Towards a Leaner Geo-distributed Cloud Infrastructure

Iyswarya Narayanan[‡], Aman Kansal[†], Anand Sivasubramaniam[‡],
Bhuvan Urgaonkar[‡], Sriram Govindan[†]
[‡] *Pennsylvania State University* [†]*Microsoft Corporation*

Abstract

Modern cloud infrastructures are geo-distributed. Geo-distribution offers many advantages but can increase the total cloud capacity required. To achieve low latency, geo-distribution forfeits statistical multiplexing of demand that a single data center could benefit from. Geo-distribution also complicates software design due to storage consistency issues. On the other hand, geo-distribution can lower costs through eliminating redundancies at individual sites or exploiting regional differences in energy prices. We discuss several factors that influence geo-distributed capacity provisioning, and quantify latency, availability, and capacity trade-offs that emerge. We describe open research challenges in designing software that efficiently uses cloud capacity.

1 Introduction

Most large online services are geo-distributed. Cloud infrastructures make it easy for any application to become geo-distributed. Many techniques have been developed for geo-distributed application design such as distributed failure detection and request re-routing [41, 48, 7, 38, 8], geo-distributed storage consistency management [36, 46, 15, 16, 14, 12, 47, 31], data placement [1, 49, 43], and distributed consensus [11, 28, 2].

However, geo-distribution complicates allocation of compute resources compared to building a single large data center. In this paper we explore multiple factors that affect geo-distributed capacity provisioning and discuss open challenges in realizing efficient resource allocation.

Capacity implications of geo-distribution: Arguably, the most compelling reason to go geo-distributed is latency. Geo-distribution creates a point of presence close to clients, thereby reducing latency. Latency is critical for interactive online applications, and directly impacts user experience and revenue [21, 30, 9, 22]. Latency expectations for cloud services are becoming more stringent due to increasing interactive features in cloud services such as auto-complete of search queries, cloud hosted mobile games, and growth in cloud hosted versions of traditionally client-based applications such as content editing. Trends such as the use of edge data centers and hosting VMs at cellular base stations [44] will only lead to greater geo-distribution.

Latency gains come at the cost of potentially larger total data center capacity compared to a single very large data center. Consider the demand over time seen at six data center locations, where each is serving the clients nearest to itself (Figure 1). The demand is based on measured server utilization at one data center location hosting Microsoft's production applications spanning tens of thousands of servers. It is re-played in the local time-zone for each data center, and the peak magnitude is scaled in proportion to the population size [10] served by that location. The peak of the sum of these demands (single data center capacity) is smaller than the sum of the peaks of these demands (capacity required for geo-distributed data centers). Table 1 shows the increase in capacity compared to a single data center, assuming the geo-distributed data centers are located at the publicly available US data center locations of Google [26], Amazon [6] and 6 of Microsoft's [1].

Locations for	Excess over single data center
Microsoft	20.2%
Google	16.4%
Amazon	16.2%

Table 1: Excess capacity due to geo-distribution.

A second advantage is that geo-distribution can safeguard against failures that backup resources (e.g., diesel generators) and isolation techniques such as availability

[1] We focus on the US since data center deployments across the world vary quite a bit in density per unit population and modeling demand in proportion to population does not give a good estimate of demand across regions.

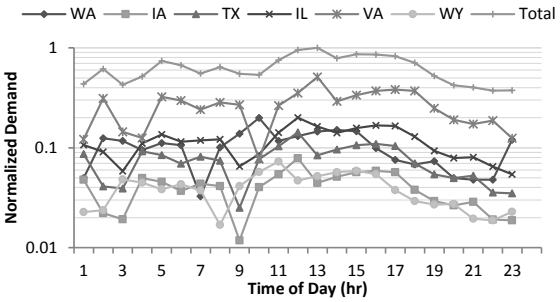

Figure 1: Population based demand projection at 6 of Microsoft data center locations in the US.

zones [4] at a single site cannot: earthquakes, civil unrest, or severe weather issues [23, 39, 3, 20]. Even at a given level of availability, geo-distribution offers new design options by trading off local redundancy costs for spare capacity [29, 19].

Availability gains also come at the cost of extra spare capacity added to make up for capacity lost in a failure. If all data centers were equally sized and served equal demand, then 6 data centers would require an excess of 1/6 of the total capacity as spare, spread across data centers remaining after a failure. This leads to a total of 20% excess spare capacity. The actual excess will be higher because not all data centers are equally sized, and failure of a larger data center necessitates larger spare at remaining ones.

A third reason to use geo-distribution is to exploit regional differences in energy prices or availability of renewable energy [45, 33, 42, 35, 34, 17, 18]. But to shut down servers when energy is expensive or unavailable from green sources at one location, we would have to add additional servers at another location. If the entire workload is to be moved, this can lead to 100% excess.

Problem: We see that geo-distribution leads to excess capacity. Capacity is expensive, and its cost needs to be weighed against the advantages of lower latency, higher availability, and energy savings. Our goal in this paper is to determine the lowest excess capacity required. Determining the optimal capacity is non-trivial. The spare created due to latency and availability reasons is not disjoint and some of it may overlap, depending on latency requirements after a failure. While current server and network costs may be too high to justify buying excess capacity for reduced energy price, green incentives, or customer goodwill, the excess capacity already available due to latency and availability objectives could be used for that purpose. Capacity allocation also depends on factors such as reduced the cost per unit capacity due to reduced local availability requirements [29] and revenue from sale of excess capacity on the spot market [5].

Optimized capacity allocation also impacts software design because existing geo-distribution methods do not use capacity efficiently. For instance, routing client requests to the next-nearest data center after a failure may not allocate clients in the right proportion from a capacity provisioning perspective. Distributed storage methods may need to account for capacity provisioning decisions when determining placement of data replicas. We discuss such open challenges in making geo-distributed software more efficient in terms of capacity used.

2 Minimizing Waste

We want to understand the minimum capacity required to realize the advantages of geo-distribution.

Consider n geo-distributed data centers, located based on proximity to clients and availability of power, network, land or other resources. The clients are spread across m locations where each location could be a small geographic unit within which network latency variation is insignificant. Response time consists of the service time within the data center and the network latency outside the data center. Assuming that the service time (including queuing delay) is the same regardless of which data center is used, we focus on network latency. Let l_{ij} represent the latency when data center at location j serves a client at location i. Suppose client demand over time is known based on history, denoted by $\{d_i(t)\}$ for clients at location i in time slot t, where $t \in \{1, ..., T\}$ and T is a period over which demand repeats (e.g., T=24 hours if demand has a diurnal pattern).

We wish to determine the capacity c_j for the data center at location j, such that there is sufficient capacity to serve all of the client demand within a latency target, l, when no data centers have failed and within latency l' ($l' \geq l$) when up to one data center has failed, while minimizing total cloud capacity. We assume each data center is large enough to benefit from economies of scale applicable to multi-megawatt data centers and hence do not model any extra fixed costs aside from the capacity.

The solution also requires specifying how clients are allocated to different data centers. Let $f_{ijk}(t)$ denote the demand originating from client i that will be serviced by data center j when the k^{th} data center has failed, during time slot t. Here, $k = 0$ represents no data center failed, and $k = \{1, ..., n\}$ represents one of the n data centers having failed.

Objective: The optimization objective thus becomes:

$$\min \sum_{j \in \{1,...,n\}} c_j$$

Constraints: At any given time, demand from a client location should be serviced by data centers that are

within the latency target. Before failure, this can be expressed mathematically as:

$$\sum_{j:l_{ij}\leq l} f_{ij0}(t) \geq d_i(t), \forall i,t. \quad (1)$$

where the summation represents the total demand from client i allocated to data centers that are within a network latency of l from this client location.

After failure, the constraint is similar, except that the failed data center is not used in the summation and the latency constraint could be less stringent.

$$\sum_{j:j\neq k,l_{ij}\leq l'} f_{ijk}(t) \geq d_i(t), k>0, \forall i,t. \quad (2)$$

The capacity at each data center should be sufficient to serve all of the demand allocated to it in any time slot. Before failure, this implies:

$$c_j \geq \sum_i f_{ij0}(t) \quad \forall j,t, \quad (3)$$

After each possible failure, this implies:

$$c_j \geq \sum_i f_{ijk}(t) \quad k>0, k\neq j, \forall j,t. \quad (4)$$

The above optimization problem is a linear program (LP) and can be solved using any LP solver.

We use the above optimization to explore some of the interesting latency, availability and capacity trade-offs.

Latency and Availability: As latency constraints become tighter, we expect greater spare capacity to be required, since more capacity would have to be placed closer to each client, leading to lower multiplexing of demand across client locations. Figure 2 shows the excess capacity required, compared to a single very large data center, for different post-failure latency constraints with up to one data center failure. The pre-failure latency (l) is set to the latency achieved when each client is mapped to the nearest data center, and is different for each set of data center locations.

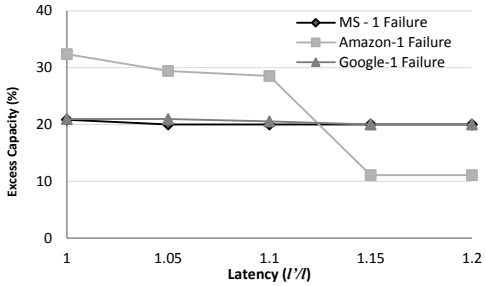

Figure 2: Excess capacity required for supporting one data center failure.

Interestingly, the excess capacities required jointly for latency and availability for each set of locations are similar to the excess capacity present anyway to reduce latency, shown in Table 1. For Amazon locations, increasing l' to above $1.1l$ allows more data centers to be within range giving a large reduction in excess capacity at that latency. Prior works have shown that tighter latency constraints at a given availability led to an increase in the number of locations but not in total capacity [19], because they ignored the statistical multiplexing of demand across locations over time. This is significant since the increase in capacity can be leveraged for other uses such as higher availability and energy cost reduction.

Energy Savings Opportunity: The geo-distributed system has idle capacity at any given time since the demand varies over time, both at each data center location and also globally. Figure 3 shows the idle capacity at different times of the day summed over Microsoft's data center locations. This capacity is available for use on the spot market for cloud resources [5], or for shifting load latency tolerant load to a location with lower energy price or greater renewable availability. The load shifting opportunity depnds on how the spare is distributed across multiple sites.

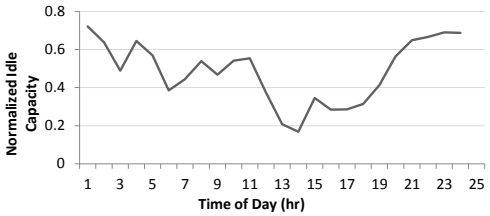

Figure 3: Capacity available for energy optimizations, normalized with respect to total capacity.

Software Impact: Figure 4 shows the excess capacity required without exploiting time of day effects, i.e., assuming demand is always at its peak at each location. Significant excess capacity results. However, using efficient capacity complicates the allocation of client traffic. If peak based capacity is provisioned, the client demand allocations to different data centers stay constant over time. Re-allocation is required only in case of a failure. But when time of day effects are exploited, client demand requires re-allocation even without failures, as demands in different regions change.

Figure 5 shows the optimal demand allocations at three different times of the day for Pennsylvania clients. We see that the demand shifts between Virginia and Iowa data centers. One of the mechanisms used by geo-load balancers to allocate clients to a data center is to set different DNS entries for the geo-distributed application. In this method, DNS entries may have to be changed over

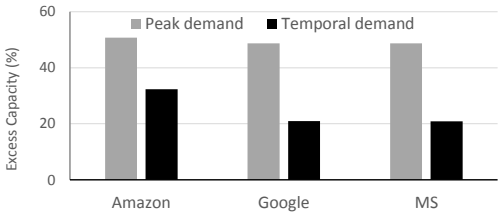

Figure 4: Excess capacity assuming peak demand and when exploiting temporal variations.

time.

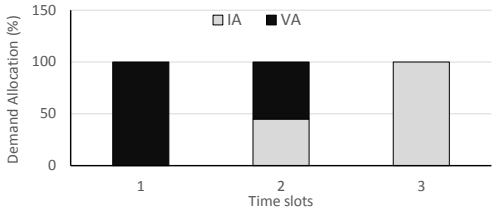

Figure 5: Demand allocations to two data centers for Pennsylvania for three time slots.

3 Open Challenges

3.1 Infrastructure Design

Optimization: The optimization above can be expanded to consider additional factors. Data center construction takes time and demand curves over time are often not available for the time horizon for which capacity is built. Errors in demand projection need to be accounted for. While in today's clouds more than one data center failing at the same time is highly unlikely, multiple failures may become more practically relevant when the number of geo-distributed data centers grows much larger. The capacity required for a given demand can also be optimized by running servers at higher utilization and increasing the queuing delays inside the data center, when tolerable. The optimization could thus obtain a more efficient solution when considering queuing delays and network latencies jointly.

Heterogeneity and Right-Sizing: One size may not fit all applications, because multiple cloud applications share the infrastructure. A key opportunity is to allow finer grained control of latency and availability objectives. Different applications may have different latency requirements, or even multiple latency objectives for different percentiles of the client population.

Availability requirements also vary across applications depending on factors such as whether an application faces external customers or is for internal use. This opens up opportunities to also consider heterogeneous data center infrastructures, where some portion of the capacity is highly resilient and provisioned with power backups, redundant network connectivity, and highly reliable hardware, but other portions of the infrastructure economize on redundancy, environmental controls, or hardware quality. Cost and availability are not linearly related and not all points in the trade-off space may be practical to build given available technology. A few classes of such data centers are likely to suffice to meet a wide range of requirements but determining the most appropriate cost-reliability trade-offs is an open problem.

Spatio-temporal variations: Failures do not necessarily happen when demand is at its peak. Since the peak only happens for a small portion of the time, we could provision for meeting availability objectives only for a high percentile of the time and reduce overall capacity. Failures and their impact may also be time dependent. Failures may be more likely in winter months. A failure during non-work hours may take longer to recover since the right staff may not be on-site.

Not all data centers may be equally likely to fail. Spatial differences such as varying susceptibility to severe weather, nearness to earthquake zones, network backbone redundancy in the region can make certain locations much more reliable than others. Exploiting these variations offers an opportunity to further optimize both the total capacity provisioned and its distribution across sites. While nearness to clients may force us to place data centers in a less reliable region, the backup capacity may be preferrentially placed at more reliable sites.

3.2 Software Design

For the infrastructure to be efficient, it is essential that software be designed to use it efficiently. Several opportunities exist to improve geo-distributed software design.

Request Routing: Existing geo-load balancing solutions are largely designed to route client traffic to the nearest available data center. They may rely on maintaining Internet latency maps and set DNS entries in every regions to point to the nearest data center, or they may rely on IP anycast [40] to automatically find the route from the client to the nearest available application endpoint. As we saw, the nearest data center may not lead to the most efficient capacity allocation. Especially after a failure, routing to the nearest data center will not make efficient use of available capacity. Compute capacity allocation information is typically not something that the network can easily access or make use of even if available. Application layer intervention will likely be needed to ensure that clients are routed to the correct data center, either through logic hosted in edge data centers or even at the client as part of the application. If coordination be-

tween the application and cloud infrastructure is needed to realize the correct demand allocations, new APIs to facilitate that will be needed. Cost structures that guide applications to behave efficiently are also important in this context.

State Replication: Many geo-replication systems save a copy of the state at a remote site [12]. If the objective of the replication is only to improve data durability and protect it against data destruction at the primary site, replicating to any remote site suffices. However, if the data will be actively used by the application from the replica site, it is essential to make sure that the replicas exist at the same site where the compute capacity will be allocated after a failure. This allocation may depend on time of day since availability of spare capacity changes with temporal variations in demand. If the data center fails at an off peak hour, the nearest data center to it may be able to serve all of its demand. However, if the failure happens at peak demand time, the nearest data center may itself be at its peak demand, and many clients from the region may have to be routed to multiple different data centers. At the very least, geo-replication solutions need to take input from the capacity provisioning system regarding where clients will be re-directed after a failure. Since we do not know when the failure will happen, statically determining replication sites will likely require significantly conservative copying to too many possible backup sites. On the other hand, backing up to just one site may mean that large amounts of data has to be migrated after a failure. A good solution will carefully trade-off the network costs of data replication, increased consistency protocol overheads due to increased number of copies, support for concurrent access at multiple locations for shared data, the higher storage space requirement, and post-failure migration penalty. It may even turn out that for data intensive applications with high network usage, minimizing network usage at the cost of increased excess capacity nearby is more cost effective. Since many applications have user specific data, placement of replicas will also influence which users are to be re-directed to which data center and this can require coordination with the request routing mechanism. Determining the best APIs to share the required information is an interesting research problem.

Auto-scaling: Cloud applications can auto-scale the compute capacity they are using in response to client demand. As long as demand follows expected patterns, the provisioned capacity at each data center would allow for this scaling. However, if an application sees an unexpected demand spike, and the local data center is unable to satisfy the scaling needs, auto-scaling can re-direct this demand to other data centers and scale its number of instances across multiple data centers. While existing auto-scaling algorithms only tune up (or down) the number of compute instances, leaving the storage layer scaling to the cloud provider, a global auto-scaling algorithm will have to consider additional factors aside from compute utilization, such as which data centers have the required data replicas and if additional data needs to be migrated to enable a scaling event.

Virtualized Availability: The cloud infrastructure as constructed may be capable of meeting certain latency and availability targets if an application is geo-distributed to all the cloud locations. However, not all applications may need that same latency and availability. Based on actual needs, an application may determine which and how many of the cloud data centers it spreads itself across. Using fewer data centers may not yield the lowest latency but can help reduce data replication complexity and networking costs. Reduced spare capacity, or longer post-failure latency constraints can be used if the application can tolerate lower performance after a failure. A good design can exploit the flexibility offered by geo-distribution to virtualize the cloud in the availability dimension, allowing applications to pay for what they need.

4 Related Work

Generic capacity and placement planning for geo-distributed facilities has been studied in depth in the context of the facility location problem [13, 27]. Specifically for data centers, both placement and capacity optimizations have been considered in [19, 24, 25, 32]. For instance, the work in [19] used sophisticated cost models for a geo-distributed cloud infrastructure to show that a larger number of locations provides lower latency, and that for a given availability target, a larger number of cheaper (reduced redundancy) data centers are more cost effective. However, prior works have not considered the excess capacity created due to temporal variations in demand across sites and how this capacity can be leveraged to achieve higher availability for no additional cost. Prior works [37] have turned servers off temporarily by exploiting multiple data center locations within a latency bound from a client but these works do not minimize the actual capacity built at each of those data centers, and do not determine the excess capacity required for availability reasons. Also, we model availability in terms of supporting a data center failure, rather than assuming hard numbers quantifying the probability of failure. Such numbers are typically only available for the power infrastructure, designated as tiers or number of 9s, and do not account for other causes of data center failures observed in practice such as electrical storms, software issues and operator errors [39, 3, 20].

References

[1] AGARWAL, S., DUNAGAN, J., JAIN, N., SAROIU, S., WOLMAN, A., AND BHOGAN, H. Volley: automated data placement for geo-distributed cloud services. In *Proceedings of the 7th USENIX conference on Networked systems design and implementation* (Berkeley, CA, USA, 2010), NSDI'10, USENIX Association, pp. 2–2.

[2] AGUILERA, M. K., MERCHANT, A., SHAH, M., VEITCH, A., AND KARAMANOLIS, C. Sinfonia: A new paradigm for building scalable distributed systems. *ACM Trans. Comput. Syst. 27*, 3 (Nov. 2009), 5:1–5:48.

[3] AMAZON-AWS. Summary of the october 22,2012 aws service event in the us-east region. `https://aws.amazon.com/message/680342/`, Oct. 2012.

[4] AWS, A. Amazon ec2 regions and availability zones. `http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-regions-availability-zones.html`.

[5] AWS, A. Amazon ec2 spot instances. `https://aws.amazon.com/ec2/purchasing-options/spot-instances/`.

[6] AWS, A. Global infrastructure north america. http://aws.amazon.com/about-aws/globalinfrastructure/.

[7] BALLANI, H., AND FRANCIS, P. Towards a global ip anycast service. In *Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications* (New York, NY, USA, 2005), SIGCOMM '05, ACM, pp. 301–312.

[8] BIRMAN, K. P. The process group approach to reliable distributed computing. *Commun. ACM 36*, 12 (Dec. 1993), 37–53.

[9] BRUTLAG, J. Speed matters for google web search. `http://services.google.com/fh/files/blogs/google_delayexp.pdf`, July 2009.

[10] BUREAU, U. C. Census data access tools.

[11] BURROWS, M. The chubby lock service for loosely-coupled distributed systems. In *Proceedings of the 7th Symposium on Operating Systems Design and Implementation* (Berkeley, CA, USA, 2006), OSDI '06, USENIX Association, pp. 335–350.

[12] CALDER, B., WANG, J., OGUS, A., NILAKANTAN, N., SKJOLSVOLD, A., MCKELVIE, S., XU, Y., SRIVASTAV, S., WU, J., SIMITCI, H., HARIDAS, J., UDDARAJU, C., KHATRI, H., EDWARDS, A., BEDEKAR, V., MAINALI, S., ABBASI, R., AGARWAL, A., HAQ, M. F. U., HAQ, M. I. U., BHARDWAJ, D., DAYANAND, S., ADUSUMILLI, A., MCNETT, M., SANKARAN, S., MANIVANNAN, K., AND RIGAS, L. Windows azure storage: a highly available cloud storage service with strong consistency. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles* (New York, NY, USA, 2011), SOSP '11, ACM, pp. 143–157.

[13] CHURCH, R., AND REVELLE, C. The maximal covering location problem. *Papers of the Regional Science Association 32*, 1 (1974), 101–118.

[14] COOPER, B. F., RAMAKRISHNAN, R., SRIVASTAVA, U., SILBERSTEIN, A., BOHANNON, P., JACOBSEN, H.-A., PUZ, N., WEAVER, D., AND YERNENI, R. Pnuts: Yahoo!'s hosted data serving platform. *Proc. VLDB Endow. 1*, 2 (Aug. 2008), 1277–1288.

[15] CORBETT, J. C., DEAN, J., EPSTEIN, M., FIKES, A., FROST, C., FURMAN, J. J., GHEMAWAT, S., GUBAREV, A., HEISER, C., HOCHSCHILD, P., HSIEH, W., KANTHAK, S., KOGAN, E., LI, H., LLOYD, A., MELNIK, S., MWAURA, D., NAGLE, D., QUINLAN, S., RAO, R., ROLIG, L., SAITO, Y., SZYMANIAK, M., TAYLOR, C., WANG, R., AND WOODFORD, D. Spanner: Google's globally-distributed database. In *Proceedings of the 10th USENIX conference on Operating Systems Design and Implementation* (Berkeley, CA, USA, 2012), OSDI'12, USENIX Association, pp. 251–264.

[16] DECANDIA, G., HASTORUN, D., JAMPANI, M., KAKULAPATI, G., LAKSHMAN, A., PILCHIN, A., SIVASUBRAMANIAN, S., VOSSHALL, P., AND VOGELS, W. Dynamo: amazon's highly available key-value store. In *Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles* (New York, NY, USA, 2007), SOSP '07, ACM, pp. 205–220.

[17] DENG, N., STEWART, C., GMACH, D., ARLITT, M., AND KELLEY, J. Adaptive green hosting. In *Proceedings of the 9th International Conference on Autonomic Computing* (New York, NY, USA, 2012), ICAC '12, ACM, pp. 135–144.

[18] GAO, V., ZENG, Z., LIU, X., AND KUMAR, P. The answer is blowing in the wind: Analysis of powering internet data centers with wind energy. In *INFOCOM, 2013 Proceedings IEEE* (April 2013), pp. 520–524.

[19] GOIRI, I., LE, K., GUITART, J., TORRES, J., AND BIANCHINI, R. Intelligent placement of datacenters for internet services. In *Distributed Computing Systems (ICDCS), 2011 31st International Conference on* (June 2011), pp. 131–142.

[20] GOOGLE APPENGINE FORUM. Post-mortem for february 24th, 2010 outage. `https://groups.google.com/forum/#!topic/google-appengine/p2QKJ0OSLc8`, Mar. 2010.

[21] GRAY, W., AND BOEHM-DAVIS, D. Milliseconds matter: An introduction to microstrategies and to their use in describing and predicting interactive behavior. *Journal of Experimental Psychology: Applied 6*, 4 (2000), 322.

[22] GREENBERG, A., HAMILTON, J., MALTZ, D. A., AND PATEL, P. The cost of a cloud: research problems in data center networks. *SIGCOMM Comput. Commun. Rev. 39*, 1 (Dec. 2008), 68–73.

[23] HAMILTON, J. Inter-datacenter replication and georedundancy. `http://perspectives.mvdirona.com/2010/05/10/InterDatacenterReplicationGeoRedundancy.aspx`, 2010.

[24] HANS, R., LAMPE, U., AND STEINMETZ, R. Qos-aware, cost-efficient selection of cloud data centers. In *Proceedings of the 2013 IEEE Sixth International Conference on Cloud Computing (CLOUD)* (2013).

[25] HASAN, S., GORINSKY, S., DOVROLIS, C., AND SITARAMAN, R. K. Trade-offs in optimizing the cache deployments of cdns. In *Proceedings of the International Conference on INFOCOM* (2014).

[26] INC, G. Data center locations. `http://www.google.com/about/datacenters/inside/locations/index.html`.

[27] JAIN, K., MAHDIAN, M., MARKAKIS, E., SABERI, A., AND VAZIRANI, V. V. Greedy facility location algorithms analyzed using dual fitting with factor-revealing lp. *J. ACM 50*, 6 (Nov. 2003), 795–824.

[28] JUNQUEIRA, F. P., AND REED, B. C. The life and times of a zookeeper. In *Proceedings of the 28th ACM Symposium on Principles of Distributed Computing* (New York, NY, USA, 2009), PODC '09, ACM, pp. 4–4.

[29] KANSAL, A., URGAONKAR, B., AND GOVINDAN, S. Using dark fiber to displace diesel generators. In *Proceedings of the 14th USENIX Conference on Hot Topics in Operating Systems* (Berkeley, CA, USA, 2013), HotOS'13, USENIX Association, pp. 15–15.

[30] KOHAVI, R., HENNE, R. M., AND SOMMERFIELD, D. Practical guide to controlled experiments on the web: listen to your customers not to the hippo. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining* (New York, NY, USA, 2007), KDD '07, ACM, pp. 959–967.

[31] LAKSHMAN, A., AND MALIK, P. Cassandra: A decentralized structured storage system. *SIGOPS Oper. Syst. Rev. 44*, 2 (Apr. 2010), 35–40.

[32] LARUMBE, F., AND SANSÒ, B. Optimal location of data centers and software components in cloud computing network design. In *Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (Ccgrid 2012)* (2012), CCGRID '12.

[33] LE, K., BIANCHINI, R., MARTONOSI, M., AND NGUYEN, T. D. Cost- and energy-aware load distribution across data centers. In *Proceedings of the Workshop on Power-Aware Computing and Systems (HotPower)* (October 2009).

[34] LE, K., BILGIR, O., BIANCHINI, R., MARTONOSI, M., AND NGUYEN, T. D. Capping the brown energy consumption of internet services at low cost. In *Proceedings of the International Green Computing Conference (IGCC)* (August 2010).

[35] LIU, Z., LIN, M., WIERMAN, A., LOW, S. H., AND ANDREW, L. L. Greening geographical load balancing. In *Proceedings of the ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems* (New York, NY, USA, 2011), SIGMETRICS '11, ACM, pp. 233–244.

[36] LLOYD, W., FREEDMAN, M. J., KAMINSKY, M., AND ANDERSEN, D. G. Stronger semantics for low-latency geo-replicated storage. In *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation* (Berkeley, CA, USA, 2013), nsdi'13, USENIX Association, pp. 313–328.

[37] MATHEW, V., SITARAMAN, R., AND SHENOY, P. Energy-aware load balancing in content delivery networks. In *INFOCOM, 2012 Proceedings IEEE* (March 2012), pp. 954–962.

[38] MICROSOFT. Windows azure traffic manager overview. `http://msdn.microsoft.com/en-us/library/windowsazure/hh744833.aspx`.

[39] MSDN BLOGS. Details of the december 28th, 2012 windows azure storage disruption in us south. `http://blogs.msdn.com/b/windowsazure/archive/2013/01/16/details-of-the-december-28th-2012-windows-azure-storage-disruption-in-us-south.aspx`, Jan. 2013.

[40] PRINCE, M. A brief primer on anycast. `http://blog.cloudflare.com/a-brief-anycast-primer`.

[41] RAJAGOPALAN, S., CULLY, B., O'CONNOR, R., AND WARFIELD, A. Secondsite: disaster tolerance as a service. In *Proceedings of the 8th ACM SIGPLAN/SIGOPS conference on Virtual Execution Environments* (New York, NY, USA, 2012), VEE '12, ACM, pp. 97–108.

[42] SINGH, R., IRWIN, D., SHENOY, P., AND RAMAKRISHNAN, K. K. Yank: Enabling green data centers to pull the plug. In *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation* (Berkeley, CA, USA, 2013), nsdi'13, USENIX Association, pp. 143–156.

[43] SIVASUBRAMANIAN, S., ALONSO, G., PIERRE, G., AND VAN STEEN, M. Globedb: Autonomic data replication for web applications. In *Proceedings of the 14th International Conference on World Wide Web (WWW)* (2005).

[44] SOLUTIONS, N., AND NETWORKS. Liquid applications. `http://nsn.com/portfolio/liquid-net/intelligent-broadband-management/liquid-applications`.

[45] STEWART, C., AND SHEN, K. Some joules are more precious than others: Managing renewable energy in the datacenter. In *HotPower* (2009).

[46] TERRY, D. B., PRABHAKARAN, V., KOTLA, R., BALAKRISHNAN, M., AGUILERA, M. K., AND ABU-LIBDEH, H. Consistency-based service level agreements for cloud storage. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles* (New York, NY, USA, 2013), SOSP '13, ACM, pp. 309–324.

[47] WU, Z., BUTKIEWICZ, M., PERKINS, D., KATZ-BASSETT, E., AND MADHYASTHA, H. V. Spanstore: Cost-effective geo-replicated storage spanning multiple cloud services. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles* (New York, NY, USA, 2013), SOSP '13, ACM, pp. 292–308.

[48] XU, H., AND LI, B. Joint request mapping and response routing for geo-distributed cloud services. In *INFOCOM, 2013 Proceedings IEEE* (April 2013), pp. 854–862.

[49] YE, Z., LI, S., AND ZHOU, X. Gcplace: Geo-cloud based correlation aware data replica placement. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing* (New York, NY, USA, 2013), SAC '13, ACM, pp. 371–376.