A Quantitative Analysis of Quorum System Availability in Data Centers

Xu Wang, Hailong Sun, Ting Deng and Jinpeng Huai
School of Computer Science and Engineering
Beihang University
Beijing, China
Email: {wangxu, sunhl, dengting, huaijp}@act.buaa.edu.cn

***Abstract*—Large-scale distributed storage systems often replicate data across servers and even geographically-distributed data centers for high availability, while existing theories like CAP and PACELC show that there is a tradeoff between availability and consistency. However, current practice is mainly experience-based and lacks quantitative analysis for identifying a good tradeoff between the two. In this work, we are concerned with providing a quantitative analysis on availability for widely-used quorum systems in data centers. First, a probabilistic model is presented to quantify availability for typical data center networks: 2-tier basic tree, 3-tier basic tree, fat tree and folded clos network. Second, we build the availability-consistency table and propose a set of rules to quantitatively make tradeoff between availability and consistency. Finally, with Monte Carlo based simulations, we validate our presented quantitative results and show that our approach to make tradeoff between availability and consistency is effective.**

I. Introduction

Large-scale distributed storage systems often face the challenges of high availability and scalability [1]. They typically replicate data across multiple severs and even geographically-distributed data centers for tolerating network partitions. However, the CAP [2] and PACELC [3] theorems show that there is a tradeoff between availability and consistency in data replication. In order to provide extremely high availability, systems such as Dynamo [4] and Cassandra [5] often eschew strong consistent replicas and instead provide eventual consistency [6]. But the eventual consistency may lead to inconsistency such as read staleness [7] and write conflicts [4]. Therefore it is necessary to know how much availability can be obtained in different consistency levels such that we can quantitatively make tradeoff between them.

Distributed quorums are widely used for eventually consistent storage systems in data centers [4] [5]. With quorum replication, these storage systems write a data item by sending it to a set of replicas (write quorums) and read from a possibly different set of replicas (read quorums). Typically any W replicas act as write quorums and any R replicas as read ones. With N replicas, if $W+R > N$, strong consistency is achieved since the overlapping of write and read quorums is always guaranteed; and if $W + R \leq N$, the overlapping condition may not be satisfied, therefore it can cause inconsistency. The former is a strict quorum system [8] and the latter is a probabilistic quorum system [9]. It is obvious that greater W and R will produce stronger consistency, but will lower system availability since greater W and R require more live replicas to be accessed for available write and read requests during failures. How to configure (W, R) to get the best solution for both consistency and availability? The current practice is mainly experience-based and lack quantitative analysis for identifying a good tradeoff between them.

978-1-4799-4852-9/14/$31.00

The quantitative consistency analysis for quorum systems is well studied [7] [9]. However, availability is network topology aware because network partition is the main cause of system unavailability according to the CAP theorem [2], and data center networks are complex [10]–[12] such as basic tree, fat tree, folded clos network, therefore the quantitative analysis on availability of quorum systems in data centers is challenging. Several works [9], [13]–[17] on quantifying the availability of quorum systems have been done, but there are two limitations: (1) they usually assume simple network topologies such as the fully connected network and ring topology; (2) they define the availability of quorum systems as the probability that at least one quorum is alive, but in practice live quorums may be inaccessible due to failures of switches in networks.

In this work, we focus on evaluating the availability of quorum systems in four typical data center networks: 2-tier basic tree, 3-tier basic tree, fat tree and folded clos network. At first, we propose a system model $QS(DCN, PM, W/R)$ for quorum systems, where DCN represents data center network topologies, PM is a vector describing the placement of replicas on servers, and W/R are the sizes of write/read quorums. Since W and R are equivalent to our analysis, we only consider write requests. Based on $QS(DCN, PM, W)$, the write availability is defined as the probability that a write can reach at least one live replica (i.e. the coordinator) and can arrive at least other $W - 1$ live replicas from the coordinator. Although typical data center networks are complex especially for the fat tree and folded clos network, they are essentially tree-like. Therefore, after calculating the write availability for simple 2-tier basic tree, we use super nodes to represent sub-trees or groups of nodes for 3-tier basic tree, fat tree and folded clos network which are logically equivalent to each other and obtain simplified network topologies. Then the write availability for 3-tier basic tree, fat tree and folded clos network is computed by conditional probability and reusing the result of 2-tier basic tree. Based on the quantitative results of availability, we build a (W, R) availability-consistency table filled with values of $\langle Avaialbility, Consistency \rangle$ for quorum systems, and propose a set of rules to choose the best (W, R) to make tradeoff between availability and consistency.

Specifically, we make the contributions as follows: (1) On the basis of system model $QS(DCN, PM, W)$, we build a probabilistic model for the write availability of quorum systems in four typical data center networks: 2-tier basic tree, 3-tier basic tree, fat tree and folded clos network; (2) We build a (W, R) availability-consistency table and propose a set of rules to choose the best (W, R) configuration for a specific quorum system to balance the availability and consistency; (3) And with Monte Carlo based event driven simulations, we validate our quantitative results and show the effectiveness of our proposed approach to quantitatively make tradeoff between availability and consistency.

The remainder of this paper is structured as follows. Section II presents our system model. Section III describes how to quantify the write availability of quorum systems. How to make tradeoff between availability and consistency is shown in Section IV. Section V provides our experimental results. Related work is presented in Section VI. Finally, Section VII concludes.

II. System Model

Our quorum system model for data center networks is a triple $\langle DCN, PM, W/R\rangle$, where

- $DCN \in$ {2-tier basic tree, 3-tier basic tree, K-ary fat tree, folded clos}; that is, DCN is one of the four typical data center networks described in [10]–[12], as shown in Figure 2,3,4,6. They usually have three-tier switches (or routers): core tier in the root of the tree, aggregation tier in the middle and edge tier at the leaves of the tree. The core switches (CS), the aggregation switches (AS) and the ToR (Top of Rack) switches are in different tier;
- PM is a 0/1 vector representing the placement of replicas on servers; and
- W/R describe the sizes of write quorums and read quorums.

In the four data center networks, we assume that there are three kinds of switches: core switches (10-GigE switches) with the crash probability P_c, aggregation switches (10-GigE switches) with the crash probability P_a and ToR switches (1-GigE switches) with the crash probability P_t. We also assume that the links never crash, servers are homogenous with the crash probability P_s, and the crash events are independent.

Without loss of generality, we assume there is an order for all devices with the same type, and we use 1 and 0 to represent a replica on and not on a server, respectively. So PM is a 0/1 vector whose length equals the number of servers (i.e., n_{server}), and the number of 1 is the number of replicas (N). That is $PM = \langle r_1^s, r_2^s, ..., r_{n_{server}}^s\rangle$, where $r_i^s = 0$ or $1 (1 \le i \le n_{server})$ and $\sum_{i=1}^{n_{server}} r_i^s = N$. In addition, we define $PM^{tor} = \langle r_1^t, r_2^t, ..., r_{n_{tor}}^s\rangle$ where n_{tor} denotes the number of ToR switches and element r_i^t means the number of replicas that the ith ToRS can access from downlinks, thus $\sum_{i=1}^{n_{tor}} r_i^{tor} = N$. Similarly, PM^{as} for aggregation switches can be defined.

III. Quantitative Analysis of Availability

Given the replica placement vector PM, the write availability of a quorum system in one data center network is the probability that a write request reaches at least one live replica through core switches and the replica can arrive at at least other $(W-1)$ live replicas. In this section, we will compute the write availability for four typical data center networks: 2-tier basic tree (bt2), 3-tier basic tree (bt3), k-ary fat tree (ft) and folded clos network (fc), respectively.

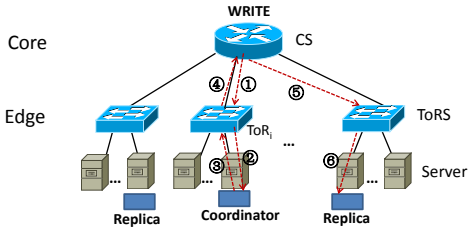

Fig. 1. Available Write for 2-tier Basic Tree

A. 2-tier Basic Tree

For a two-tier basic tree, since only the one core switch can receive write requests, we just consider the involved sub-network which contains all possible paths from the core switch to N replicas. As shown in Figure 1, the sub-network contains N ToRS's, and each ToRS connects to N servers, thus any placement of N replicas is possible. And a write for the two-tier basic tree is available if and only if it can access at least W live replicas.

Let stochastic variable A_i be the number of accessible live replicas underlying the ith ToRS from the live CS, where $1 \le i \le N$. Given $PM = \langle r_1^s, r_2^s, ..., r_{n_{server}}^s\rangle (n_{server} = N^2)$, we can get $PM^{tor} = \langle r_1^t, r_2^t, ..., r_N^t\rangle$ where $r_i^t = \sum_{j=(i-1)N+1}^{iN} r_j^s$. Then we calculate the probability density function (pdf) of A_i as follows:

$$Pr(A_i = m_i) = \begin{cases} P_t + (1-P_t)(P_s)^{r_i^t} & r_i^t \ge m_i = 0; \\ (1-P_t)\binom{r_i^t}{m_i}(1-P_s)^{m_i}(P_s)^{r_i^t - m_i} & r_i^t \ge m_i > 0. \end{cases}$$

where $m_i = 0$ if the ith ToRS fails, or the ith ToRS is alive and all r_i^t servers with replicas fail; and $m_i > 0$ if the ith ToRS is alive and exact m_i servers with replicas are alive.

Let stochastic variable $X_{bt2-tor}$ be the number of accessible live replicas from the live CS in the two-tier basic tree network, and the probability of $X_{bt2-tor} = x$ is the sum of the probability for all possible cases that $A_1 + A_2 + ... + A_N = x$. Since the CS is alive, $A_1, ..., A_N$ are independent, we have:

$$Pr(X_{bt2-tor} = x) = \sum_{m_1+m_2+...+m_N=x} \prod_{i=1}^{N} Pr(A_i = m_i)$$

where $0 \le m_i \le r_i^t$. Define a function $Q_{bt2-tor}(x) = Pr(X_{bt2-tor} = x)$ for simplification, then:

$$Avail(QS(bt2, W)) = (1 - P_c) \sum_{x=W}^{N} Q_{bt2-tor}(x) \quad (1)$$

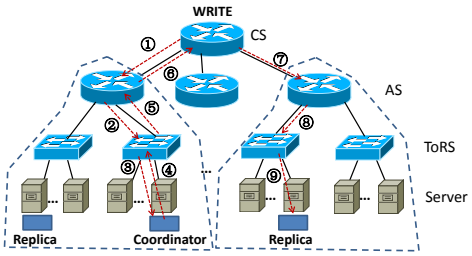

Fig. 2. Available Write for 3-tier Basic Tree

B. 3-tier Basic Tree

In the three-tier basic tree, we also only consider the possible involved sub-network. Figure 2 shows the sub-network which has N AS's, and each AS connects to N ToRS's and each ToRS connects to N servers. Obviously a write for the three-tier basic tree is available if and only if it can access at least W live replicas.

Let stochastic variable B_j be the number of accessible live replicas under the jth AS from the live CS, where $1 \leq j \leq N$. Given $PM = \langle r_1^s, r_2^s, ..., r_{n_{N^3}}^s \rangle$, we can get $PM^{tor} = \langle r_1^t, r_2^t, ..., r_{N^2}^t \rangle$ where $r_i^t = \sum_{j=(i-1)N+1}^{iN} r_j^s$, and $PM^{as} = \langle r_1^a, r_2^a, ..., r_N^a \rangle$ where $r_i^a = \sum_{j=(i-1)N+1}^{iN} r_j^t$. As shown in Figure 2, the sub-trees marked by dash lines can be seen as two-tier basic trees when we replace AS's with CS's, thus we calculate the pdf of B_j by reusing the results of two-tier basic trees:

$$Pr(B_j = m'_j) = \begin{cases} P_a + (1-P_a)Q_{bt2-tor}(0) & r_i^a \geq m'_j = 0; \\ (1-P_a)Q_{bt2-tor}(m'_j) & r_i^a \geq m'_j > 0. \end{cases}$$

where $m'_j = 0$ if the jth AS fails, or the jth AS is alive and the sub-tree whose root is the jth AS can access 0 live replicas; and $m'_j > 0$ if the jth AS is alive and there are exact m'_j replicas which can be accessed in the sub-tree whose root is the jth AS.

Let stochastic variable X_{bt3-as} be the number of accessible live replicas from the live CS in the three-tier basic tree network, and the probability of $X_{bt3-as} = x'$ is the sum of the probability for all possible cases that $B_1 + B_2 + ... + B_N = x'$. Since the CS is alive, $B_1, ..., B_N$ are independent. Thus:

$$Pr(X_{bt3-as} = x') = \sum_{m'_1+m'_2+...+m'_N=x'} \prod_{j=1}^{N} Pr(B_j = m'_j)$$

where $0 \leq m'_j \leq r_j^a$.

Therefore we have:

$$Avail(QS(bt3, W)) = (1-P_c) \sum_{x'=W}^{N} Pr(X_{bt3-as} = x') \quad (2)$$

C. Fat Tree

The K-ary fat tree has K pods and $(K/2)^2$ CS's. There are $K/2$ AS's and $K/2$ ToRS's in a pod where each AS connects to all ToRS's, and each ToRS connects to N servers. In addition, the ith CS connects to the $([i/(K/2)] + 1)$th aggregation switch in all pods. From Figure 3 we can see that, if we put every $(K/2)$ CS's into a group from left to right, we

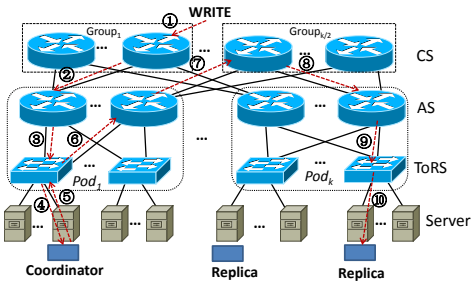

Fig. 3. Available Write for Fat Tree

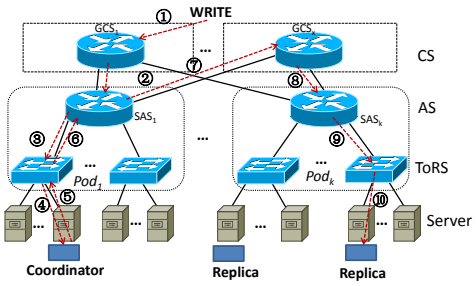

Fig. 4. GCS's and SAS's in Fat Tree

can have $(K/2)$ groups of CS's and the jth group of CS's are all connecting to the jth AS in every pods.

To simplify the K-ary fat tree, we use a core switch group (GCS) to represent a group of CS's since they connect to the same AS's, and use a super aggregation switches (SAS) to represent all AS's in a pod because they connect to the same ToRS's. For a GCS, its crash probability P_{gc} is the probability that all internal CS's crash, thus $P_{gc} = (P_c)^{K/2}$. If there are exact $x (1 \leq x \leq K/2)$ live GCS's, the crash probability P_{sa} for a SAS is the probability that all the internal x AS's which connect to the x live GCS's crash, thus $P_{sa} = (P_a)^x$. Figure 4 shows GCS's and SAS's in the simplified fat tree.

Let stochastic variable C_j be the number of accessible live replicas underlying the jth SAS from the x live GCS's, where $1 \leq j \leq K$. Given $PM = \langle r_1^s, r_2^s, ..., r_{n_{server}}^s \rangle$, we can get $PM^{tor} = \langle r_1^t, r_2^t, ..., r_{K^2/2}^t \rangle$ where $r_i^t = \sum_{j=(i-1)N+1}^{iN} r_j^s$, and $PM^{sas} = \langle r_1^{sa}, r_2^{sa}, ..., r_K^{sa} \rangle$ where $r_i^{sa} = \sum_{j=(i-1)K/2+1}^{iK/2} r_j^t$. As shown in Figure 4, pods can be seen as two-tier basic trees when we replace SAS's with CS's, thus we can compute the pdf of C_j when there are exact x live GCS's:

$$Pr(C_j = m'_j) = \begin{cases} P_{sa} + (1-P_{sa})Q_{bt2-tor}(0) & r_i^{sa} \geq m'_j = 0; \\ (1-P_{sa})Q_{bt2-tor}(m'_j) & r_i^{sa} \geq m'_j > 0. \end{cases}$$

where $m'_j = 0$ if the jth SAS fails, or the jth SAS is alive and the sub-tree whose root is the jth SAS can access 0 live replicas; and $m'_j > 0$ if the jth SAS is alive and there are exact m'_j replicas which can be accessed in the sub-tree whose root is the jth SAS.

Let stochastic variable X_{ft-sas} be the number of accessible live replicas in the fat tree network and the number of live GCS's is denoted by stochastic variable X_{gcs}. When there are exact x live GCS's, the probability of $X_{ft-sas} = x'$

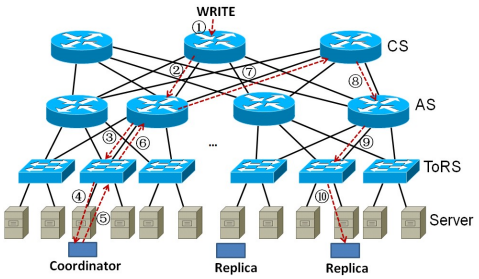

Fig. 5. Available Write for Folded Clos Network

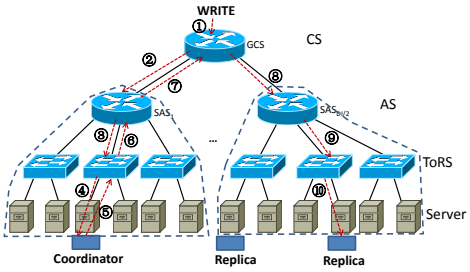

Fig. 6. GCS's and SAS's in Folded Clos

is the sum of the probability for all possible cases that $C_1 + C_2 + ... + C_K = x'$. Since every live GCS connects to all K SAS's, $C_1, ..., C_K$ are independent. Thus:

$$Pr(X_{ft-sas} = x' | X_{gcs} = x) = \sum_{m'_1+m'_2+...+m'_K=x'} \prod_{j=1}^{K} Pr(C_j = m'_j)$$

where $0 \leq m'_j \leq r^{sa}_j$. At the same time, we can have:

$$Pr(X_{gcs} = x) = \binom{K/2}{x} (1 - P_{gc})^x (P_{gc})^{K/2-x}$$

Then we can get $Pr(X_{ft-sas} = x')$ by summing the conditional probability for each possible $x (1 \leq x \leq K/2)$:

$$Pr(X_{ft-sas} = x') = \sum_{x=1}^{K/2} Pr(X_{ft-sas} = x' | X_{gcs} = x) Pr(X_{gcs} = x)$$

Finally the probability that a write can access at least W live replicas for the K-ary fat tree can also be seen as the probability that a write is available. That is:

$$Avail(QS(ft, W)) = \sum_{x'=W}^{N} Pr(X_{ft-sas} = x') \quad (3)$$

Note that, if a write is available, it must be able to access at least W live replicas in the K-ary fat tree. However, when a write can access at least W live replicas in the fat tree where the W live replicas can be accessed only by CS's in different groups, the write coordinator may not cross multiple CS's to reach other replicas, but this possibility is very small since partitioning two CS's in different groups requires cutting so many redundant paths between them in the fat tree. Therefore Equation (3) is a conservative upper bound on write availability of K-ary fat tree but accurate enough for us.

D. Folded Clos Network

The folded clos network includes $D_A/2$ CS's and D_I AS's. Any CS and any AS are connected, and every $D_A/2$ ToRS's connect to two AS's. And each ToRS connects to N servers. As shown in Figure 5, a write for the folded clos network is available if and only if it can access at least W live replicas.

Like the K-ary fat tree, we use a GCS to represent all CS's since they connect to the same AS's, and use a SAS to represent two AS's which connect to the same ToRS's. Figure 6 shows the GCS and SAS's in the simplified folded clos. For the GCS, its crash probability P'_{gc} is the probability that all CS's crash, thus $P'_{gc} = (P_c)^{D_A/2}$; For the SAS's, the crash probability P'_{sa} for a SAS is the probability that both internal AS's crash, thus $P'_{sa} = (P_a)^2$.

Let stochastic variable D_j be the number of accessible live replicas underlying the jth SAS from the GCS, where $1 \leq j \leq D_I/2$. Given $PM = \langle r^s_1, r^s_2, ..., r^s_{n_{serper}} \rangle$, we can get $PM^{tor} = \langle r^t_1, r^t_2, ..., r^t_{D_A D_I/4} \rangle$ where $r^t_i = \sum_{j=(i-1)N+1}^{iN} r^s_j$, and $PM^{sas} = \langle r^{sa}_1, r^{sa}_2, ..., r^{sa}_{D_I/2} \rangle$ where $r^{sa}_i = \sum_{j=(i-1)D_A/2+1}^{iD_A/2} r^t_j$. As shown in Figure 6, the sub-trees marked by dash lines can be seen as two-tier basic trees when we replace SAS's with CS's, thus we calculate the pdf of D_j by reusing the results of two-tier basic trees:

$$Pr(D_j = m'_j) = \begin{cases} P'_{sa} + (1 - P'_{sa}) Q_{bt2-tor}(0) & r^{sa}_i \geq m'_j = 0; \\ (1 - P'_{sa}) Q_{bt2-tor}(m'_j) & r^{sa}_i \geq m'_j > 0. \end{cases}$$

where $m'_j = 0$ if the jth SAS fails, or the jth SAS is alive and the sub-tree whose root is the jth SAS can access 0 live replicas; and $m'_j > 0$ if the jth SAS is alive and there are exact m'_j replicas which can be accessed in the sub-tree whose root is the jth SAS.

Let stochastic variable X_{fc-sas} be the number of accessible live replicas from the live GCS in the folded clos network, and the probability of $X_{fc-sas} = x'$ is the sum of the probability for all possible cases that $D_1 + D_2 + ... + D_{D_I/2} = x'$. Since the GCS is alive, $D_1, ..., D_{D_I/2}$ are independent. Thus:

$$Pr(X_{fc-sas} = x') = \sum_{m'_1+m'_2+...+m'_{D_I/2}=x'} \prod_{j=1}^{D_I/2} Pr(D_j = m'_j)$$

where $0 \leq m'_j \leq r^{sa}_j$. Finally we can have:

$$Avail(QS(fc, W)) = (1 - P'_{gc}) \sum_{x'=W}^{N} Pr(X_{fc-sas} = x') \quad (4)$$

Note that since read requests are performed similarly with write requests in quorum systems, thus the read availability of quorum systems can be represented by $Avail\ (QS(DCN, R))$ where we replace W with R in the equations of write availability.

IV. Availability and Consistency Tradeoff

For a specific quorum system and all possible placements of replicas, we assume that the ratio of write requests is $\alpha(0 \leq \alpha \leq 1)$. Then the system availability with the configuration (W, R) is:

$$Avail(W,R) = \max_{PM}(\alpha Avail(QS(W)) + (1-\alpha) Avail(QS(R)))$$

According to [7], $\binom{N-W}{R} / \binom{N}{R}$ can be viewed as an important measurement of inconsistency for practical quorum systems. In this work we use it to measure the inconsistency, that is:

$$Consistency(W,R) = 1 - \frac{\binom{N-W}{R}}{\binom{N}{R}}$$

where $W + R \leq N$. When $W + R > N$, quorum systems are strong consistent ($Consistency = 1$).

In practical systems, system providers should promise a bounded availability in their service agreement levels, which is usually described by the number of nines:

$$N_n = -\lg(1 - Avail(W,R))$$

However, given a bounded availability by N_n (often one positive integer), it does not mean that greater N_n is better since it will cause weaker consistency of data replicas. In fact, we want to know a configuration (W, R) for a specific quorum system whose availability is no lower than the promised N_n, as well as the possible strongest consistency. Since $W, R(1 \leq W, R \leq N)$ are positive integers and N is not very big, we suggest an approach based on the availability-consistency table to look up the best configuration.

The **availability-consistency table** (ACT) has N rows and N columns, where the row represents the value of W and the column represents the value of R. Element $ACT(W, R)$ in the table is filled with a tuple $\langle [N_n], Consistency \rangle$. Note that N_n is rounded down to match the bounded availability. Since availability decreases and consistency increases when W and R rise, the number of nines will decline and consistency will increase from $ACT(1, 1)$ to $ACT\ (N, N)$. An example of the availability-consistency table is shown in Table II.

When the bounded availability is given as one positive integer N_{n_0} , we look up all feasible elements whose availabilities are no lower than N_{n_0} in the availability-consistency table. But how to choose the best one when there are more than one feasible element? a set of rules are provided as follows: (1) if we want to obtain a stronger consistency at the same time, the one with smaller $[N_n]$ in two feasible elements is better; (2) if two feasible elements have the same $[N_n]$, the one with higher consistency is better; (3) and if two feasible elements have the same $[N_n]$ and the same consistency, the one with smaller value of $\alpha W + (1 - \alpha)R$ is better since $\alpha W + (1 - \alpha)R$ can be viewed as an estimation of system performance. For example, for a read-dominated quorum system which means α is near 0, we prefer that R can be as smaller as possible if we have the same $[N_n]$ and the same consistency.

In fact, above rules show that we prefer to configure (W, R) by an $availability \rightarrow consistency \rightarrow performance$ priority order. But this is just an example, system providers can adapt it based on the availability-consistency table by their personalized preferences.

TABLE I. Write Availability Validation

	RMSE	STD. DEV.
2-tier Basic Tree	0.000472%	0.000417%
3-tier Basic Tree	0.000763%	0.000668%
K-ary Fat Tree	0.00117%	0.000934%
Folded Clos	0.000845%	0.000689%

V. Experiments

In this section, we focus on validating our presented results, and discussing the trends while choosing different experimental parameters. Considering the complexity of equations (1)-(4) and verifiability for experimental results, we implement $QS(DCN, PM, W)$ using Monte Carlo based event driven simulations. The data center network topologies are simulated by extending CloudSim [18].

A. Validating Write Availability

In this experiment, we compare our prediction results calculated by equations (3)-(6) with the observed experimental values to validate them. The parameters we use are tried to be chosen as more reasonable and practical ones. According to the statistics of [1], [19], failure rate P_s is set to 2%,3% and 4%,P_t is 1%,2%,3%,4% and 5%, P_a is one of 1%,5% and 10%, and P_c is 1%,2%. The placement of replicas is represented by PM^{tor} and equivalent placements can be chosen one of them, thus the number of possible placements is less than $N!$ for 2-tier basic tree and $(N!)^2$ for 3-tier basic tree, fat tree and folded clos network, which makes our simulations feasible when N is often small, e.g., Cassandra default configuration($N = 3, W = R = 1$ [20]).

Table I shows all the experimental results of average RMSE (root mean square error) and std. dev. between the observed values and predicted write availability for four typical data center networks when $N \in [3, 9]$, $W \in [1, N]$, $K = 2N$ and $D_A = D_I = 2N$, which validate our predicted results. In addition, the accuracy of our prediction is high enough to at least 4 nines of availability.

B. Impact of W on Write Availability

Equations (1)-(4) reveal the relation of write availability and W. In this experiment, we want to show the change of write availability with respect to W more intuitively. Without loss of generality, we set $P_c = 1\%$, $P_a = 5\%$, $P_t = 2\%$ and $P_s = 2\%$. The number of replicas N is a common value 3, and the placement of replicas is the one which maximizes write availability. For highlighting the variances of different write availability, we use the number of nines to represent it. As shown in Figure 7, we can see that the number of nines decreases as W is increased for any data center networks. In addition, we can obtain different nines of write availability by tuning the value of W for the fat tree and folded clos network but may not for the 2-tier and 3-tier basic tree.

C. An Example of Quantitative Configuration

This experiment is designed to show an example of quantitative (W, R) configuration for making tradeoff between availability and consistency.

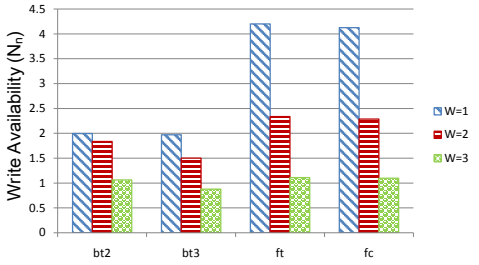

Fig. 7. Impact of W on Write Availability

TABLE II. AVAILABILITY-CONSISTENCY TABLE

$\langle A, C\rangle$	$R=1$	$R=2$	R=3
W=1	$\langle 4, 0.333\rangle$	$\langle 2, 0.667\rangle$	$\langle 1, 1.0\rangle$
W=2	$\langle 3, 0.667\rangle$	$\langle 2, 1.0\rangle$	$\langle 1, 1.0\rangle$
W=3	$\langle 2, 1.0\rangle$	$\langle 2, 1.0\rangle$	$\langle 1, 1.0\rangle$

Take the fat tree as an example. For a quorum system in a K-ary fat tree network ($K = 2N$), the quorum system is read dominated ($\alpha = 0.05$). The number of replicas N is 3. Based on the results of above experiments, the availability-consistency table is built in Table II, where $P_c = 1\%$, $P_a = 5\%$, $P_t = 2\%$ and $P_s = 2\%$. When we need a 3-nines availability, the best configuration of (W, R) is $(2, 1)$ not $(1, 1)$ if we want to obtain a stronger consistency at the same time. Similarly, $(W, R) = (1, 2)$ is not a good configuration while requiring 2-nines availability. And $(W, R) = (3, 1)$ is the best among $(2, 2), (3, 1), (3, 2)$ because it guarantees the strong consistency as well as better performance than other two ones.

VI. RELATED WORK

For strict quorum systems, many works measured their availability and found optimal available quorum systems. [16] found that the most available strict quorum systems are non-dominated coteries. [15] discussed the tradeoffs between the availability, the load and capacity of strict quorum systems. [17] provided explicit optimal available solution for the ring topology. However, our work focuses on practical (W, R) quorum systems which are usually not strict quorum systems.

For probabilistic quorum systems, [9] [13] discussed their availability. [9] showed that probabilistic quorum systems can offer high availability and heavy load at the same time. [13] increased system availability by guaranteeing an upper bound on the staleness of data for probabilistic quorum systems. Both works quantified the availability of probabilistic quorum systems based on fully-connected networks and thus did not consider the impact of network topologies on the availability.

VII. CONCLUSION

In this paper, based on our presented system model $QS(DCN, PM, W/R)$, we quantify the availability of quorum systems for typical data center networks: 2-tier basic tree, 3-tier basic tree, fat tree, folded clos network. With the quantitative results of availability, we build the availability-consistency table and propose a set of rules to choose the best (W, R) configuration for quantitatively making tradeoff between availability and consistency for quorum systems. Through Monte Carlo simulations, we validate our quantitative results and show the effectiveness of our approach to quantitatively balance availability and consistency.

ACKNOWLEDGMENT

This work was supported by National Natural Science Foundation of China (No. 61370057), China 863 program (No. 2012AA011203), China 973 program (No. 2014CB340300), A Foundation for the Author of National Excellent Doctoral Dissertation of PR China (No. 201159), and Beijing Nova Program(2011022).

REFERENCES

[1] J. Dean, "Designs, lessons, and advice from building large distributed systems," in *LADIS Keynote*, 2009.

[2] E. A. Brewer, "Towards robust distributed systems," in *PODC*, 2000, pp. 7–7.

[3] D. J. Abadi., "Consistency tradeoffs in modern distributed database system design: Cap is only part of the story," *IEEE Computer*, vol. 45, no. 2, pp. 37–42, 2012.

[4] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, "Dynamo: Amazons highly available key-value store," in *SOSP*, 2007.

[5] "The apache cassandra project," http://cassandra.apache.org/.

[6] W. Vogels., "Eventually consistent," *Commun. ACM*, vol. 52, pp. 40–44, January 2009.

[7] P. Bailis, S. Venkataraman, M. J. Franklin, J. M. Hellerstein, and I. Stoica, "Probabilistically bounded staleness for practical partial quorums," in *VLDB*, 2012.

[8] G. A. R. Jimenez-Peris, M. Patino Martinez and K. Bettina, "Are quorums an alternative for data replication?" *ACM Trans. Database Syst.*, vol. 28, pp. 257–294, September 2003.

[9] A. W. D. Malkhi, M. Reiter and R. Wright, "Probabilistic quorum systems," *Information and Communication*, vol. 170, pp. 184–206, 2001.

[10] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "Vl2: A scalable and flexible data center network," in *SIGCOMM*, 2009.

[11] A. L. M. Al-Fares and A. Vahdat, "A scalable, commodity data center network architecture," in *SIGCOMM*, 2008.

[12] R. Niranjan Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat, "Portland: A scalable fault-tolerant layer 2 data center network fabric," in *SIGCOMM*, 2009.

[13] A. Aiyer, L. Alvisi, and R. A. Bazzi, "On the availability of non-strict quorum systems," in *DISC*, 2005.

[14] G.-M. H. Barbara D, "The reliability of voting mechanisms," *IEEE Transactions on Computer*, vol. 100, no. 10, pp. 1197–1208, 1987.

[15] M. Naor and A. Wool, "The load, capacity, and availability of quorum systems," *SIAM Journal on Computing*, vol. 27, no. 2, pp. 423–447, 1998.

[16] W. A. Peleg D, "The availability of quorum systems," *Information and Computation*, vol. 123, no. 2, pp. 210–223, 1995.

[17] H. N. T. Ibaraki and T. Kameda, "Optimal coteries for rings and related networks," *Distributed Computing*, vol. 8, pp. 191–201, 1995.

[18] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011.

[19] P. Gill, N. Jain, and N. Nagappan, "Understanding network failures in data centers: measurement, analysis, and implications," in *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4. ACM, 2011, pp. 350–361.

[20] A. Cassandra, "Cassandra 1.0 thrift configuration," https://github.com/apache/cassandra/blob/cassandra-1.0/interface /cassandra.thrift.