Journal of the Korean Data & Information Science Society
2011, **22**(5), 989–998

한국데이터정보과학회지

Design and evaluation of a GQS-based time-critical event dissemination for distributed clouds

Ihn-Han Bae[1]

[1]School of Computer and Information Communication Engineering,
Catholic University of Daegu

Received 29 August 2011, revised 15 September 2011, accepted 19 September 2011

Abstract

Cloud computing provides computation, software, data access, and storage services that do not require end-user knowledge of the physical location and configuration of the system that delivers the services. Cloud computing providers have setup several data centers at different geographical locations over the Internet in order to optimally serve needs of their customers around the world. One of the fundamental challenges in geographically distributed clouds is to provide efficient algorithms for supporting intercloud data management and dissemination. In this paper, we propose a group quorum system (GQS)-based dissemination for improving the interoperability of intercloud in time-critical event dissemination service, such as computing policy updating, message sharing, event notification and so forth. The proposed GQS-based method organizes these distributed clouds into a group quorum ring overlay to support a constant event dissemination latency. Our numerical results show that the GQS-based method improves the efficiency as compared with *Chord*-based and *Plume* methods.

Keywords: Distributed clouds, intercloud data management, quorum system, time-critical event dissemination.

1. Introduction

Cloud computing is a recent trend in information technology (IT) that moves computing and data away from desktop and portable personal computers into large data centers. It refers to applications delivered as services over the Internet as well as to the actual cloud infrastructure (Laoutaris *et al.*, 2008). Cloud service providers have to scatter geographically as many data-center containers to support modular data-centers that can provide predictable, repeatable components for the desired organizations. Moreover, IT organizations/enterprises, government/ departments, universities may deploy their own clouds in the future (Decandia *et al.*, 2007). In the future, a lot of distributed clouds, including public and private clouds, will soon be built around the glove.

[1] Professor, School of Computer and Information Communication Engineering, Catholic University of Daegu, Gyeongbuk 712-702, Korea. E-mail: ihbae@cu.ac.kr

Figure 1.1 shows the four entities that typically compose the distributed cloud computing ecosystem: end user, cloud user, cloud provider, and cloud applications (Endo *et al.*, 2011). The cloud user located in the middle, between end users and the cloud provider, and is responsible for providing applications. A cloud user can be seen as a service provider, who leaves resources/services offered by the provider in order to host applications that will be consumed by end users. In turn, the end user is the consumer of an application that simply uses applications, generating demand for the cloud. The cloud provider is the owner of the infrastructure. In this way, a provider is responsible for managing physical and virtual resources to host applications. These cloud applications may be of different types (a farm of web servers, a scientific application, etc.) that all have different requirements. For example, in the case of network virtualization (NV), a request for a virtual network (VN) may be represented with constraints associated with nodes and links. For each VN request has to assign virtual resources to be hosted on its physical resources.

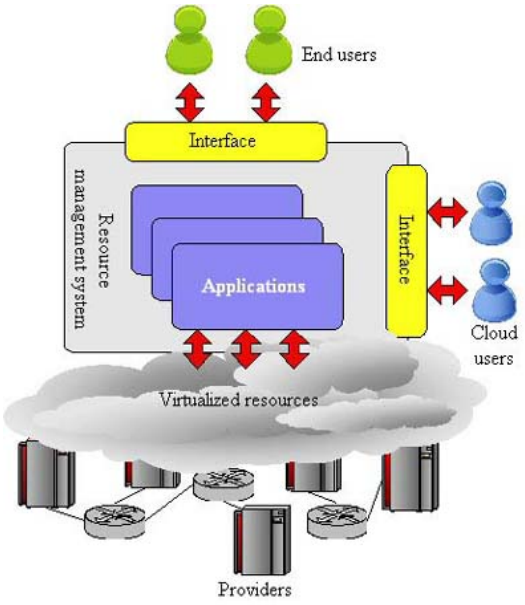

Figure 1.1 Entities in the cloud computing ecosystem

One of the fundamental challenges in large-scale geographically distributed clouds is to provide efficient algorithms for supporting intercloud data management and dissemination. In other words, it is that intercloud applications/systems have to interact with a set of clouds via the Internet. Especially for time-critical services across a large geographic area, these service providers prefer to delivery the events as early as possible. Therefore, as the number of clouds continues to grow, the scale of distributed clouds may achieve a planetary-scale, with even thousand or hundred thousands of clouds. One of major challenges in these planetary-scale distributed clouds is to promptly disseminate the time-critical event to other clouds (Wu *et al.*, 2011).

A typical time-critical service example is the disaster warning system which has a disaster

alert cloud for hosting the disaster early warning system. The alert notification for the system should be quickly disseminated to clouds to minimize the damage.

In this paper, we present a GQS-based dissemination for improving the interoperability of intercloud in time-critical event dissemination service, such as computing policy updating, message sharing, event notification and so forth. The proposed GQS-based method organizes these distributed clouds into a group quorum ring overlay to support a constant event dissemination latency. Our numerical results show that the GQS-based method improves the efficiency as compared with *Chord*-based and *Plume* methods.

The remainder of this paper is organized as follows. Section 2 describes the background and the related works on disseminating time-critical message to distributed clouds. Section 3 designs the GQS-based time-critical event dissemination. Section 4 presents performance results on our proposed GQS-based, *Chord*-based and *Plume* methods. Section 5 concludes the paper and discusses future work.

2. Related works

To manage and deliver contents effectively, distributed clouds have to organize themselves into a federated intercloud overlay network. Currently, each large-scale cloud service provider uses its own approach to manage or deliver contents. For instance, Amazon's Dynamo leverages a fully-meshed topology for managing data across multiple data centers (Dikaiakos *et al.*, 2009). To meet the demand of time-critical services, this overlay requires a fundamental property, which is that scalability, constant dissemination delay and robustness in the intercloud overlay should be considered carefully. The fully-meshed design is the simplest way to disseminate time-critical message to each other nodes. This scheme takes constant transmission delay easily. However, each node in the fully-meshed design is required to maintain $O(n)$ connections with other nodes in a distributed system of n nodes. Thus this scheme does not scale well in large-scale distributed systems.

Chord (Stoica *et al.*, 2001) is a scalable system for node lookup in a dynamic peer-to-peer system with frequent node arrivals and departures. In this way it is similar to other distributed hash tables (DHTs). *Chord* its supports just one operation: given a key as part of a large, circular key space $(0..2^{160} - 1)$, it maps the key to a node. It achieves this in a scalable manner by limiting the routing information each node needs to only a few other nodes. Because the routing table is distributed, a *Chord* node communicates with its neighbours in order to perform a lookup. In the steady state with n-node system, each node maintains information about only $O(logn)$ other nodes, and resolves all lookups via $O(logn)$ messages to other nodes. *Chord* maintains its routing information as nodes join and leave the system.

Plume (Wu *et al.*, 2011) is a generic distributed intercloud overlay for time-critical event dissemination services. *Plume* organizes these distributed clouds into a novel quorum ring overlay to support a constant event dissemination latency. The *Plume* nodes can construct the quorum ring overlay (QRO) topology according to their ring edges and q-tables respectively. The size of ring edges plus q-table in some *Plume* nodes is $2\sqrt{n}$, where n represents the size of an intercloud network.

3. GQS-based time-critical event dissemination

3.1. System model

We consider an intercloud network with a set of geographically distributed clouds connected via the Internet, and each cloud has a cloud manager (resource management system), applications and a cloud node (provider) as shown earlier in Figure 1.1.

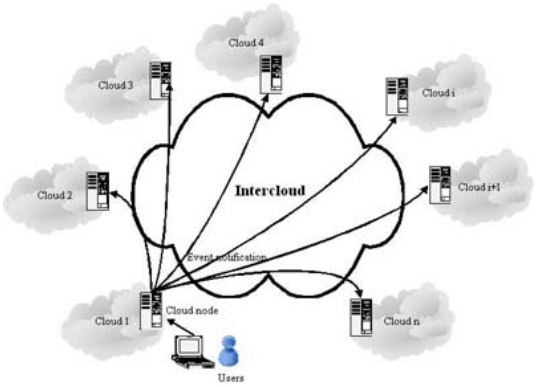

Figure 3.1 An overview of a distributed clouds

Figure 3.1 shows a simple overview of a distributed clouds for one-to-many event dissemination scenario. The distributed clouds are organized into an intercloud overlay via cloud nodes. If a time-critical event is occurred in a cloud 1, the event must be disseminated to all clouds via the intercloud overlay.

Formally, an intercloud network can be represented an undirected graph $G = (V, E)$, where vertices V corresponding to cloud nodes and edges E corresponding to unidirectional overlay links. It is assumed that distributed clouds are uniquely identified. An edge $e(u, v) \in E$ indicates that u and v know each other and can communicate directly through the intercloud overlay.

3.2. GQS Construction

In a proposed algorithm, the dissemination operation for the time-critical event of a cloud node is performed based on GQS. GQS can be constructed from the unfolded surface quorum system with cloud nodes in a distributed clouds as shown Figure 3.1. The definition of GQS is as follows (Joung, 2003).

Definition 3.1 : Let $P = \{1, 2, \cdots, n\}$ be a set of nodes. An m-GQS $C = (C_1, C_2, \cdots, C_m)$ over P consists of m sets, where each $C_i \in 2^P$ is a set of subset of P satisfying the following properties :

intersection: $\forall 1 \le i,\ j \le m, i \ne j,\ \forall Q_1 \in C_i,\ \forall Q_2 \in C_j :\ Q_1 \cap Q_2 \ne \pi$

minimality: $\forall 1 \le i \le m,\ \forall Q_1,\ Q_2 \in C_1,\ Q_1 \ne Q_2;\ Q_1 \nsubseteq Q_2$

Each C_i is called a cartel and each Q is called a quorum.

If we spread three sides of the cubic on a plane, the 3-group quorum system, $S_3 = (C_1, C_2, C_3)$ can be constructed as shown in Figure 3.2. Each quorum in C_1 corresponds to the vertical line crossing the right row of the square. Moreover, each quorum in C_2 corresponds to the horizontal line across the top square, and a vertical line across the left square on the bottom. Finally, each quorum in C_3 corresponds to the horizontal line cross the two squares on the bottom.

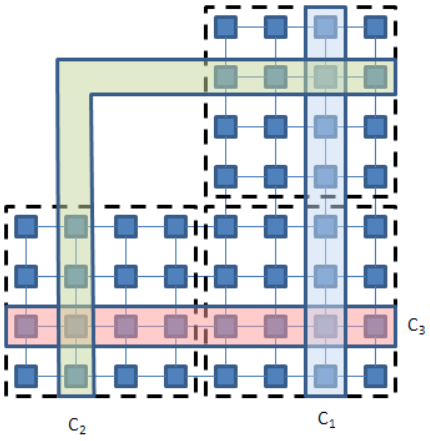

Figure 3.2 Unfolded surface quorum system S_3

Each C_i in S_3 needs 2 squares, each of which is to be shared with one of the other 2 cartels so that the corresponding lines of the two cartels intersect at exactly one node on the square. Overall, there are 3 squares. Let k be the width of each square, Then, each square consists of k^2 nodes, the total number of nodes on the 3 squares is $3k^2$. Let $3k^2 = n$, where n represents the total number of cloud nodes composing a distributed clouds. Therefore, $k = \sqrt{n/3}$. The size of quorum is $q = 2k = 2\sqrt{n/3}$.

3.3. Event dissemination method

The proposed event dissemination method is based on the intercloud which organizes the nodes of distributed clouds into the intercloud overlay based on GQS ring overlay (GRO) topology. In the GQS, we can construct 3-GQS $S_3 = (C_1, C_2, C_3)$ by the staircase construction of the surficial quorum system as shown in Figure 3.2.

Figure 3.3 shows a GRO which consists of 12 cloud nodes and 3 squares, where each square consist of 4 cloud nodes and the dimension of squares is 2. In the all squares of the GQS, the cloud nodes are arranged in the elements of the squares by row-wise.

In Figure 3.3, the quorum of cloud node 3, Q_3 is consisted of C_1 and C_3, where $C_1 = \{1, 3, 5, 7\}$ and $C_3 = \{3, 4, 11, 12\}$. Cloud node 3 disseminates a time-critical event (e) to all cloud nodes within Q_3 with calling $Send(e, 3, p)$, $\forall p \in C_1 \cup C_3$. Then, the cloud nodes 1, 5 and 7 receive the event from the cloud node 3 and relay the event to the cloud nodes $2 \in Row_1$, $6 \in Row_5$ and $8 \in Row_7$. Also, the cloud nodes 11 and 12 receive from the cloud node 3 and relay the event to the cloud nodes $9 \in Col_{11}$ and $10 \in Col_{12}$, where Row_i and Col_i represent the row entries and the column entries in the square that includes

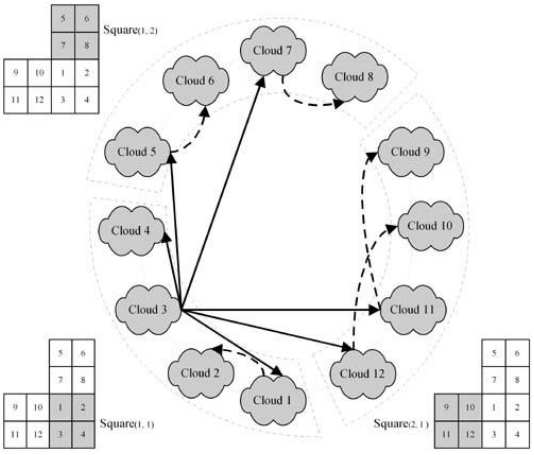

Figure 3.3 An example of GRO of 12 cloud nodes

cloud node i, respectively. Therefore, the proposed GQS-based event dissemination method ensures that any cloud node only needs at most 2 hops to reach any other cloud nodes in the distributed clouds. From this instance, we know that every event relays of the received nodes are bounded on their local square. Thus, the dissemination latency of the event relay will be shorten.

We design a GQS-based event dissemination method, in which time-critical events are disseminated to all cloud nodes via the GQS on the GRO topology. The intercloud which is based on GRO topology can typically provides quick event dissemination in large-scale distributed clouds.

The event dissemination algorithm can be invoked by any cloud node to initiate a dissemination operation. The event dissemination algorithm is represented in Algorithm 3.1, where a cloud node s requires to disseminate an event e to every other nodes in a distributed clouds. Algorithm 3.2 represents the $relay(e)$ operation in a cloud node r that receives the event e from s, where $C_{(r,1)}, C_{(r,2)}$ and $C_{(r,3)}$ represent the group quorums C_1, C_2 and C_3 of the receiver r, respectively.

```
/* (i,j ) represents the index of the square that is in sender s */
if (i == j) then
  for all cloud node p ∈ C1 ∪ C3 do
    Send (e, s, p);
  end for
else if (i > j) then
  for all cloud node p ∈ C3 ∪ C2 do
    Send (e, s, p);
  end for
else
  for all cloud node p ∈ C1 ∪ C2 do
    Send (e, s, p);
  end for
end if
```

Algorithm 3.1 *Dissemination(e)* algorithm

```
/* Cloud node r receives an event e from s */
if (s ∈ Square_(1,1)) then
  if (r ∈ (C_(s,3) − Square_(1,1))) then
    for all cloud node q ∈ Col_r do
      Send (e, r, q);
    end for
  else if (r ∈ C_(s,1)) then
    for all cloud node q ∈ Row_r do
      Send (e, r, q);
    end for
  else
    return;
  end if
  else if (s ∈ Square_(2,1)) then
  if (r ∈ (C_(s,2) − Square_(2,1))) then
    for all cloud node q ∈ Col_r do
      Send (e, r, q);
    end for
  else if (r ∈ C_(s,3)) then
   for all cloud node q ∈ Col_r do
      Send (e, r, q);
    end for
  else
  return;
  end if
else
  if (r ∈ (C_(s,2) − Square_(1,2))) then
    for all cloud node q ∈ Row_r do
      Send (e, r, q);
    end for
  else if (r ∈ C_(s,1)) then
    for all cloud node q ∈ Row_r do
      Send (e, r, q);
    end for
  else
    return;
  end if
end if
```

Algorithm 3.2 *Relay(e)* algorithm

4. Performance evaluation

In this section, we present the performance of the proposed GQS-based event dissemination method through an analytical model. The performance is evaluated in terms of an average dissemination delay. In a stable intercloud networks of n nodes, the sender cloud node disseminates the events to the two quorums of its GQS with one hop, The size of two quorums is $4\sqrt{n/3}-1$. And the cloud nodes which exclude from the nodes of two quorums receive the event with two hops, the total number of the cloud nodes is $(n-4\sqrt{n/3})$. Thus, the total dissemination delay in the GQS-based method is approximately equal to $1\times(4\sqrt{n/3})+2\times(n-4\sqrt{n/3})=2(n-2\sqrt{n/3})$ and the average dissemination delay of the GQS-based method is $2(n-2\sqrt{n/3})/n-1$.

Figure 4.1 shows the results of the performance evaluation of *Chord*-based method and our GQS-based method in terms of average dissemination delay over number of nodes in distributed clouds. As shown in Figure 4.1, the average dissemination delay of GQS-based method does not grow with the number of nodes in distributed clouds. However, the average dissemination operation of *Chord*-based method requires $(logn)/2$ number hops to delivery event to other clouds (Stoica *et al.*, 2001; Terpstra *et al.*, 2003). Therefore, the average

dissemination delay of GQS-based method is much lower than that of *Chord*-based method regardless of the number of nodes in distributed clouds.

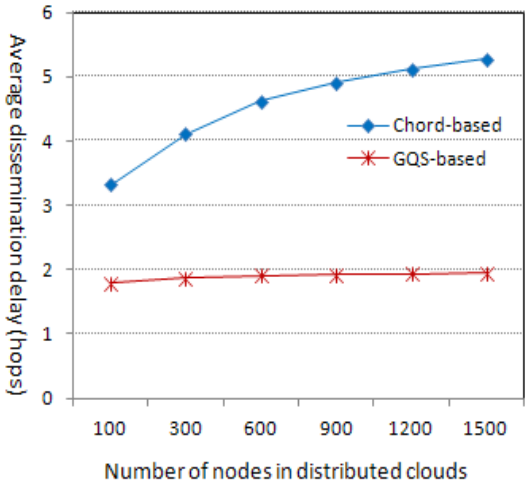

Figure 4.1 Average dissemination delay over number of nodes in distributed clouds with *Chord*-based and GQS-based methods

Figure 4.2 plots the results of the performance evaluation of *Plume* method and our GQS-based method in terms of average dissemination delay over number of nodes in distributed clouds. Where the average dissemination delay of *Plume* method in a stable intercloud network of n nodes is equal to $2(n - \sqrt{n})/(n - 1)$ (Wu *et al.*, 2011). Therefore, the average dissemination delay of GQS-based method is little lower than that of *Plume* method regardless of the number of nodes in distributed clouds. From these results, we know that GQS-based method provides a suitable technique for time-critical event dissemination in large-scale distributed clouds.

5. Conclusions

Cloud computing is a style of computing where dynamically scalable and virtualized resources are provided as a service over the Internet. The cloud refers to the datacenter hardware and software that supports a clients needs, often in the form of datastores and remotely hosted applications. These infrastructures enable companies to cut costs by eliminating the need for physical hardware, allowing companies to outsource data and computations on demand. The Intercloud is based on the key concept that each single cloud does not have infinite physical resources. If a cloud saturates, the computational and storage resources of its infrastructure, it would not be able to satisfy further requests for service allocations sent from its clients. The intercloud scenario aims to address such a situation, and in theory, each cloud can use the computational and storage resources of the infrastructures of other clouds.

One of the fundamental challenges in large-scale geographically distributed clouds is to provide efficient algorithms for supporting intercloud data management and spread. Accord-

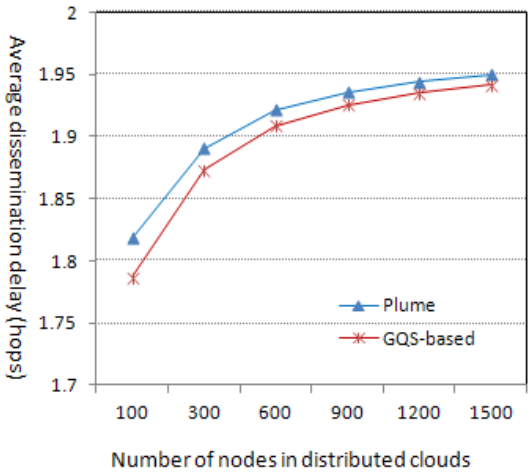

Figure 4.2 Average dissemination delay over number of nodes in distributed clouds with *Plume* and GQS-based methods

ingly, we present the GQS-based event dissemination method for improving the interoperability of intercloud in time-critical event dissemination service. The proposed GQS-based method organizes these distributed clouds into a group quorum ring overlay to support a constant event dissemination latency. The performance of the proposed GQS-based event dissemination method is evaluated through an analytical model. From the results of the analytical evaluation, we know that the average dissemination delay of GQS-based method is lower than those of *Chord*-based and *Plume* methods regardless of the number of nodes in distributed clouds. Therefore, the performance of the GQS-based method is superior to those of *Chord*-based and *Plume* methods. Also, the proposed GQS-based method provides a suitable technique for time-critical event dissemination in large-scale distributed clouds.

Our future works include studying on a hybrid time-critical event dissemination method that combines fully meshed method with GQS-based method by squares in group quorum systems.

References

DeCandia, G., Hastorun, D., Jamd, M. and Vogels, W. (2007). Dynamo: Amazon's highly available key-value store. *Proceeding of 21st ACM Symposium on Operating Systems Principles*, 205-220.

Dikaiakos, M. D., Katsaros, D., Mehra, P., Pallis, G. and Vakali, A. (2009). Cloud computing: Distributed internet computing for IT and scientific research. *IEEE Internet Computing*, **13**, 10-13.

Endo, P. T., de Almeida Palhares, A. V., Pereira, N. N., Goncalves, G. E., Sadok, D., Kelner, J., Melander, B. and Mangs, J.-E. (2011). Resource allocation for distributed cloud: Concepts and research challenges. *IEEE Network Magazine*, **25**, 42-46.

Joung, Y-J. (2003). Quorum-based algorithms for group mutual exclusion. *IEEE Transaction on Parallel and Distributed Systems*, **14**, 463-476.

Laoutaris, N., Rodriguez, P. and Massoulié, L. (2008). ECHOS: Edge capacity hosting overlays of nano data centers. *ACM SIGCOMM Computer Communication Review* , **38**, 51-54.

Stoica, I., Morris, R., Karger, D., Kaashoek, M. F. and Balakrishnan, H. (2001) Chord: A scalable peer-to-peer lookup service for internet applications. *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, 149-160.

Terpstra, W. W., Behnel, S., Fiege, L., Zeidler, A. and Buchmann, A. P. (2003). A peer-to-peer approach to content-based publish/subscribe. *Proceedings of the 2nd International Workshop on Distributed Event-Based Systems*, 1-8.

Wu, C-J., Ho, J-M. and Chen, M-S. (2011). Time-critical event dissemination in geographically distributed clouds. *IEEE Conference on Computer Communications Workshops*, 654-659.