Cooperative Monitoring for Internet Data Centers

Kuai Xu Feng Wang
Arizona State University
Division of Mathematical and Natural Sciences
New College of Interdisciplinary Arts & Sciences
P.O. Box 37100, Phoenix, AZ 85069
{kxu01, fwang25}@asu.edu

Abstract

Recent outages in several Web services have demonstrated the potential damage of availability disruptions, since millions of end users rely on these services powered by thousands of servers in large-scale Internet data centers. As Internet data centers continue to grow in scale and complexity, it has become a daunting task to monitor and manage thousands of servers simultaneously. This paper presents a cooperative monitoring framework to continuously monitor availability of thousands of servers in Internet data centers. We propose a simple yet effective algorithm for locating monitor nodes for the purposes of load balancing and resilience, and demonstrate the performance of this method through simulations based on dataset collected from a large Internet content provider. The results show that the monitoring load are well divided among the servers in the data centers, and cooperative monitoring adapts gracefully to the events of monitoring node failures.

1 Introduction

Recent outages in Amazon storage service [4] and other Web or cloud computing services have demonstrated the potential damage of availability disruptions, as millions of end users and enterprises heavily rely on these services powered by thousands of servers in large scale Internet data centers. To ensure normal services, it is very important to continuously monitor the health of the data centers such as service availability and network performance, and send early warning messages or alerts to system operators. However, as the Internet data centers continue to grow in scale and complexity, e.g., Microsoft doubles the number of servers for Windows Live service every 14 months [13], it has become a daunting task to monitor and manage thousands of servers simultaneously [7].

Most existing approaches for data center monitoring are centralized by deploying a number of dedicated *monitor nodes* (or *monitors* in short), which periodically collect the information of the health status from the servers via SNMP protocol or other methods. Such centralized approaches are very simple and easy to manage, however they lack two important characteristics of monitoring applications: scalability and resilience. Such approaches are not scalable, because the monitoring load for each monitor node increases dramatically as more servers are added into the data centers. In addition, as each node monitors thousands of servers, the polling frequency of each server has be set to a large time window, e.g., 5 or 10 minutes, due to the resource constraint of monitor nodes. More importantly, these monitor nodes often become *single points of failure* for service availability monitoring if they experience hardware or software failures. In other words, centralized monitoring is not robust in the events of monitor node failures. While there exists an extensive body of prior work on Internet or peer-to-peer measurement, there has been very little attempt to build scalable and resilient monitoring framework for large scale Internet data centers which power increasingly popular web services e.g., Google online search, and cloud computing services e.g., Amazon Elastic Compute Cloud (EC2).

In light of the limitations of existing approaches, in this paper we propose a novel approach to address the following question: *how can we monitor large scale Internet data centers in a scalable and resilient fashion, as they continue to grow*. In particular, we propose a cooperative monitoring scheme for monitoring service availability in large scale Internet data centers. The idea of cooperative monitoring is largely inspired by existing cooperative applications in file and storage systems [5] as well as Internet measurment [17, 15, 18, 12], which have demonstrated the benefits of collaborative caching and storage in load balance, resilience and scalability. Commodity PCs are widely deployed in Internet data centers as servers due to the desirable performance/cost ratio, i.e., low cost and powerful process-

978-1-4244-3367-4/08/$25.00 ©2008 IEEE

ing and storage resources. These servers are typically underutilized, as most service providers provision the server capability based on the peak load, and balance the application load across servers to improve the performance. These observations of idle resources also motivate the use of cooperative monitoring among the servers.

To realize the cooperative monitoring, we introduce a simple yet effective algorithm based on distributed hash tables for locating monitor nodes for the servers, and analyze its performance through simulations based on the dataset collected from a large Internet service provider. The results show that the cooperative monitoring approach balances the monitoring load across all servers in the data centers, and is resilient to simultaneous monitor node failures.

The contributions of this paper are two-fold:

- We introduce the problem of scalable monitoring in large scale Internet data centers and present a cooperative monitoring framework for scalable and resilient monitoring;
- We propose a simple yet effective algorithm based on distributed hash tables for the purposes of load balancing and resilience, and demonstrate the feasibility and performance of cooperative monitoring through simulations.

2 Internet Data Center Monitoring

In the recent years, as Internet applications and cloud computing services continue to grow, many service providers have been building large scale Internet data centers to host thousands of servers. Figure 1 shows a typical tree-like network topology of a data center [8, 3, 6] which connects the servers to the Internet through core routers, aggregate and edge switches. The core routers connect to one or more ISPs for Internet connectivity, while aggregation and edge switches form a two layer networking infrastructure to support the network capacity of the large number of servers.

Given the size of data centers, hardware, software and network failures are very common, and such failures could potentially affect a large number of users, if not discovered or responded in time. Therefore, it is very important to continuously monitor the health of the servers and alert system operators immediately when such failures occur. The metrics of interest include network metrics, system metrics and application metrics. The typical network metrics are availability, delay, and losses, while the system metrics include resource utilization such as CPU and memory usage, and I/O throughput. The application metrics largely depends on the service running on the servers. For example, the important metrics of web servers are the request rates from end users as well as open TCP connections, while database transaction statistics, e.g., read or write transactions per second are interesting metrics for database servers. The metrics are collected through a variety of methods. SNMP (Simple Network Management Protocol) is the widely used method to collect system and network performance metrics. Other methods include `ping`, `traceroute`, TCP-based or UDP-based active checks. The time granularity varies depending on the metrics and methods.

Many commercial systems such as HP Openview [1] as well as open source efforts including Nagios [2] are built to provide monitoring services. However, most of these monitoring systems are centralized. The advantage of the centralized method is simple to manage, however such method is not scalable, as the servers in the data centers grow in an exponential rate [13]. More importantly, the monitor nodes themselves are single points of failures when they experience hardware or software failures.

Inspired by existing cooperative applications in storage and file systems which have demonstrated the cooperative advantages for load balancing, scalability and resilience, this paper proposes a collaborative monitoring approach to monitor servers in data centers, in which every server finds its own monitor nodes and also serves as monitors if selected by other servers. There are several advantages of cooperative or collaborative monitoring[1]. First, it is economical, as there is no hardware requirement for dedicated monitor machines. Secondly, collaborative monitoring has better scalability, since additional servers share the monitoring responsibility while introducing monitoring tasks at the same time. Lastly, cooperative monitoring adapts gracefully in the events of the failures of single or multiple monitor nodes. Unlike traditional centralized monitoring approaches, the failures of monitor nodes in cooperative monitoring framework do not affect the majority of server monitoring in the data centers, since the monitoring tasks are split across all servers.

3 Cooperative Monitoring

In this section, we describe the design details of the collaborative monitoring scheme for large scale Internet data centers. First, we introduce a *Hierarchical Monitor Discovery* methodology for locating monitor nodes for each server based on weighed consistent hashing and hamming distance. Subsequently we explain the handling of monitor node failures.

3.1 Searching Monitor Nodes

A fundamental limit of centralized monitoring schemes lies in the fact that the monitor node itself is a single point of

[1] We will use the words cooperative and collaborative interchangeably throughout this paper.

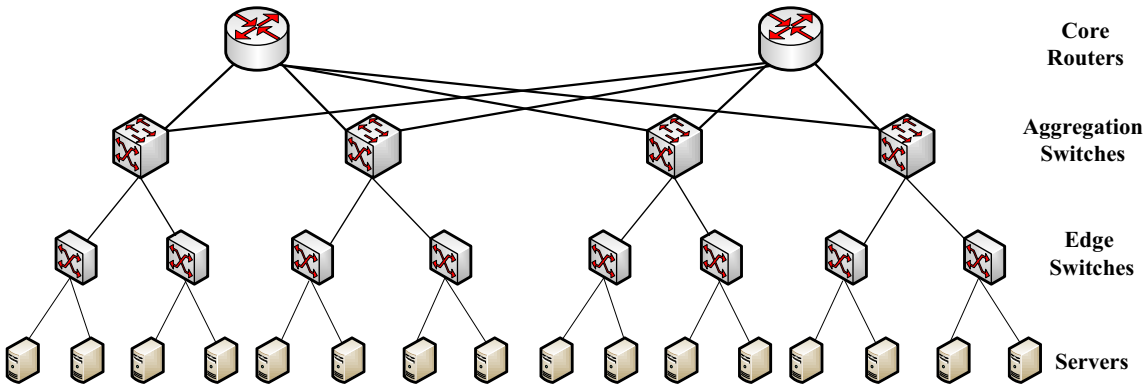


Figure 1. Network topology of an Internet data center.

failure. Thus, an important requirement for locating monitors in collaborative monitoring is to distribute monitoring load across servers. Such load distribution has two desirable advantages: load balancing and resilience against monitor node failures. With a balanced monitoring load, the servers are unlikely to be overwhelmed by the monitoring tasks, so their ability to perform their primary functions, e.g., serving web traffic to end users, is not affected. Given the distribution of monitoring tasks, a single node is only responsible for monitoring a small number of servers. Thus, single or multiple monitor node failures will unlikely disrupt the majority of server monitoring. In other words, cooperative monitoring is resilient against the failures of monitor nodes.

3.1.1 A Simple Method

An intuitive method of distributing the monitoring load is *neighboring monitoring* through maintaining a list of active servers in the data center. Let $S = \{s_1, s_2, \cdots, s_n\}$ denote the servers. Each server in the data center is assigned with a unique identification number from 1 to n, where n indicates the total number of servers. One simple way of assigning monitoring tasks is that each server s_i monitors its *next* neighboring node. Let M_{s_i} represent the servers monitored by s_i. In other words, s_i is a *monitor node*, or *monitor*, of M_{s_i}. Then we have

$$M_{s_i} = \{s_{(i+1) \mod n}\}. \quad (1)$$

If the monitoring redundancy is desired, the method can be modified by requiring each node s_i to monitor its *next* k neighboring nodes, i.e.,

$$M_{s_i} = \{s_{(i+1) \mod n}, \cdots, s_{(i+k) \mod n}\}. \quad (2)$$

Clearly, in the neighboring monitoring approach every server has the same monitoring load, and is monitored by the other k servers at the same time. Theoretically speaking, the solution is optimal in terms of monitoring load balancing. However, the list of active servers in a large Internet data center is constantly changing due to hardware failure [9] or capacity growth [13]. As a result, the identification number for each sever is likely be updated frequently, and the monitoring load of the server will also change. Thus, the neighboring monitoring approach is not adaptive to the server changes, and in the rest of this section we propose a novel monitor discovery method that adapts to such changes.

3.1.2 Hierarchical Monitor Discovery

3.1.2.1 Searching Subnets of Monitor Nodes based on Consistent Hashing

Although the list of active servers are constantly changing in Internet data centers, the subnets of these servers tend to be more stable over time since the subnets are provisioned in advance to accommodate a large number of servers. Given this observation, we propose a *Hierarchical Monitor Discovery* method to locate cooperative monitor nodes for data center servers: 1) search the subnets of candidate monitor nodes, and 2) locate monitor nodes from a given subnet. Each server needs to perform these two steps to find its own monitor nodes, and at the same time each server could be selected as cooperative monitor nodes by other servers. Due to the small number of subnets, it is very easy to collect the list of subnets from an Internet data center and assign a unique number to each subnet. Each server can obtain this list through simple database queries or web services.

The first step of the *Hierarchical Monitor Discovery* approach is to locate the subnet of the monitor nodes. Consistent hashing, also known as distributed hash tables, has been

widely used in peer-to-peer applications and content delivery networks because of its ability to map keys to nodes consistently even during the events of node joining and leaving [10, 11, 16]. For regular hashing functions, the mapping from keys to nodes (the buckets) could change dramatically when the number of buckets changes. For example, for the hashing function $h(x) = (15*x+7) \mod n$, where x denotes a random value, and n denotes the total number of buckets. If the number of buckets increase to $n+1$, then the new hashing value of x, $h'(x) = (15 * x + 7) \mod (n+1)$ will apparently change the nodes for most keys. For Internet service providers, the changes of the subnet could still happen due to server relocations and topology optimization, thus standard hashing functions for searching subnets of monitor nodes are not desirable.

Before we explain how to find the subnets of the monitors for each server, we first summarize the basic idea of consistent hashing below. Given a set of n nodes, $S = \{s_1, s_2, \cdots, s_n\}$, use base hash functions, e.g., SHA-1 or MD5, to map each node s_i, where $i \in (1..n)$, to a β bit identifier ($2^\beta \geq n \geq 2^{\beta-1}$), which can be considered as a point in a circle of [0 .. $2^\beta - 1$]. As a result, the n nodes are randomly distributed along the circle and create n intervals in the circle.

Figure 2[a] illustrates the mapping and the intervals for an example of 5 nodes, A, B, C, D, and E. Given a set of keys, $X = \{x_1, x_2, \cdots, x_m\}$, where m denotes the number of keys and m is typically much larger than n, i.e., $m >> n$, for any given key x_i, we could use the base hash function to find an identifier in the same circle. Clearly, the identifier has to be follow in one of the n intervals created by n nodes. As a result, consistent hashing maps the key to the node which is the closest clockwise neighbor of the identifier.

When a new node F joins, as shown in Figure 2[b], the key which is mapped to the identifier 13 will be re-mapped to F since F is the closest node in the circle to the identifier 13. The mapping of the other keys to nodes is not affected. Similarly, if a node leaves, only the keys, which were mapped to the leaving node, will be re-mapped, while the mapping of the other keys remains unchanged. Let M be the mapping between the keys and nodes, i.e., $M : X \rightarrow S$, $M(x_j) = s_i$. For a given node s_i, use $Y(s_i)$ to denote all the keys which are mapped to s_i, i.e., $Y(s_i) = \{x_j\}$ where $M(x_j) = s_i$, we have

$$|Y(s_i)| = m/n. \tag{3}$$

In other words, the keys are uniformly mapped to all the nodes when m is large.

Through consistent hashing or distributed hash tables, each server finds the subnets of its candidate monitor nodes in a very stable manner even with subnet changes. This stable mapping also leads to a balanced distribution of monitoring load to all subnets. This is a nice property if all subnets have the same number of active servers to share the monitoring load. However, there is no guarantee that active servers are uniformly deployed across all subnets. If we assign the monitoring tasks uniformly across subnets, the hosts in the small subnets will likely get more monitoring loads than hosts in the larger subnets. Thus, it is very important to consider balancing the monitoring load among subnets with varying numbers of active servers.

To balance the load across the subnets, we adopt the *weighted consistent hashing* techniques, also known as *weighted distributed hash tables*, introduced in [14]. In particularly, we assign a weight for each subnet, $w(s_i)$, based on the number of active hosts in the subnet, denoted by $|s_i|$. A natural question will be *how does the cooperative monitoring scheme know the size of all subnets?*. The solution is similar to the collection of the subnet list. Internet data centers can maintain an inventory system that tracks the status of server deployment, so the size of subnets can be generated by querying such an inventory system.

Based on the analysis in [14], the logarithmic method, $w(s_i) = log|s_i|$ where $|s_i|$ denotes the size of the subnet s_i, achieves a fairer load distribution than the linear method, $w(s_i) = |s_i|$ for weighted consistent hashing. Thus, we use $w(s_i) = log|s_i|$ as the preferred weight function. In other words, we construct $log|s_i|$ "virtual nodes" for each subnet, which leads to multiple identifiers for each subnet in the circle. An additional benefit of this weight choice is that the system does not require an accurate statistics of active hosts for subnets. A close estimate of $|s_i|$ serves well for computing the weight $log(|s_i|)$. Therefore, it is acceptable that the inventory system might be updated in a regular fashion.

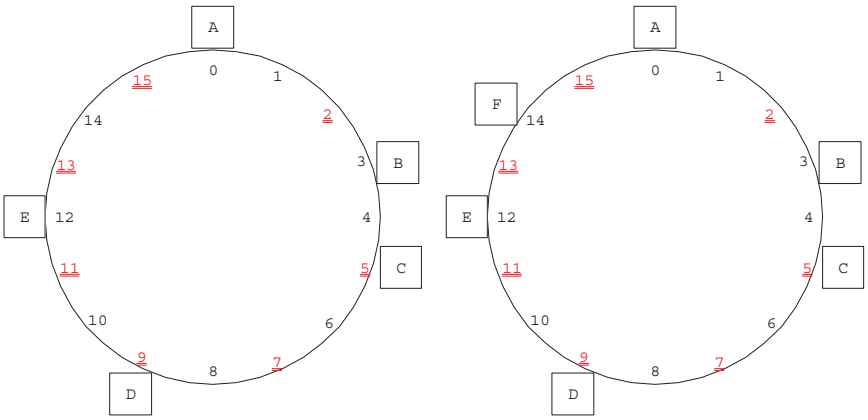
Using the weighted consistent hashing functions, we then have

$$|Y(s_i)| = m * \frac{w(s_i)}{\sum_{i=1}^{n} w(s_i)}. \tag{4}$$

As a result, the monitoring load of each server is more balanced compared with non-weighted consistent hashing method, even though the monitoring load across the subnets varies.

3.1.2.2 Locating Monitor Nodes from a Subnet

The second step of locating monitor nodes is to discover active hosts that could potentially serve as *collaborative monitors*. In our study, we use the *ping broadcast* messages to collect the information of active hosts in the subnet. Once the information of live hosts is collected, the host needs to select one as a cooperative monitor. A simple method is random selection. However, this method could likely lead different selections of monitors among live hosts if the selected node fails and recovers again. In order to achieve stable selections of collaborative monitors, we introduce the ham-



(a) An example of consistent hashing

(b) The mapping of keys and nodes after a new node joins

Figure 2. The ability of consistent hashing for maintaining stable mapping between keys and nodes.

ming distance metrics to compute the distance of IP spaces between hosts and select the potential monitors. For a given IPv4 address X, we can compute its binary string with a length of 32, e.g., $x_1x_2.....x_{32}$. The hamming distance between any two IP address, X and Y, can be represented as

$$z_1z_2.....z_{32} = x_1x_2.....x_{32}\ XOR\ y_1y_2.....y_{32} \quad (5)$$

$$H(X,Y) = number\ of\ bits\ in\ z_1z_2.....z_{32} \quad (6)$$

Based on the hamming distance function, we select the live IP address, which has the shortest hamming distance to the server, as the collaborative monitor node. In case of a tie, we select the one with the smallest last byte. The simulation results show that random selection is slightly better than the distance approach in terms of load distribution, but random selection almost always select different collaborative monitors when monitor nodes fail and recover later. In the simulations, we implement both methods, but use the results based on hamming distance metrics in the evaluations.

3.2 Handling Monitor Node Failures

Handling monitor node failures is a critical step for any robust monitoring system. Servers in data centers could fail due to various reasons including hardware failures, loss of network connectivity caused by switch failures. Thus, it is very normal that servers or their collaborative monitor nodes fail.

When a server fails, the collaborative monitor sends real-time alerts to system operators for further actions. However, if the server itself is a collaborative monitor node for other servers, then those servers monitored by the failing node is left in an unmonitored state. To handle this failure scenario, we require each node to locate k collaborative monitors with one being the *primary* monitor, and the others as *secondary* or *backup* monitors. If the primary monitor fails, one of the secondary monitors can be elected as the primary, and continues to monitor the node. Clearly, multiple monitors increase the resilience of collaborative monitors. Since the probability of multiple monitors for the same server fail at the same time is low, we use $k = 2$ in our simulations.

There are several methods to find multiple monitors. One simple method is that in the second step of locating monitor nodes, we find multiple active hosts in the same subnet. Alternatively, we could employ various base hash functions, e.g., MD5, SHA-1, in the first step to find different subnets. The advantage of the second method is its robustness against the failure of subnets. In the simulations, we use two hash functions, MD5 and SHA-1 as the base hash functions to find multiple monitor nodes.

4 Evaluations

In this section, we evaluate the collaborative monitoring approach through simulations based on the dataset collected from a large Internet content provider, and analyze the performance of our proposed approach.

4.1 Experiment Setup

We evaluate the collaborative monitoring method via simulations. We collect the information of the subnets as well active hosts from an Internet data center of a large Internet content provider. The size of the subnets varies

ranging from /28 CIDR (Classless Inter-Domain Routing) blocks to /24 blocks. In other words, the maximum number of active hosts for these subnets ranges from 16 to 256.

For each active server, we simulate a collaborative monitoring agent that performs the following tasks: i) locating monitor nodes for the server, ii) communicating with the monitors and handling failure scenarios; and iii) serving monitoring requests from other hosts. The first two tasks are implemented for the servers to find monitor nodes. The task of locating monitors implements the methodology described in the previous section. After the host finds the monitors who accept the monitoring request, the agent needs to maintain communication channels with the monitors through periodic "heart-beat" signals. If the agent does not receive signals from a monitor node during a given time period, the agent will assume the particular monitor fails, and immediately starts to handle failure exceptions. The last task is implemented for the servers to become cooperative monitor nodes. The agent waits for monitoring requests from other hosts, and performs continuous monitoring tasks over the hosts such as collecting system and network metrics of the hosts, alerting system operators for failure or anomalous events.

We also build a database which records the information of the subnets as well as the number of active hosts for each subnet. Each agent synchronizes with the database during the scheduled time window to collect the up-to-date information of subnets and their sizes. If the number of subnets changes, or the total number of active hosts in the data center changes beyond a certain threshold (δ), e.g., $\delta \leq 0.1$, each agent re-computes its collaborative monitor nodes based on weighted consistent hashing techniques using the updated *weights* of subnets. In addition we also simulate various failure scenarios to evaluate the resilience or robustness of the cooperative monitoring approach.

4.2 Load Balancing of Server Monitoring

One important observation from a number of repetitive simulations is that the load of server monitoring is well balanced across servers. Figure 3 shows i) the distribution of active hosts within these subnets represented by the solid line; and ii) the distribution of cooperative monitor nodes within these subnets represented by the dash line. Clearly, for each subnet the majority of active hosts serve as collaborative monitors that collect the system and network metrics for at least one other host in the data center. This result suggests that our algorithm of locating monitors for the severs in Section 3 successfully achieves the goal of balancing the cooperative monitors across all subnets.

More importantly, the simulation results also show that the algorithm balances the monitoring load across all active hosts. Figure 4 illustrates the distribution of subnet size vs.

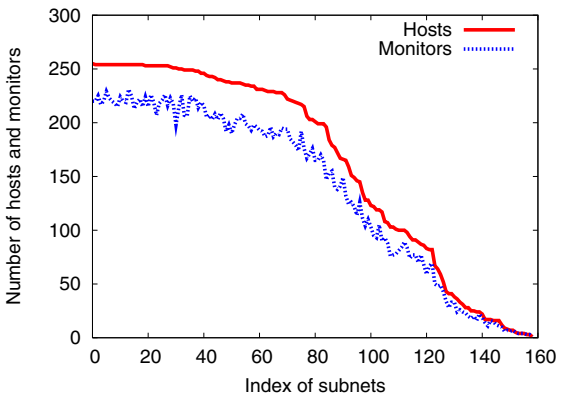


Figure 3. The numbers of active hosts and collaborative monitor nodes in all subnets.

subnet monitoring load, where the size of a subnet represents the total number of active hosts in the subnet, while the total monitoring load represents the total number of nodes monitored by the monitors in the subnet. As shown in Figure 4, the monitoring load grows linearly as the subnet size, which indicates that the monitoring tasks are well divided into subnets based on their size. This balance largely reflects the benefit of weighted consistent hashing, i.e., introducing $log|s_i|$ virtual nodes for each subnet s_i. To validate this conjecture, we further perform similar experiments using non-weighted consistent hashing functions, and find that the monitoring loads are nearly uniformly assigned to each subnet such that active hosts in a smaller subnet tend to be assigned with more monitoring tasks.

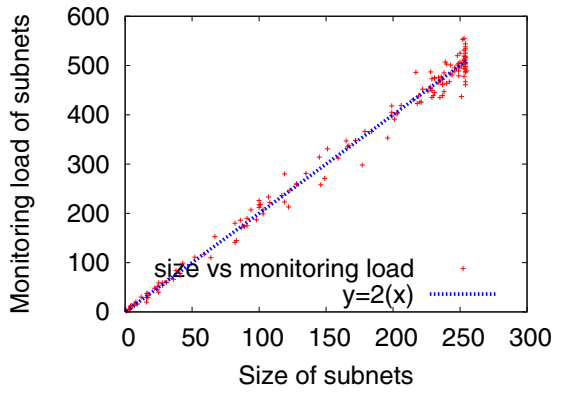


Figure 4. The number of active hosts and the total monitoring load for all subnets.

4.3 Resilience Analysis

Server and subnet failures are very common in Internet data centers because of hardware, software or network reasons. To demonstrate the resilience nature of collaborative monitoring, we simulate various failure scenarios and analyze how cooperative monitoring recovers from these failures. The failure scenarios include single or multiple monitor node failures and single or multiple subnet failures.

4.3.1 Monitor Node Failures

The servers in data centers consist of commodity PCs, and they could experience failures caused by various reasons. Let us assume that the probability of a server failure in a give time is p, which is very small in practice. Apparently, the hosts monitored by the failed servers are also affected by such failures since these hosts likely experience monitoring disruptions, even though the failed servers can be recovered by system administrators within minutes or hours.

There are two possible scenarios for monitor failures: single failure and multiple failures. For a single monitor failure, the affected hosts are those monitored by the monitor. Recall that each host has primary and secondary monitors for redundancy or failover purposes. If the failed monitor is the primary, then the host can immediately use a secondary monitor node as the primary, and search another monitor node as the secondary. Similarly, if the failed monitor node is the secondary, the host simply searches for a new one to replace the secondary monitor.

In case of multiple monitor failures, the collaborative monitoring method uses the same method to recover as in single monitor failures. The main challenge in the scenario lies in the parallel failures of the primary and secondary monitors for the some host, although such probability is fairly small, i.e., p^2. In this case, the hosts cannot simply switch between two failover monitors since both of them failed. The affected hosts have to search for two other monitors in the same subnets of the previous monitors to replace them, thus experience monitoring disruptions. The disruption period is the time between the old monitor fails and the new monitor starts to work. The disruption time period can be significantly reduced if each host caches a list of available active hosts in the same subnets during the initial search process.

Clearly, the failures of single or multiple monitors have a limited impact on continuous monitoring. Next we will study the scenarios of larger failures, i.e., subnet failures.

4.3.2 Subnet Failures

The failure of subnets, e.g., the switch failure or the power loss, not only affect all the active hosts in the failed subnets, but also affect the servers monitored by these active hosts.

When a node discovers that it has not received the heartbeat signals from monitors in the failed subnet, the node performs a simple diagnosis step to verify whether the failure is node failure or subnet failure by contacting the cached list of active hosts in the same subnet. If none of the active hosts responds, then the host determines that the subnet fails[2]. Unlike the monitor failure scenarios, the nodes cannot simply find replacement monitors in the same subnet, and they must remove the failed subnet from the subnet list and locate a new subnet by re-computing the weighted consistent hashing functions.

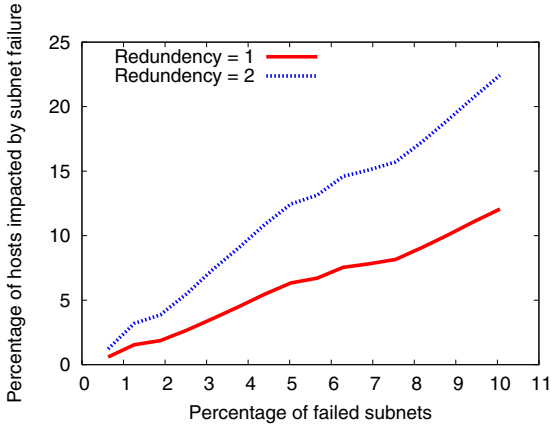
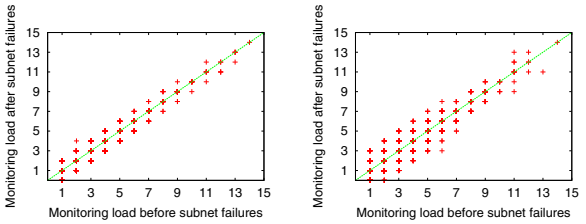


Figure 5. The monitoring impact of subnet failures for the servers.

In the experiments, we simulate the scenarios of failing 1% subnets to 10% subnets in parallel. Figure 5 shows the number of servers that lose one or two monitors during the failures for the cases of $k = 1$ and $k = 2$, respectively. In the case of $k = 1$ (no monitoring redundancy), the percentage of affected server is approximately same as the percentage of the failed subnet. Similarly, for the case of $k = 2$ (monitoring redundancy), the percentage of affected servers is close to two times of the percentage of the failed subnet. These results indicate the impact of subnet failure to continuous monitoring. On the other hand, these affected servers can quickly locate the monitors from other normal subnets to reduce the time of monitoring disruptions.

After all the affected servers find the new monitors, it is not surprising to find the changes of the monitoring load for all monitors in the remaining subnets. Figure 6 shows the changes of monitoring load for the cases of 1 and 4 subnet failures, respectively. Some monitors get less monitoring load as the hosts in failed subnets do not need monitoring any more, while a number of monitors get more load from

[2]It is also possible, although in a very small probability, that all other active hosts become inactive. If this happens, the host can send a ping broadcast message to determine whether there are other newly added hosts in the subnet.



(a) The case of 1 subnet failure (b) The case of 4 subnet failure

Figure 6. The monitoring impact of subnet failures for hosts.

hosts which lose monitor nodes in failed subnets. However, in general the majority of the monitors are assigned to similar monitoring load compared with the load prior to the subnet failures, since the re-assignment of monitoring tasks are also balanced because of weighed consistent hashing approach applied in the algorithm.

To summarize, the simulation results demonstrate the advantages of collaborative monitoring for monitoring scalability and resilience. The monitoring load is fairly balanced across the servers in the data centers. The monitoring approach is resilient against various failures scenarios including monitor node failures and subnet failures.

5 Conclusions and Future Work

This paper proposes a cooperative monitoring approach to continuously monitor thousands of servers in large scale Internet data centers based on weighted consistent hashing algorithms. Compared with existing centralized monitoring methods in data centers, cooperative monitoring has two desired advantages: scalability and resilience. Cooperative monitoring balances the monitoring load across all severs based on consistent hashing techniques, and more importantly is resilient and robust in the events of monitor or subnet failures. We have demonstrated these advantages through simulations based on the dataset collected from a large Internet content service provider. We are currently in the process of implementing a real-time collaborative monitoring system and planning to deploy the system on the PlanetLab test-bed to study its operational feasibility. In addition, we are also looking into the problem of correlating the alert and event streams collected from distributed cooperative monitors to analyze the root causes of hardware, software and network failures in Internet data centers.

References

[1] HP OpenView. `http://www.openview.hp.com`.

[2] Nagios. `http://www.nagios.org/`.

[3] M. Al-Fares, A. Loukissas, and A. Vahdat. A Scalable, Commodity Data Center Network Architecture. In *Proc. of ACM SIGCOMM*, August 2008.

[4] Amazon. Amazon S3 Availability Event: July 20, 2008. `http://status.aws.amazon.com/s3-20080720.html`.

[5] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica. Wide-area cooperative storage with CFS. In *Proc. of ACM Symposium on Operating Systems Principles (SOSP)*, October 2001.

[6] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu. DCell: A Scalable and Fault-Tolerant Network Structure for Data Centers. In *Proc. of ACM SIGCOMM*, August 2008.

[7] J. Moore, J. Chase, K. Farkas, and P. Ranganathan. Data Center Workload Monitoring, Analysis, and Emulation. In *Proc. of Workshop on Computer Architecture Evaluation using Commercial Workloads*, February 2005.

[8] D. Joseph, A. Tavakoli, and I. Stoica. A Policy-aware Switching Layer for Data Centers. In *Proc. of ACM SIGCOMM*, August 2008.

[9] T. Karagiannis, R. Mortier, and A. Rowstron. Network exception handlers: Host-network control in enterprise networks. In *Proc. of ACM SIGCOMM*, August 2008.

[10] D. Karger, E. Lehman, T. Leighton, R. Panigrahy, M. Levine, and D. Lewin. Consistent Hashing and Random Trees: Distributed Caching protocols for Relieving Hot Spots on the World Wide Web. In *Proceedings of ACM Symposium on Theory of Computing*, May 1997.

[11] D. Karger, A. Sherman, A. Berkheimer, B. Bogstad, R. Dhanidina, K. Iwamoto, B. Kim, L. Matkins, and Y. Yerushalmi. Web Caching with Consistent Hashing. In *Proceedings of the Eighth World-Wide Web Conference*, May 1999.

[12] W. Liu and R. Boutaba. pMeasure: A peer-to-peer measurement infrastructure for the Internet. *Computer Communications*, 29(10):1665–1674, June 2006.

[13] PC World. Microsoft: Datacenter Growth Defies Moore's Law. `http://www.pcworld.com/article/130921/article.html`.

[14] C. Schindelhauer and G. Schomaker. Weighted distributed hash tables. In *Proc. of ACM Symposium on Parallel Algorithms and Architectures*, July 2005.

[15] S. Srinivasan and E. Zegura. Network Measurement as a Cooperative Enterprise. In *Proc. of International Workshop on Peer-to-Peer Systems (IPTPS)*, March 2002.

[16] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In *Proc. of ACM SIGCOMM*, September 2001.

[17] R. van Renesse and D. Dumitriu. Collaborative Networking in an Uncooperative Internet. In *Proc. of IEEE Symposium on Reliable Distributed Systems*, October 2002.

[18] X. Zhang, J. Liu, Q. Zhang, and W. Zhu. gMeasure: a group-based network performance measurement service for peer-to-peer applications. In *Proc. of IEEE GLOBECOM*, November 2002.