International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2014): 5.611

Multi Agent Model Resource Identification in Cloud Environment by Bloom Filtration

Dodla Snigdha[1], Dr. P. SivaKumar[2]

[1]Computer Science and Engineering, Rise Krishna Sai Gandhi Group of Institutions, Ongole, India.

[2]Computer Science and Engineering, Rise Krishna Sai Gandhi Group of Institutions, Ongole, India

Abstract: ***Regarding the fact that, there are large number of resources in cloud computing system. Developing effective solutions for their queries are very important. The success of cloud computing systems fundamentally relies on the efficient use of resources. Discovering and managing resources is one important step in the system effecting. In this paper, a method of Agent-based resource discovery using Bloom filter is proposed. In this strategy, resources information are stored in a Bloom filter, and then, the Bloom filter is sent to the related broker agent and broker agent sends it to the related database. In addition, users make their requests using their hash values and send them to related broker agent. The information needed for search and discovery is stored inside the Bloom filters and then the achieved Bloom filter is transferred, therefore, the amount data transmitted is reduced significantly. The results show that, in the proposed approach, the number of transferred bits in the network for user's requests, the number of transferred bits in the network for resources information, and the average of consumed time to search and discovery requested resources, approximately are 5 times, 15 times, and 3 times less than the case without using Bloom filter, respectively. The consumed memory, used bandwidth in the network, the time of search and discovery of resources, has improved significantly.***

Keywords: cloud computing, resource discovery in cloud computing, agent based resource discovery, Bloom filter.

1. Introduction

Cloud computing is developed model of distributed processing, parallel processing and grid computing, which make data stored on the cloud [1]. Some of important issues in cloud computing are the less response time, cost in data transmit and network bandwidth. The basic principle of cloud computing is that user data is not stored locally and it is stored in data centers on the internet [2].

The cloud computing is model for easy and comprehensive access to set of resources such as: networks, servers, storage resources and applications. This model can be used for reducing cost, integration of existing hardware and software resources, speeding up to applications and information resources of networks [1,3]. Regarding to developing information technology, doing computing works are needed in anytime and anywhere. Also, needing that people are able to done their complex computing works through services and without having expensive software and hardware. The latest response of technology to these needs, is cloud computing. In fact cloud computing has been designed to access wide range of sharing and using of resources, So resource management and efficiency is one important effective properties in clouds performance.

Because the resources and the number of users in cloud computing systems are increasing, and regarding to high and persistent changes in system, is a dynamic environment, so finding effective and efficient ways to query and discover resources in these systems are necessary. Resource discovery problem is one of the basic functions of the efficiency in cloud computing system which is able to define it as "the process of finding all the available resources for a specific application tasks" [3].

In this paper, an Agent-based resource discovery approach using Bloom filters is used. In this strategy, resources information are stored in a Bloom filter, where the Bloom filter is sent to the related broker agent and broker agent also sends it to the related database. Likewise the users make their requests based on Bloom filter and send them to related broker agent in order to search and discover the requested resources.

The proposed approach is evaluated using three criteria: the number of transferred bits in the network for user"s requests, the number of transferred bits in the network for resources information, and the average of consumed time to search and discovery requested resources that are 5 times, 15 times, and 3 times less than the case without using bloom filter, respectively.

In this regards, remaining sections of this paper are organized as follows: Section II presents related works done in the field of discovering cloud and grid resource; Section III introduces the data structure of Bloom filter; Section IV outlines the proposed method that is used in details; and section V provides conclusion.

2. Related Work

In this part, the related works of resource discovery in cloud and grid computing are described as follows: In [4] has been used agent based resource discovery in cloud computing through a 4-stage resource discovery process (selection, evaluation, filtering, and recommendation). [5] has used agent based resource discovery in cloud computing with develop a multi-agent system that cooperates efficiently by introducing a flexible ontology-based matching. In [6], proposed a hierarchical 2-level structure for resource discovery in grid computing using

Bloom filter, that is improved the consumed time and memory in that system. In [7], a model has been discussed which mainly relies on combined GIS and Bloom filter based on results of grid P2P system based on distributed hash table. In [8] has been discussed a discovery algorithm for grid resource discovery based on agent, which in this method the agents are organized in a graph and discovery algorithm is based on cooperation of multi agents. Scalability and dynamism of grid are supported in this method. In [9], a service discovery protocol has been introduced for ad hoc networks using attenuated Bloom filter. Their model is focused on service discovery in local domains where large number of nodes are connected to each other based on a wireless technology. In this model periodic dissemination is used to keep information update, deleting services which have not been declared for several periods. In [10], has been introduced a grid resource discovery model using the routing network mechanisms which for storing network elements used Bloom filter structure. In this method, to encrypting resources has been used Resource Distribution Framework (RDF) which shows both of resources and queries. Regarding to the importance of resource discovery in computing systems such as grid and cloud systems, and specification of Bloom filter data structure to storing large amount of data briefly, we proposed an approach in cloud resource discovery using Bloom filters that reduced the consumed memory, used bandwidth in the network, the time of search and discovery of resources significantly.

3. Bloom Filter

A Bloom filter is a simple randomized data structure with efficient time and space used for storing and representing a set in order to support membership queries. By efficiency of time and space of the model, one means that less time and space is required for storing elements of a set and doing membership test compared to the other methods. This data structure works mainly based on hash functions [11, 12].

3.1. Standard Bloom filter

Standard Bloom filter includes the m bit-array in which, primarily, all the array cells are set to zero. Adding elements of set $\{x_1, x_2, \ldots x_n\}$ are done according to the following. For each element x_i that is added, different k hash functions $(h_1, .. h_k)$ with the range of {1, …, m} are used which are obtained different k value of hash functions $h_1(x_i)$, …, . $h_k(x_i)$Such hash functions map each elements to a random number of domains. Then, for j=1, 2, …, k the bits of $h_k(x_i)$ are set with value=1. One bit in Bloom filter can be set values several times with value=1, but only the first value setting is effective. A similar approach to adding elements is used to examine a given element in Bloom filter. k hash functions are calculated for element y. If at least one of h_i (y) bits is equal to 0 for i= 1, …, K, this element does not exist in the set, certainly. But, if all bits are equal to 1, then this element exists in the set probably [11, 12].

4. An Overview Of The Proposed Approach

In this paper an agent-based resource discovery approach is proposed. The main components of the system are classified as three categories: resource agent, user agent and broker agents, which establishes the relationship between resource agent and user agent [4]. Also, there is a database to store the received resources of all broker agents, which in case of needing any exchange of information, can do it through the database [5]. This structure is shown in Fig. 1.

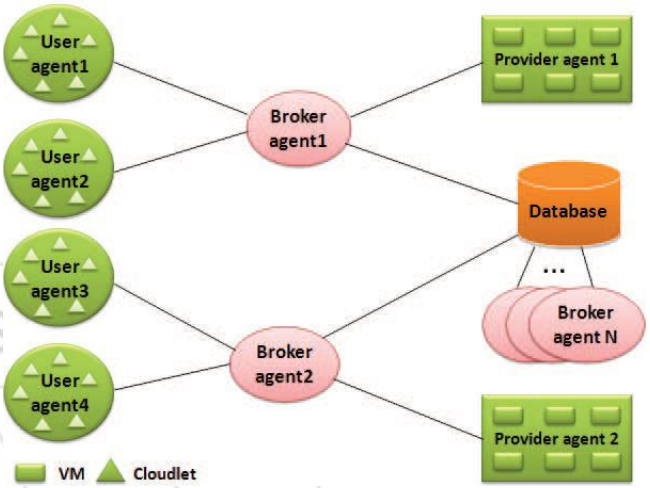

Figure 1: An agent-based Cloud service discovery system[5].

In this strategy, information of VMs1 (resources) are stored in a standard Bloom filter, which is in turn sent to related broker agent. Then, the broker agent sends the Bloom filter into the database for storage. Also users make their requests based on Bloom filter and send them to the related broker agent.

Then, the broker agent searches requests in Bloom filter of VMs, and after discovering appropriate VMs, sends their identifications to the user, otherwise, sends the requests to database. If there are appropriate VMs, the broker sends VMs id to the user, otherwise a message that "there isn't any appropriate VM for the request" is sent to the user. Proposed system is depicted in Fig. 2.

4.1. Proposed Approach for Storing Information of VMs and Users Requests

In the proposed method, standard Bloom filter and CRC16 hash function are used for storing and transmitting the VMs information. To create standard Bloom filter, there is a need for its factors including: size of Bloom filter (m), number of resources (n), and number of hash functions (k). To minimize the false positive probability the formula k = (m/n)*ln2 is used [11].

The procedure of storing VMs in Bloom filter is that, the storing positions of VMs information obtain using k hash functions, then these hash values are set in Bloom filter.

Because the broker agent needs to determine discovered VMs for requests of users, beside storing VM information

in Bloom filter, it is required that VM id to be stored. So, we need to consider a list for storing VMs id2 and in addition each element of this list contains a list of similar VMs id and a string code. In storing procedure, first, indexes that are obtained from k hash functions of VMs information are set to be 1 in the bit-array of Bloom filter. Then, in the list connected to 1st hash function are stored VM id and a string code that contain stored position in Bloom filter in order from 2st to kst position. This way, the information of all VMs are stored in Bloom filter, where it is sent to the broker agent to discover the requested resources of the user. The broker agent, then, sends it to the database. This string code is very effective in facilitating and accelerating the discovery of similar VMs. Because after the result of membership test of requested VM in Bloom filter bit-array proves positive, the broker agent refers to the position of 1st hash function number, and after comparing string code, it easily gains access to the similar VMs id that are stored together in a list and discovers them.

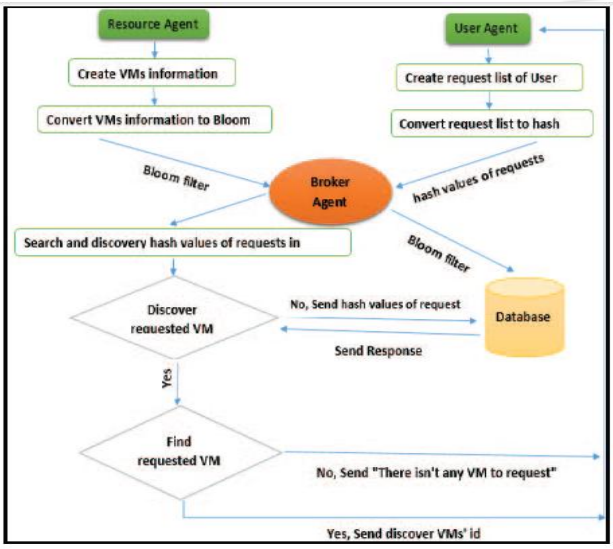

Figure 2: A diagram of the proposed system

Inside the user, for each user task, one request based on the ordered VM is created, and using k hash functions, the requested VM information indexes are obtained. Then, to search these indexes with task id and user id, they are sent to the broker agent. The broker agent, possessing the users" requests and the VMs information, searches and discovers appropriate VMs at the user request. To further illustrate this process, below an example is brought about. We consider a $VM_i$with specification(p) and id(i) that is mapped to 7, 3, …, 14, 10 positions respectively, and these positions are set to be 1 in bit-array of Bloom filter. String code and related VM id are stored in the list connected to the 1st position which is, in this example, h1(p) is equal 3. In this strategy the VMs id similar to $VM_i$are stored in connected list as a whole. If there is a VM with specification (q) where h1(q) is equal 3, its string code is stored in another cell of the list connected to 3st position of Bloom filter. That is shown in Fig. 3. This way, this method provides the possibility that similar VMs are discovered faster. Also, false probability is reduced in VM discovering and VM is discovered at exact request of the user. If the broker agent is not able to find any appropriate VM for users" requests, the request is sent to the database. If there are appropriate VMs, the broker sends their identification to the user, otherwise a message that „there is not any appropriate VM for the request is sent to the user. These procedures are shown in Algorithms 1 and 2. Figure

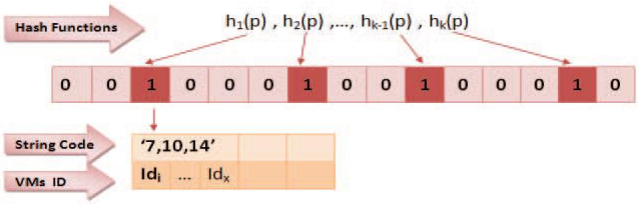

Figure 3: Proposed method of storing VMs information in standard Bloom filter.

Algorithm 1. Pseudocode of storing VMs information in Bloom filter and sending it to related broker agent and database

```
for i ←0 to vmlist.size do
strcharVM ←String(vmCharachterestics)
 for j ← 0 to k-1 d
  bloom_index[j] ← hash_j(strcharVM);
    for z ← 1 to k-1 do
        p ← bloom_index[z];
        strcode ← strcode + "p";
        for t ← 0 to k-1 do
          q ←bloom_index[t];
            Bloomfilter.bitarray[q].set(true);
                Bloomfilter[bloom_index[0]].list.
          add(strcod,VM_id)
Send (Bloomfilter) to related broker
```

Algorithm 2. Pseudocode of creating users requests list and sending it to related broker agent

```
for i ← 0 to requestlist.size do
   strcharRequest ← Strin(requestlist[i].charachterestics)
   for j← 0 to k-1 do
      bloom_index[j] ← hash_j (strcharRequest);
  hash_requestlist[i].add (bloom_index, user_id, task_id);
Send (hash_requestlist) to related broker
```

4.2 The Procedure of VMs Search and Discovery

In this algorithm, the broker agent searches and discovers VMs after the received VMs Bloom filter and user requested list. The broker agent obtains the positions of hash functions on requests and searches the value of these positions in bitarray Bloom filter. There is not any requested VM, if at least one of these positions in Bloom

filter equal 0. But, if anyone positions are not 0, then creates request string code from 2st to kst values of hash functions, next, in connected list of 1st hash function position, searches string code in VMs Bloom filter. If there is requested VM in the list, then sends the appropriate VMs id list to the related user. Otherwise, user request is sent to the database and have been searched requested VM in Bloom filters of broker agents. If there are appropriate VMs is declared to user, otherwise a message that "there isn't any appropriate VM for the request" is declared to the user. In this order this step increases discovery probability of suitable VM for user requests. This procedure is shown in Algorithm 3.

Algorithm 3. Pseudocode of searching and discovering requested VMs in related broker agent

```
Receive (bloom_filter) from resourceAgent
Receive (hash_request list) from userAgent
for i ← 0 to hash_requestlist.size do
    for j← 0 to k-1 do
        p ← hash_requestlist[i].bloom_index[j];
        if bloomfilter.bitarray[p]== false then
            Send (hash_requestlist [i]) to database
            return;
            else
            for z ← 1 to k-1 do
                q ← hash_request list[i].bloom_index[z];
                strreqcode ← strreqcode +"q";
    vmidlist ← Search (bloomfilter[hash_request
    list[i].bloom_index[0] , strreqcode)
    If vmidlist == null then //Recognize false
            positive in     this step
            Send ( hash_request list[i]) to database
            return;
            else
            Send(vmidlist, task_id) to user
```

5. Conclusion

In this paper, regarding to an appropriate technique, the Bloom filter is used in storing VMs information. The proposed approach is evaluated using three criteria: the number of transferred bits in the network for users" requests, the number of transferred bits in the network for resources information and the average of consumed time to search and discovery requested resources. The results show that, in the proposed approach, the number of transferred bits in the network for users" requests, the number of transferred bits in the network for resources information, and the average of consumed time to search and discovery requested resources, approximately are 5 times, 15 times, and 3 times less than the case without using Bloom filter, respectively. It is able to say that the proposed approach is caused that the number of transferred bits in network and the average consumed time to search and discovery requested resources have improved significantly. So Bloom filter is appropriate structure to storing and transmitting resources information in cloud computing systems, especially when network is confronted to high volume of resources and requests. With decreasing these items, have improved in resource discovery procedure in cloud computing.

References

[1] L. Xuning, et al. "Research of campus resource management based on cloud computing." 5th IEEE International Conference on Computer Science and Education, 2010. pp. 1407-1409.
[2] K. Rasmi, and V. Vivek, "Resource Management Techniques in Cloud Environment-A Brief Survey". International Journal of Innovation and Applied Studies, 2013. 2(4): pp. 525-532.
[3] Y. Yuan, and W.C. Liu. "Efficient resource management for cloud computing." IEEE International Conference on System Science, Engineering Design and Manufacturing Informatization (ICSEM), 2011. Vol. 2, pp. 233-236.
[4] K.M. Sim. "Agent-based cloud commerce." IEEE International Conference on Industrial Engineering and Engineering Management, 2009. pp. 717-721.
[5] J. Kang, and K.M. Sim. "Towards agents and ontology for cloud service discovery." IEEE International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2011. pp. 483-490.
[6] R. Tavuseh, and M. Ahmadi. "Optimization of Resources Discovery in Grid Computing Using Bloom Filter." World Applied Programming, 2013. *3*(4),pp.169-177
[7] X. Li, L. Peng and C. Zhang. "Application of Bloom filter in grid information service." IEEE International Conference on Multimedia Information Networking and Security (MINES), 2010. pp. 866-870.
[8] S. Ding, J. Yuan, J. Ju and L. Hu. "A heuristic algorithm for agent-based grid resource discovery." Proceedings of IEEE International Conference on e-Technology, e-Commerce and e- Service, 2005. pp. 222-225.
[9] P. Goering, and G. Heijenk. "Service discovery using Bloom filters." Proceedings of the twelfth annual conference of the Advanced School for Computing and Imaging, June 14-16, 2006. pp.219-227.
[10] J. Li, and S. Vuong. "Semantic overlay network for grid resource discovery." Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing. IEEE Computer Society, 2005. pp. 288-291.

[11] A. Broder, and M. Mitzenmacher, "Network applications of Bloom filters: A survey." Internet mathematics 2004, 1(4): pp. 485-509.
[12] Sh. Geravand, M. Ahmadi. "Bloom filter applications in network security: a state-of-the-art survey". Computer Networks 2013, 57(18): 4047–4064.
[13] Desktop PCs Specification. Available from: http://www.compareindia.in.com/products/desktop-pcs/152

Author Profile

Dodla Snigdha Obtained the B.Tech. degree in Information Technology(IT) from NIMRA Engineering College, ONGOLE. At present persuing the M. Tech in Computer Science and Engineering (CSE) Department at RISE GANDHI Institute Of Technology, ONGOLE.

Dr. P. Siva Kumar Obtained Ph.d degree from Ranchi University. M.Tech. degree in Computer Science and Engineering (CSE) from Andhra University, Visakhapatnam. He is dedicated to teaching field from the last 9 years. He has guided 7 P.G and 24 U.G students. His research areas included Data Mining,. At present he is working as Associate Professor in Rise Group of Institutions, Ongole, Andhra Pradesh, India.