International Journal of Research
Available at https://edupediapublications.org/journals
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 03 Issue 14
October 2016

Search Service Which Is Based On the Reliable and Efficient Content Publish / Subscribe Systems

[1]GURRAM RAJASEKHAR, [2]P NAGESWARA RAO

[1] PG Scholar , Department Of CSE. Gandhi Academy Of Technical Education, Ramapuram (kattakommu Gudem), Chilkur(M), Kodad, Telangana 508206.

[2]Assistant Professor, Department Of CSE. Gandhi Academy Of Technical Education, Ramapuram (kattakommu Gudem), Chilkur(M), Kodad, Telangana 508206

ABSTRACT— Characterized by the increasing arrival rate of live content, the emergency applications pose a great challenge: how to disseminate large-scale live content to interested users in a scalable and reliable manner. The publish/subscribe (pub/sub) model is widely used for data dissemination because of its capacity of seamlessly expanding the system to massive size. However, most event matching services of existing pub/sub systems either lead to low matching throughput when matching a large number of skewed subscriptions, or interrupt dissemination when a large number of servers fail. The cloud computing provides great opportunities for the requirements of complex computing and reliable communication. In this paper, we propose SREM, a scalable and reliable event matching service for content-based pub/sub systems in cloud computing environment. To achieve low routing latency and reliable links among servers, we propose a distributed overlay SkipCloud to organize servers of SREM.

1.INTRODUCTION

What is cloud computing? Cloud computing is the use of computing resources (hardware and software) that are delivered as a service over a network (typically the Internet). The name comes from the common use of a cloud-shaped symbol as an abstraction for the complex infrastructure it contains in system diagrams. Cloud computing entrusts remote services with a user's data, software and computation. Cloud computing consists of hardware and software resources made available on the Internet as managed third-party services. These services typically provide access to advanced software applications and high-end networks of server computers.

2 DESIGN OF SREM

2.1 CONTENT-BASED DATA MODEL

SREM uses a multi-dimensional content-based data model. Consider our data model consists of k dimensions A1;A2; . . .;Ak. Let Ri be the ordered set of all possible values of Ai. So, V ¼ R1 _ R2 __ _ __Rk is the entire content space. A subscription is a conjunction of predicates over one or more dimensions. Each predicate Pi specifies a continuous range for a dimension Ai, and it can be described by the tuple ðAi; vi;OiÞ, where vi 2 Ri and Oi represents a relational operator (<;_; 6¼;_; >, etc). The general form of a subscription is S ¼ ^ki ¼1Pi. An event is a point within the content space V. It can be represented as k dimension-value pairs. By this definition we say an event e matches S if each predicate of S satisfies some pairs of e.

2.2 OVERVIEW OF SREM

To support large-scale users, we consider a cloud computing environment with a set of geographically distributed datacenters through the Internet. Each datacenter contains a large number of servers (brokers), which are managed by a datacenter management service such as Amazon EC2 or OpenStack.

We illustrate a simple overview of SREM in brokers in SREM as the front-end are exposed to the Internet, and any subscriber and publisher can connect to them directly. To achieve reliable connectivity and low routing latency, these brokers are connected through an distributed overlay, called SkipCloud. The entire content space is partitioned into disjoint subspaces, each of which is managed by a number of brokers. Subscriptions and events are dispatched to the subspaces that are overlapping with them through SkipCloud. Thus, subscriptions and events falling into the same subspace are matched on the same broker. After the matching process completes, events are broadcasted to the corresponding interested subscribers. the subscriptions generated by subscribers S1 and S2 are dispatched to broker B2 and B5, respectively. Upon receiving events from publishers, B2 and B5 will send matched events to S1 and S2, respectively.

One may argue that different datacenters are responsible for some subset of the subscriptions according to the geographical location, such that we do not really need much collaboration among the servers [3], [4]. In this case, since the pub/sub system needs to find all the matched subscribers, it requires each event to be matched in all datacenters, which leads to large traffic overhead with the increasing number of datacenters and the increasing arrival rate of live content. Besides, it's hard to achieve workload balance among the servers of all datacenters due to the various skewed

distributions of users' interests. Another question is that why we need a distributed overlay like SkipCloud to ensure reliable logical connectivity in datacenter environment where servers are more stable than the peers in P2P networks. This is because as the number of servers increases in datacenters, the node failure becomes normal, but not rare exception [18]. The node failure may lead to unreliable and inefficient routing among servers. To this end, we try to organize servers into SkipCloud to reduce the routing latency in a scalable and reliable manner.

Such a framework offers a number of advantages for real-time and reliable data dissemination. First, it allows the system to timely group similar subscriptions into the same broker due to the high bandwidth among brokers in the cloud computing environment, such that the local searching time can be greatly reduced. This is critical to reach high matching throughput. Second, since each subspace is managed by multiple brokers, this framework is fault-tolerant even if a large number of brokers crash instantaneously. Third, because the datacenter management service provides scalable and elastic servers, the system can be easily expanded to Internet-scale.

3.SKIPCLOUD

3.1.TOPOLOGY CONSTRUCTION

Generally speaking, SkipCloud organizes all brokers into levels of clusters., the clusters at each level of SkipCloud can be treated as a partition of the whole broker set.

At the top level, brokers are organized into multiple clusters whose topologies are complete graphs. Each cluster at this level is called top cluster. It contains a leader broker which generates a unique b-ary identifier with length of m using a hash function (e.g. MD-5). This identifier is called ClusterID. Correspondingly, each broker's identifier is a unique string with length ofm þ 1 and shares common prefix of length m with its ClusterID. At this level, brokers in the same cluster are responsible for the same content subspaces, which provides multiple matching candidates for each event. Since brokers in the same top cluster generate frequent communication among themselves, such as updating subscriptions and dispatching events, they are organized into a complete graph to reach each other in one hop. After the top clusters have been well organized, the clusters at the rest levels can be generated level by level. Specifically, each broker decides to join which cluster at every level. The brokers whose identifiers share the common prefix with length i would join the same cluster at level i, and the common prefix is referred to as the ClusterID at level i. That is, clusters at level

i þ 1 can be regarded as a b-partition of the clusters at level i. Thus, the number of clusters reduces linearly with the decreasing of levels. Let " be the empty identifier. All brokers at level 0 join one single cluster, called global cluster. Therefore, there are bi clusters at level i SkipCloud organizes eight brokers into three levels of clusters by binary identifers.

To organize clusters of the non-top levels, we employ a light-weighted peer sampling protocol based on Cyclon [19], which provides robust connectivity of each cluster.

Suppose there are m levels in SkipCloud. Specifically, each cluster runs Cyclon to keep reliable connectivity among brokers in the cluster. Since each broker falls into a cluster at each level, it maintains m neighbor lists. For a neighbor list at the cluster of level i, it samples equal number of neighbors from their corresponding children clusters at level i þ 1. This ensures that the routing from the bottom level can always find a broker pointing to a higher level. Because brokers maintain levels of neighbor lists, they update the neighbor list of one level and the subsequent ones in a ring to reduce the traffic cost. The pseudo-code of view maintenance algorithm is show in

Algorithm 1. This topology of the multi-level neighbor lists is similar to Tapestry [20]. Compared with Tapestry,SkipCloud uses multiple brokers in top clusters as targets to ensure reliable routing.

Algorithm 1. Neighbor List Maintenance

Input: views: the neighbor lists. m: the total number of levels in SkipCloud. cycle: current cycle.

1: j ¼ cycle%ðm þ 1Þ;

2: for each i in [0,m-1] do

3: update views[i] by the peer sampling service based on Cyclon.

4: for each i in [0,m-1] do

5: if views[i] contains empty slots then

6: fill these empty slots with other levels' items who share common prefix of length i with the ClusterID of views[i].

3.2.PREFIX ROUTING

Prefix routing in SkipCloud is mainly used to efficiently route subscriptions and events to the top clusters. Note that the cluster identifiers at level i þ 1 are generated by appending one b-ary to the corresponding clusters at level i. The relation of identifiers between clusters is the foundation of routing to target clusters. Briefly, when receiving a routing

Notations in SkipCloud Nb the number of brokers in SkipCloud m the number of levels in SkipCloud Dc the average degree in each cluster of SkipCloud Nc the number of top clusters in SkipCloud . An example of SkipCloud with eight brokers and three levels. Each broker is identified by a binary

string.request to a specific cluster, a broker examines its neighbor lists of all levels and chooses the neighbor which shares the longest common prefix with the target ClusterID as the next hop. The routing operation repeats until a broker can not
find a neighbor whose identifier is more closer than itself.

Algorithm 2 describes the prefix routing algorithm in
Prefix Routing
1: l ¼ commonPrefixLength(self.ID, event.ClusterID);
2: if (l ¼¼ m) then
3: process(event);
4: else
5: destB the broker whose identifier matches event.ClusterID with the longest common prefix
from self:views.
6:lmax¼ commonPrefixLength (destB.identifier, event.ClusterID);
7: if (lmax _ l) then
8: destB the broker whose identifier is closest to event.ClusterID from view½l_.
9: if (destB and myself is in the same cluster of level l) then
10: process(event);
11: forwardTo(destB, event);

Since a neighbor list in the cluster at level i is a uniform sampling from the corresponding children clusters at level i þ 1, each broker can find a neighbor whose identifier matches at least one more longer common prefix with the target identifier before reaching the target cluster. Therefore, the prefix routing in Algorithm 2 guarantees that any top cluster will be reached in at most logbNc hops, where Nc is the the number of top clusters. Assume the average degree of brokers in each cluster is Dc. Thus, each broker only needs to keep Dc _ logbNc neighbors.

4.HPARTITION

In order to take advantage of multiple distributed brokers, SREM divides the entire content space among the top clusters of SkipCloud, so that each top cluster only handles a subset of the entire space and searches a small number of candidate subscriptions. SREM employs a hybrid multidimensional space partitioning technique, called HPartition, to achieve scalable and reliable event matching. Generally speaking, HPartition divides the entire content space into disjoint subspaces (Section 5.1). Subscriptions and events with overlapping subspaces are dispatched and matched on the same top cluster of To keep workload balance among servers, HPartition divides the hot spots into multiple cold spots in an adaptive manner Table 2 shows key notations used in this section.

4.1.LOGICALSPACE CONSTRUCTION

Our idea of logical space construction is inspired by existing

single-dimensional space partitioning (called SPartition) and all-dimensional space partitioning (called APartition). Let k be the number of dimensions in the entire space V and Ri be the ordered set of values of the dimension Ai. Each Ri is split into Nseg continuous and disjoint segments fRj i; j ¼ 1; 2; . . .;Nsegg, where j is the segment identifier (called SegID) of the dimension Ai. The basic idea of SPartition like BlueDove [15] is to treat each dimension as a separated space, as show in. Specifically, the range of each dimension is divided into Nseg segments, each of which is regarded as a separated subspace. Thus, the entire space V is divided into k _ Nseg subspaces. Subscriptions and events falling into the same subspace are matched against each other

4.2 SUBSCRIPTION INSTALLATION

A user can specify a subscription by defining the ranges over one or more dimensions. We treat S ¼ ^ki ¼1Pi as a general form of subscriptions, where Pi is a predicate of the dimension Ai. If S does not contain a dimension Ai, the corresponding Pi is the entire range of Ai. When receiving a subscription S, the broker first obtains all subspaces Gx1___xk which are overlapping with S. Based on the logical space construction

of HPartition, all dimensions are classified into t individual spacesVj; j 2 ½1; t_. For each Gx1___xk of Vi, it satisfies

xi ¼

s; if Ai 6¼ Ri;Rsi

\ Pi 6¼ ;;

0; ifAi¼ Ri: (1)

V consists of two groups, and each range is divided into two segments. For a subscription

S1 ¼ ð20 _ A1 _ 80Þ ^ ð80 _ A2 _ 100Þ ^ð105 _ A3 _

170Þ ^ ð92 _ A4 _ 115Þ, its matched subspaces are G1200,

G1100 and G0022.

For the subscription S1, its every matched subspace Gx1___xk is hashed to a b-ary identifier with length of m, represented by Thus, S1 is forwarded to the top cluster whose ClusterID is nearest to broker in the top cluster receives the subscription S1, it broadcasts S1 to other brokers in the same cluster, such that the top cluster provides reliable event matching and balanced workloads among its brokers.

5.EXISTING SYSTEM

In traditional data dissemination applications, the live content are generated by publishers at a low speed, which makes many pub/subs adopt the multi-hop routing

techniques to disseminate events. A large body of broker-based pub/subs forward events and subscriptions through organizing nodes into diverse distributed overlays, such as tree based design.

6.PROPOSED SYSTEM

Specifically we mainly focus on two problems one is how to organize servers in the cloud computing environment to achieve scalable and reliable routing. The other is how to manage subscriptions and events to achieve parallel matching among these servers. We propose a distributed overlay protocol, called SkipCloud, to organize servers in the cloud computing environment. SkipCloud enables subscriptions and events to be forwarded among brokers in a scalable and reliable manner. Also it is easy to implement and maintain.

7.CONCLUSION

This paper introduces SREM, a scalable and reliable event matching service for content-based pub/sub systems in cloud computing environment. SREM connects the brokers through a distributed overlay SkipCloud, which ensures reliable connectivity among brokers through its multi-level clusters and brings a low routing latency through a prefix routing algorithm. Through a hybrid multi-dimensional space partitioning technique, SREM reaches scalable and balanced clustering of high dimensional skewed subscriptions, and each event is allowed to be matched on any of its candidate servers.

8.REFERENCES

[1] Dataperminite. (2014). [Online]. Available: http://www.domo. com/blog/2012/06/how-much-data-is-created-every-minute/

[2] F. Benevenuto, T. Rodrigues, M. Cha, and V. Almeida, “Characterizing user behavior in online social networks,” in Proc. 9th ACM SIGCOMM Conf. Internet Meas. Conf., 2009, pp. 49–62.

[3] A. Carzaniga, “Architectures for an event notification service scalable to wide-area networks,” Ph.D. dissertation, Ingegneria Informatica e Automatica, Politecnico di Milano, Milan, Italy, 1998.

[4] P. Eugster and J. Stephen, “Universal cross-cloud communication,” IEEE Trans. Cloud Comput., vol. 2, no. 2, pp. 103–116, 2014. [5] R. S. Kazemzadeh and H.-A Jacobsen, “Reliable and highly available distributed publish/subscribe service,” in Proc. 28th IEEE Int. Symp. Reliable Distrib. Syst., 2009, pp. 41–50.

[6] Y. Zhao and J. Wu, "Building a reliable and high-performance content-based publish/subscribe system," J. Parallel Distrib. Comput., vol. 73, no. 4, pp. 371–382, 2013.

[7] F. Cao and J. P. Singh, "Efficient event routing in content-based publish/subscribe service network," in Proc. IEEE INFOCOM, 2004, pp. 929–940.

[8] S. Voulgaris, E. Riviere, A. Kermarrec, and M. Van Steen, "Sub-2- sub: Self-organizing content-based publish and subscribe for dynamic and large scale collaborative networks," Res. Rep. RR5772, INRIA, Rennes, France, 2005.

[9] I. Aekaterinidis and P. Triantafillou, "Pastrystrings: A comprehensive content-based publish/subscribe DHT network," in Proc. IEEE Int. Conf. Distrib. Comput. Syst., 2006, pp. 23–32.

[10] R. Ranjan, L. Chan, A. Harwood, S. Karunasekera, and R. Buyya, "Decentralised resource discovery service for large scale federated grids," in Proc. IEEE Int. Conf. e-Sci. Grid Comput., 2007, pp. 379– 387.

[11] A. Gupta, O. D. Sahin, D. Agrawal, and A. El Abbadi, "Meghdoot: Content-based publish/subscribe over p2p networks," in Proc. 5th ACM/IFIP/USENIX Int. Conf. Middleware, 2004, pp. 254–273.

[12] X. Lu, H. Wang, J. Wang, J. Xu, and D. Li, "Internet-based virtual computing environment: Beyond the data center as a computer," Future Gener. Comput. Syst., vol. 29, pp. 309–322, 2011.

[13] Y. Wang, X. Li, X. Li, and Y. Wang, "A survey of queries over uncertain data," Knowl. Inf. Syst., vol. 37, no. 3, pp. 485–530, 2013. [14] W. Rao, L. Chen, P. Hui, and S. Tarkoma, "Move: A large scale keyword-based content filtering and dissemination system," in Proc. IEEE 32nd Int. Conf. Distrib. Comput. Syst., 2012, pp. 445–454.

[15] M. Li, F. Ye, M. Kim, H. Chen, and H. Lei, "A scalable and elastic publish/subscribe service," in Proc. IEEE Int. Parallel Distrib. Process. Symp., 2011, pp. 1254–1265.

[16] X. Ma, Y. Wang, Q. Qiu, W. Sun, and X. Pei, "Scalable and elastic event matching for attribute-based publish/subscribe systems," Future Gener. Comput. Syst., vol. 36, pp. 102–119, 2013.

[17] A. Lakshman and P. Malik, "Cassandra: A decentralized structured storage system," Oper. Syst. Rev., vol. 44, no. 2, pp. 35–40, 2010.

[18] M. Sathiamoorthy, M. Asteris, D. Papailiopoulos, A. G. Dimakis, R. Vadali, S. Chen, and D. Borthakur, "Xoring elephants: Novel erasure codes for big data," in Proc. 39th Int. Conf. Very Large Data Bases, 2013, pp. 325–336.

[19] S. Voulgaris, D. Gavidia, and M. van Steen, "Cyclon: Inexpensive membership management for unstructured p2p overlays," J. Netw. Syst. Manage., vol. 13, no. 2, pp. 197–217, 2005.

[20] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. D. Kubiatowicz, "Tapestry: A resilient global-scale overlay for service deployment," IEEE J. Sel. Areas Commun., vol. 22, no. 1, pp. 41–53, Jan. 2004.

[21] Murmurhash. (2014). [Online]. Available: http://burtleburtle. net/bob/hash/doobs.html

[22] A.-M Kermarrec, L. Massoulie, and A. J. Ganesh, "Probabilistic reliable dissemination in large-scale systems," IEEE Trans. Parallel Distrib. Syst., vol. 14, no. 3, pp. 248–258, Mar. 2003.

[23] M. Jelasity, A. Montresor, and € O. Babaoglu, "Gossip-based aggregation in large dynamic networks," ACM Trans. Comput. Syst., vol. 23, no. 3, pp. 219–252, 2005.

[24] Cloudstack. (2014). [Online]. Available: (http://cloudstack. apache.org/)

[25] Zeroc. (2014). [Online]. Available: (http://www.zeroc.com/)

[26] B. He, D. Sun, and D. P. Agrawal, "Diffusion based distributed internet gateway load balancing in a wireless mesh network," in Proc. IEEE Global Telecommun. Conf., 2009, pp. 1–6.

[27] Y. Wang and S. Li, "Research and performance evaluation of data replication technology in distributed storage systems," Comput. Math. Appl., vol. 51, no. 11, pp. 1625–1632, 2006.

AUTHOR'S PROFILE:

GURRAM RAJASEKHAR

PG Scholar , Department Of CSE. Gandhi Academy Of Technical Education, Ramapuram (kattakommu Gudem), Chilkur(M), Kodad, Telangana 508206.

P AGESWARA RAO

Assistant Professor, Department Of CSE. Gandhi Academy Of Technical Education, Ramapuram (kattakommu Gudem), Chilkur(M), Kodad, Telangana 508206