Freestyle Dancing: Randomized Algorithms for Dynamic Storage Load-Balancing

Liang Liu†, Yating Wang†, Lance Fortnow†, Jin Li‡, Jun Xu†
†Georgia Institute of Technology ‡Microsoft Research Technologies
{lliu315, fortnow}@gatech.edu, Jinl@microsoft.com

ABSTRACT

In this work, we study a challenging research problem that arises in minimizing the cost of storing customer data online for reliable accesses in a cloud. It is how to near-perfectly balance the remaining capacities of all disks across the cloud system while adding new file blocks so that the inevitable event of capacity expansion can be postponed as much as possible. The challenges of solving this problem are twofold. First, new file blocks are added to the cloud concurrently by many dispatchers (computing servers) that have no communication or coordination among themselves. Though each dispatcher is updated with information on disk occupancies, the update is infrequent and not synchronized. Second, for fault-tolerance purposes, a combinatorial constraint has to be satisfied in distributing the blocks of each new file across the cloud system. We propose a randomized algorithm, in which each dispatcher independently samples a blocks-to-disks assignment according to a probability distribution on a set of assignments conforming to the aforementioned combinatorial requirement. We show that this algorithm allows a cloud system to near-perfectly balance the remaining disk capacities as rapidly as theoretically possible, when starting from any unbalanced state that is correctable mathematically.

Categories and Subject Descriptors

C.1.2 [Multiple Data Stream Architectures]
E.2 [Data Storage Representations]

Keywords

Load Balance, Capacity Distribution, Load Distribution, Birkhoff Decomposition

1. INTRODUCTION

A cloud service provider needs to provide its customers with reliable accesses to their own data, despite service disruptions of its disks, computers, and networks, caused possibly by power failures, hardware malfunctions, scheduled maintenances/upgrades, operator errors, etc. This disruption tolerance capability is achieved typically through a combination of two risk control measures. One is erasure coding [4], that is, to divide a customer file into b blocks, and then encode them into $b+b' = k$ blocks (by adding b' blocks of redundancies) so that missing one or two such blocks due to aforementioned disruptions will not prevent the file from being decoded and delivered to the customer. The other is "geographic" diversity, that is, to strategically distribute these (coded) blocks across the cloud system so that a typical service disruption event may affect the access to at most one or two such (coded) blocks.

In this work, we study a research problem concerning how to near-perfectly balance the remaining capacities of all disks across the cloud system while adding new file blocks so that the inevitable event of capacity expansion can be postponed as much as possible. This problem is challenging due to two constraints the proposed load-balancing solution has to work under. The first constraint is the aforementioned "geographic" diversity requirement. At the cloud service provider we are working with, the disk servers of the entire cloud system is logically divided into mn equal-capacity partitions – called cells – organized as an $m \times n$ matrix. This partitioning is done in such a way that, with high probability, at most one row and/or one column (of storage servers) of will be impacted at any time by an aforementioned service disruption event [6]. Hence the diversity requirement translates into that, when a new customer file, consisting of k coded blocks, is added to the cloud system, no cell can take in more than one such block, and no two cells that take in a block can be on the same row or on the same column of the matrix.

The second constraint is "distributed independent dispatching". In a cloud environment, new files are added to the system by a large number of front-end computing servers that interact with customers, which we call dispatchers. "Distributed independent dispatching" means that no communication or coordination is allowed among dispatchers when they add new files to the disks, because such communication or coordination is prohibitively expensive in a large cloud environment with a large number of dispatchers geographically distributed across the world. In addition, to minimize reporting overheads, dispatchers obtain accurate reporting of the loads (occupancy ratios) of all disks (cells) in a cloud system quite infrequently (say once a day). It can be shown this second constraint rules out naive solutions such as a (deterministic) greedy approach, in which we try to place k blocks in a diverse set of k cells that collectively have the smallest overall occupancy ratios.

2. THE LOAD-BALANCING PROBLEM AND OUR SOLUTION

In this work, we propose a randomized load-balancing algorithm that works as follows. Each dispatcher, upon receiving a disk occupancy report, precomputes a probability distribution over a set of possible blocks-to-cells assignments conforming to the afore-

SIGMETRICS '16 June 14-18, 2016, Antibes Juan-Les-Pins, France

ACM ISBN 978-1-4503-4266-7/16/06.
DOI: http://dx.doi.org/10.1145/2896377.2901481

mentioned diversity requirement. Note that every such assignment corresponds to a k-matching when the rows and the columns of the matrix are viewed as the two independent vertex sets in a bipartite graph. Later on, when a dispatcher has to add (the k blocks of) a new file to the cloud system, it randomly samples a blocks-to-cells assignment according to the computed distribution, and dispatches the blocks accordingly. Note it is important to do this in two steps, since the first step – computing the desired distribution – which is done offline, could take a long time (typically minutes). The second step – making random assignments based on the precomputed probability distribution – on the other hand, must be finished in milliseconds in order not to "freeze" the cloud service.

Our main research problem is therefore to identify (mathematically) and compute a suitable probability distribution, that, after a large number of random blocks-to-cells assignments are made – independently by many dispatchers – according to the distribution, the residual spaces of all disks in the cloud system will be made very close to one another, even when they are far apart to start with. A major contribution of this work is to identify and efficiently compute such a distribution. Armed with this distribution, our randomized load-balancing algorithm can balance the disk loads across different cells almost as perfectly as the idealized scheme (centralized control with perfect and timely knowledge of the loads of all cells), under the diversity constraint.

Computing this probability distribution above has indeed been studied in the mathematics literature [2, 1, 5]. Here we describe this computation in this context of cloud storage load-balancing. Let $L = \{l_{ij}\}$ be the load matrix of the cloud system in the sense l_{ij} is the total storage load of the cell on the i_{th} row and j_{th} column of the cloud matrix. The computation consists of two steps. The first step is to derive the minimum target load level $\hat{l}$ that can be reached by all cells simultaneously by the aforementioned idealized scheme [5]. Let L_T be an $m \times n$ matrix every element of which has value $\hat{l}$. The second step is to decompose the difference matrix $L_T - L$ into a linear combination of k-matchings [2, 1]. By normalizing the coefficients in the linear combination, we arrive at the probability distribution. This decomposition problem can be converted to the Birkhoff-Von Neumann (BVN) decomposition of an $(m+n-k) \times (m+n-k)$ doubly stochastic matrix. By using the most efficient algorithm discovered so far for BVN decomposition [3], the complexity of computing this probability distribution is $O\big((m+n-k)^3 \log(m+n-k)\big)$. One of our contributions is to propose a deterministic algorithm that can reduce this complexity to $O\big((m+n)\log(m+n) + (3k)^3 \log(3k)\big)$. For small k values that are typical for storage erasure codes, the latter complexity can be several orders of magnitude smaller than the former, for very large cloud size ($m \times n$). We omit this algorithm here due to lack of space.

After a certain number of assignments are made according to the this probability distribution, the loads across cells should become quite balanced. At that point, the load-balancing operation in the cloud system enters a cruising stage. In the cruising stage, a dispatcher adds statistically the same number of blocks to each cell so as not to change the status quo. We achieve it by using a "plane sweep" algorithm at every dispatcher. This algorithm limits the maximum such variation caused by a single dispatcher to at most two blocks. In addition, the algorithm instances at different dispatchers are independently randomly parameterized to keep the actions of different dispatchers, and hence the cells where such maximum variations occur, statistically independent of one another. The basic idea of the algorithm is for a dispatcher to make successive assignments using the same $k \times k$ k-matching matrix π that is shifted in a "sweeping pattern" over time.

3. EVALUATION

We perform a simulation study to evaluate two properties of our load-balance scheme. One is how balanced our scheme can achieve and maintain in the cloud. The other one is how fast our scheme can restore the balanced state after some typical breakdown scenarios. The parameters we use in the simulation are as follows: $m = 60$, $n = 20$, $k = 18$. In addition, each cell can hold 1.5×10^7 blocks and the total number of new blocks to be added to the cloud system per day is about 0.1% of the cloud storage capacity. In the simulation, we use metric $D = \max l_{ij} - \bar{l}$, the difference between the load ratio of the most heavily loaded cell and the average load ratio to measure how well loads are balanced across the cells, and call this D the *positive load ratio deviation*. D reflects the evenness of load distribution.

According to our simulation results (1200 simulation runs, each of which simulates 800 days that gradually fills up the cloud storage from 0% to 80% loaded), with our load balancing scheme, D could be maintained at smaller than 0.03% of each cell's maximum capacity with overwhelming probability. That means with overwhelming probability, the utilization ratio of the cloud could reach 79.97% without any cell's load exceeding 80%.

We also evaluate how fast and effectively our load-balancing scheme can "fix" load-imbalance caused by a hardware failure scenario. We assume that the hardware failure makes one row of cells unaccessible for as long as a week (7 days). We made 100 simulation runs for this scenario. Our results show that during the hardware failure period, the deviation D increases to about 0.003% of a cell's capacity. After the row impacted by the hardware failure comes back into service, our weighted randomized assignment algorithm restores the balance within four days, or after new blocks, in the amount no more than $0.1\% \times 4 = 0.4\%$ of the total storage of the cloud system, are assigned.

Acknowledgment

This work is supported in part by US NSF through grants CNS-1248117 and CNS-1423182.

4. REFERENCES

[1] BIRKHOFF, D. Tres observaciones sobre el algebra lineal. *Universidad Nacional de Tucuman Revista , Serie A 5* (1946), 147–151.

[2] CHANG, C.-S., CHEN, W.-J., AND HUANG, H.-Y. Birkhoff-von neumann input buffered crossbar switches. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE* (Mar 2000), vol. 3, pp. 1614–1623 vol.3.

[3] GOEL, A., KAPRALOV, M., AND KHANNA, S. Perfect matchings in $O(n \log n)$ time in regular bipartite graphs. *CoRR abs/0909.3346* (2009).

[4] HUANG, C., SIMITCI, H., XU, Y., OGUS, A., CALDER, B., GOPALAN, P., LI, J., AND YEKHANIN, S. Erasure coding in windows azure storage. In *Presented as part of the 2012 USENIX Annual Technical Conference (USENIX ATC 12)* (Boston, MA, 2012), USENIX, pp. 15–26.

[5] MENDELSOHN, N. S., AND DULMAGE, A. L. The convex hull of sub-permutation matrices. *Proceedings of the American Mathematical Society 9* (1958), 253–254.

[6] MICROSOFT AZURE STORAGE TEAM. Introducing zone redundant storage. bit.ly/1LFsA44, 2014.