Fu, "Cost optimization for online social networks on geo-distributed clouds," in *Network Protocols (ICNP), 2012 20th IEEE International Conference on*, 2012, pp. 1–10.

[9] G. Yadgar, M. Factor, K. Li, and A. Schuster, "Management of multilevel, multiclient cache hierarchies with application hints," *ACM Transactions on Computer Systems*, vol. 29, no. 2, p. Article 5, May 2011.

[10] E. Rappos and S. Robert, "Predictive caching in computer grids," in *Proceedings of the 2013 13th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, ser. CCGrid 2013. Delft, Netherlands: IEEE Computer Society, May 2013, pp. 188–189.

[11] G. Chockler, G. Laden, and Y. Vigfusson, "Design and implementation of caching services in the cloud," *IBM Journal of Research and Development*, vol. 55, no. 6, pp. 9:1–9:11, 2011.

[12] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and Zipf-like distributions: Evidence and implications," in *Proceedings of the IEEE INFOCOM 99 Conference on Computer Communications: the Future is Now*, vol. 1–3, 1999, pp. 126–134.

[13] A. Clauset, C. R. Shalizi, and M. E. J. Newman, "Power-law distributions in empirical data," *SIAM Review*, vol. 51, no. 4, pp. 661–703, December 2009.