CB-Cloudle: A Centroid-based Cloud Service Search Engine

Shengjie Gong and Kwang Mong Sim

***Abstract*—An increasing number of cloud services has been emerging in the recent years. Amazon, gogrid, rackspace, to name a few, are several most popular cloud service providers. Users need a tool to discover the available options and to suggest the most appropriate alternatives. CB-Cloudle, as a cloud service search engine, could satisfy such users' need. In this work, a centroid-based search engine with the help of a k-means clustering algorithm is designed and developed as a software platform specialised in searching for cloud services, aiming to improve the search effectiveness and efficiency. The centroid-based approaches were applied to search the cloud services with instant response. The k-means clustering algorithm was introduced to discover the groups of similar cloud service entries using a new similarity matrix to calculate the defined distance between cloud service entries. The similarity matrix consists of a non-numeric similarity formula and a numeric similarity formula.**

***Index Terms*—cloud service, search engine, clustering, centroid, similarity matrix.**

I. Introduction

CLOUD service, also known as cloud computing, is "a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction"[1]. Cloud services consist of cloud computing services (e.g., scalable virtual servers), cloud network services (e.g., domain name systems), cloud storage services (e.g., amazon S3), etc.. As a result, cloud service is playing an increasingly important role in such areas as allowing large enterprises to realise enhanced manageability but less maintenance and rapid adjustment of resources, and greatly reducing the cost and risk for startup companies. Instagram is a good example for this. This generates a great deal of interests among popular cloud service providers — amazon, rackspace, gogrid, to name a few — in developing and improving their cloud services. Such increasingly popularity gives rise to provide cloud service users with an easy-to-use cloud service search engine to discover the available cloud services.

General web search engines (e.g., Google, Bing, Baidu) crawl, index and rank different types of web information such as documents, images, videos and so on. The search results are listed based on certain ranking algorithm with the most related ones present on the top of the list. A well-performed web search engine is capable of retrieving the closely relevant information over the whole internet in an efficient and effective way. For instance, Google, as a well known general web search engine, has done a great job in improving both the effectiveness of information retrieval and the efficiency of the query performance. Along with the increasing popularity of the social networks, most of the recent research about general web search engine has been relying on integrating the social connectivity factors into the search results, which is hence named as social search. For example, a semantic social search engine utilising social networks Google+, Twitter and Facebook was described in detail by Zhang et al. [2]; Akiyama et al. [3] took the advantages of both hyperlinks and social links and implemented a parallel library based on Open MPI[1] to achieve the high system performance.

However, general search engines only provide users with providers' links. If a user wants to discover the available options, view the details and even compare the alternatives from different providers, he needs to browse each providers' web pages, analyse the provided cloud service products and compare their prices one by one. CB-Cloudle, however, helps to gather all the specified cloud service information from many different providers and aims to suggest the most appropriate alternatives according to users' requirements. Users, thus, could view all the details of relevant cloud services, compare items and prices, and then make the decision based on all the necessary information.

Currently, most of the popular cloud service providers have been focusing their work on the development of infrastructures and web tools for utilising, deploying and managing the computational resources. CB-Cloudle is a software system designed specifically to search cloud services provided by the cloud service providers rather than searching the whole web information. Kang and Sim [4] first proposed to build a cloud service search engine and was powered by a cloud ontology. By applying the concepts and principles of ontology, they devised the similarity reasoning methods to find the most related or appropriate cloud services — of similar price or instance storage capacity — with more possibilities. Three reasoning methods are supported in [5]: similarity reasoning to differentiate the concepts of cloud services; compatibility reasoning to compare two versions of the same service; and numeric reasoning to calculate the distance between numeric concepts. However, challenges still exist since concepts and categories of cloud services are expanding over time and the corresponding ontology needs to be updated periodically.

This work aims to develop a completely new search engine CB-Cloudle to search and rank the cloud services based on the centroids and k-means clustering algorithm. Centroid are a measure of central tendency of a set of cloud service entries. A service entry is a specification of

Manuscript received December 13th, 2013.

Shengjie Gong is with the School of Computering, University of Kent, Medway Building, Chatham Maritime, Kent, ME4 4AG.

Kwang Mong Sim is with the School of Computering, University of Kent, Medway Building, Chatham Maritime, Kent, ME4 4AG (e-mail: prof_sim_2002@yahoo.com)(Corresponding Author).

[1]A High Performance Message Passing Library(http://www.open-mpi.org/)

cloud service from cloud service providers. Results show that CB-Cloudle presents effective search results and efficient query performance. Therefore, it is of great potential that by introducing the centroid-based search approach and the k-means clustering algorithm, the query and the search performance will be significantly improved.

II. A Centroid-Based Cloud Search Engine

In the previous design of Cloudle by Sim's research laboratory [4][5][6][7][8][9][10][11], the concepts of ontology were used to reason the similarity of cloud services. This work, instead, redesigns Cloudle from the perspective of locating the cloud service entries' centroids combined with the use of k-means clustering algorithm. This method simplifies the query process by only doing calculations of each entry within the most similar cluster, rather than all the cloud service entries. CB-Cloudle's general workflow consists of the following 5 steps:

1) Data preparation: There are several ways for CB-Cloudle to gather data. One straight forward method is to collect data manually, owing to the fact that cloud service data are not very huge compared with the web information in the internet. However, the information of cloud services such as prices change frequently since they are type of virtual resource products. Therefore, manually collected data would not always be up to date. A more efficient way is to crawl automatically and periodically from the cloud service providers;

2) CB-Cloudle initialisation: After all the crawlers have assembled the data, CB-Cloudle then normalise the data by converting them to cloud service entries and filling the missing data with default, most frequent or similar values. For the sake of improving the computational efficiency, data can be preloaded into the memory when they are not so large;

3) Data clustering and centroid generating: Data clustering is the process of grouping similar data (in the sense of cloud service features) into the same groups. CB-Cloudle defines that each cluster (or group) has a centroid, which is generated from entries in the cluster;

4) Query processing: When a user enters a query to look for some suitable cloud services, similarity calculation occurs between the query and each entry whose cluster centroid is the most similar to the query;

5) Results ranking: Query requests are responded instantly and effectively using the centroid-based Cloudle and the most relevant cloud services with its details will be presented on the top of the result list according to the ranking algorithm.

III. Data Preparation and CB-Cloudle Initialisation

CB-Cloudle search engine includes cloud computing service (e.g., amazon EC2) search module, cloud storage service (e.g., amazon S3) search module and so on. An example is given in this section to demonstrate the initialisation of the cloud computing service search module. Fig. 1 shows an overview of the workflow on how CB-Cloudle crawls the cloud service providers and clusters the data. Amazon, rackspace and gogrid are three big and popular cloud computing service providers used in this example. After all the cloud service entries from these three popular cloud service

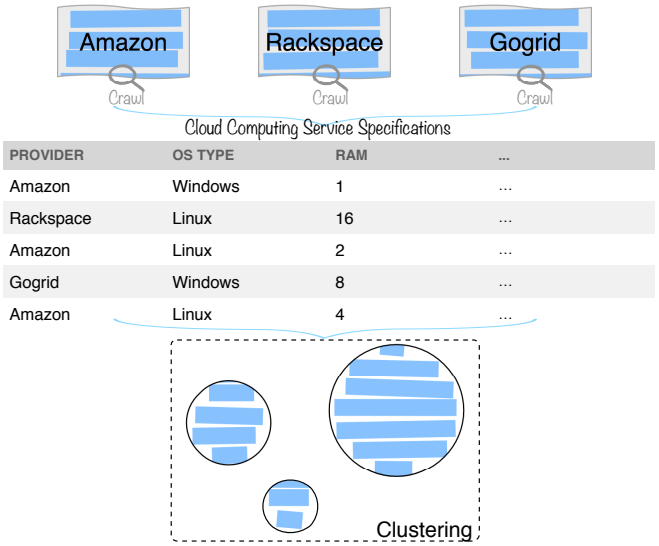


Fig. 1. A general workflow of initialisation

TABLE I
AN EXAMPLE OF CLOUD COMPUTING SERVICE

PROVIDER	gogrid
OS_TYPE	windows
OS_NAME	windows server
OS_VERSION	2008
SQL_SUPPORT	0
VCPU	24
RAM	24
INSTANCE_STORAGE	1200
SSD_STORAGE	0
PRICE_HOURLY	1.44

providers having been crawled, similar entries are grouped into the same cluster for subsequent processing. More details of steps are given as below:

Step 1: Data are needed when CB-Cloudle is deployed on production for the first time. Multiple crawling agents are used to crawl the cloud computing service entries in parallel, with data gathered and normalised automatically and periodically. In this example, 500 cloud computing service entries are collected from the three cloud service providers — amazon, rackspace and gogrid. Fig. 2 gives a partial view of the entry set after the normalisation of the crawling data.

Step 2: The normalised data set obtained in step 1 is inserted into a database and prepare for the clustering algorithm. Often the case, cloud service providers updates the price of cloud services regularly and even introduce some new services. The crawling agents could crawl them periodically to fetch the latest data. The database system used to store the collected data makes the updates easy to achieve. Then, the clustering algorithm runs again to regroup the data.

IV. Clustering Cloud Service Entries

Each entry e is a specification of service from cloud service providers. Taking cloud computing service as an example, Table I shows a windows 2008 virtual server entry provided by gogrid with 24 vCPUs, 24G RAM and 1200G instance storage which costs $1.44 for one hours usage.

ISBN: 978-988-19252-5-1
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

PROVIDER	OS TYPE	OS NAME	OS VERSION	SQL SUPPORT	VCPU	RAM	INSTANCE STORAGE	SSD STORAGE	PRICE HOURLY
amazon	windows	server	2008	1	16	128	1200	80	5.6
amazon	linux	centos	5.7	1	2	2	100	0	0.2
rackspace	linux	RHEL	5.7	0	8	4	800	0	1.4
gogrid	linux	ubuntu	12.04	0	32	64	1000	80	7.2
gogrid	windows	server	2008	0	64	256	1200	160	13.2

Fig. 2. Partial view of the cloud computing service entries

With many cloud service entries, CB-Cloudle uses the k-means clustering algorithm proposed by Hartigan and Wong [12] (see Algorithm 1) to automatically groups them into K clusters in such a way that entries in the same cluster are more similar to each other than to those in the other clusters. The value of K depends on the total number of entries CB-Cloudle has and the number of results presented per page. In this example, with about 500 entries and 25 results presented per page, $K = \frac{500}{25} \times \frac{1}{2} = 10$ is recommended. The reason K is divided by 2 is to increase the possible number of entries in each cluster to be more than 25. The level of similarity between entries $SIM(e, c)$ (see Algorithm 2) is divided into two parts: numeric features' similarity and non-numeric features' similarity. $min_j SIM(e, c_j | j = 1, 2, \ldots, n)$ is to find the most similar centroid c to e, and then label e as in the same cluster of c.

Algorithm 1 k-means Clustering Algorithm

Input: $E = \{e_1, e_2, \ldots, e_n\}$ and K ▷ E is the set of entries to be clustered and K is the number of clusters
Output: $C = \{c_1, c_2, \ldots, c_n\}$ and $L = \{l(e_i) | i = 1, 2, \ldots, n\}$ ▷ C is the set of cluster centroids and L is set of cluster labels of E

1: **function** K-MEANS-CLUSTER(E, K)
2: **for all** $c_i \in C$ **do**
3: $InitializeCentroidRandomly(c_i)$
4: **end for**
5: **repeat**
6: change = false
7: **for** $i = 1$ to n **do**
8: $\hat{k} = min_j SIM(e_i, c_j | j = 1, 2, \ldots, n)$
9: **if** $l(e_i) \neq \hat{k}$ **then**
10: $l(e_i) = \hat{k}$
11: change = true
12: **end if**
13: **end for**
14: **for all** $c_i \in C$ **do**
15: $UpdateCentroid(c_i)$
16: **end for**
17: **until** change = false
18: **return** L, C
19: **end function**

Fig. 3 shows the corresponding diagram of the k-means clustering algorithm presented in Algorithm 1. At the beginning of the clustering process, entries A, B, C, D and E are scattered around. Then the method of $InitializeCentroidRandomly$ generates two random entries as the initial centroids. After computing based on the similarity matrix, entries A and B are found to belong to

Algorithm 2 Similarity Algorithm

Input: An entry e with features $F_e = \{F_e^{non}, F_e^{num}\}$ and a centroid c with features $F_c = \{F_c^{non}, F_c^{num}\}$ ▷ non-numeric features of entry $F_e^{non} = \{f_e^{non}{}_1, f_e^{non}{}_2, \ldots, f_e^{non}{}_i\}(0 \leq i \leq n)$, numeric features of entry $F_e^{num} = \{f_e^{num}{}_1, f_e^{num}{}_2, \ldots, f_e^{num}{}_j\}(0 \leq j \leq n)$, non-numeric features of centroid $F_c^{non} = \{f_c^{non}{}_1, f_c^{non}{}_2, \ldots, f_c^{non}{}_i\}(0 \leq i \leq n)$, numeric features of centroid $F_c^{num} = \{f_c^{num}{}_1, f_c^{num}{}_2, \ldots, f_c^{num}{}_j\}(0 \leq j \leq n)$
Output: the similarity score between the entry e and centroid c

1: **function** SIM(e, c)
2: $i = numberOfNonNumericFeatures$
3: $j = numberOfNumericFeatures$
4: **for** $p = 1$ to i **do**
5: $s^{non} = Sim^{non}(f_e^{non}{}_p, f_c^{non}{}_p)$
6: **end for**
7: **for** $q = 1$ to j **do**
8: $s^{num} = Sim^{num}(f_e^{num}{}_q, f_c^{num}{}_q)$
9: **end for**
10: **return** $0.5 * j/(i + j) * s^{num} + (i/i + j) * s^{non}$
11: **end function**

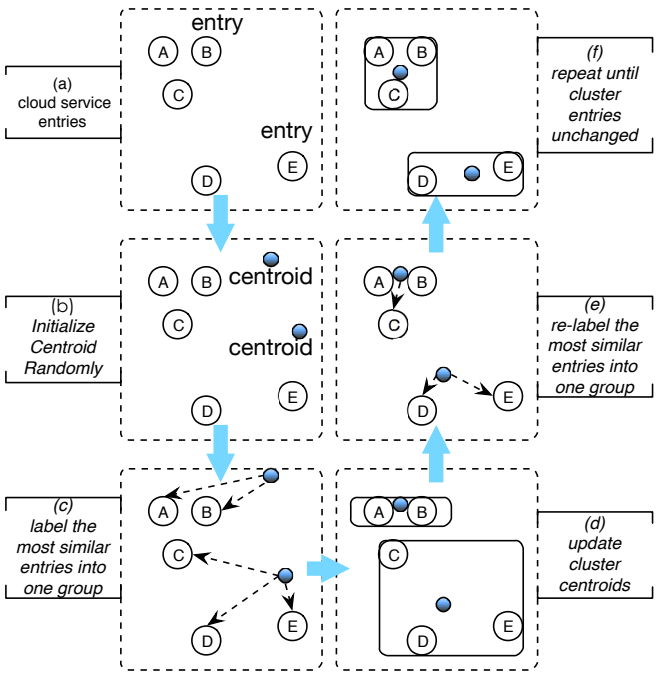


Fig. 3. k-means clustering algorithm diagram

ISBN: 978-988-19252-5-1
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

one group and entries C, D and E belong to another. At this point, the method of $UpdateCentroid$ obtains two new cluster centroids of these two groups (see Section V) and then the similarity computing is carried out again (see Section VI). In this case, entry C is found to be more similar with the former group and thus is regrouped with entries A and B. The process is repeated until both of the cluster centroids do not change(in some sense).

V. Generating centroids

With data set $E = \{e_1, e_2, \ldots, e_n\}$ crawled and preset from cloud service providers, CB-Cloudle randomly places K cluster centroids $C = \{c_1, c_2, \ldots, c_n\}$. An intuitive way to define the centroid of a finite set of points $X = \{x_1, x_2, \ldots, x_n\}$ is the arithmetic mean position of all the points in the set as follows:

$$C = \frac{x_1 + x_2 + \ldots + x_n}{n}.$$

An entry e of the cloud service is not the same as a point x, but a collection of information. An entry may contain both numeric features and non-numeric features. As shown in Table I , PROVIDER, OS_TYPE, OS_ NAME, OS_VERSION are non-numeric features, and the rest — SQL_SUPPORT, VCPU, RAM, INSTANCE_STORAGE, SSD_STORAGE and PRICE_HOURLY — are numeric features. The centroids of numeric features are the mean of each corresponding value. Non-numeric features belong to the type of nominal. In this case, we use the most frequent nominal value as the centroid.

VI. Numeric and non-numeric similarity scores

Based on these cluster centroids, the similarity matrix is designed to increase the chance of locating relevant cloud services to the cluster with centroid. The similarity score of each pair of the entry and the centroid is divided into two parts: the numeric similarity score and the non-numeric score.

A. Numeric score

There are many ways to determine the degree of similarity between two numeric set of $X = \{x_i | i = 1, 2, \ldots, n\}$ and $Y = \{y_i | i = 1, 2, \ldots, n\}$. One common practice is to measure the Euclidean distance between the two sets according to the equation as follows:

$$d = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}.$$

However, the Euclidean distance gives poor results when the data are not well normalised. For example, let X = { SQL_SUPPORT = 0, INSTANCE_STORAGE=1 } and Y = { SQL_SUPPORT = 1, INSTANCE_STORAGE=1000 }, the euclidean distance $d_{euclidean}$ between X and Y will be as follows:

$$d_{euclidean} = \sqrt{(1000 - 1)^2 + (1 - 0)^2} \approx \sqrt{(1000 - 1)^2}.$$

The SQL_SUPPORT feature in X contributes a very small part to $d_{euclidean}$ compared with the INSTANCE_STORAGE feature. However in reality, whether to support SQL or not may have equal importance as the size of instance storage for a user.

A more suitable and sophisticate way to determine the similarity is to use a Pearson correlation coefficient r. The correlation coefficient is a measure of how well two sets of data fit on a straight line. Let $X = \{x_i | i = 1, 2, \ldots, n\}$ and $Y = \{y_i | i = 1, 2, \ldots, n\}$. $\bar{X}$ and $\bar{Y}$ are the mean of X and Y respectively. So the formula $r_{X,Y}$ is as follows:

$$r_{X,Y} = \frac{\sum_{i=1}^{n} (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n} (X_i - \bar{X})^2 \times \sum_{i=1}^{n} (Y_i - \bar{Y})^2}}.$$

The value of $r_{X,Y}$ is in the range of [-1, 1]. To use it as the similarity score between two numeric sets, it can be calculated as follows:

$$Sim_{X,Y}^{num} = 1 - r_{X,Y}.$$

The value of $Sim_{X,Y}^{num}$ is thus in the range of [0, 2] and the smaller of $Sim_{X,Y}^{num}$, the more similarity X and Y have. For example, let $X = \{0, 24, 24, 1200, 0, 1.44\}$ and $Y = \{1, 24, 64, 0, 100, 2.02\}$, $Sim_{X,Y}^{num}$ will be around 1.374, which indicates that this two entries are highly different.

B. Non-numeric score

It is more difficult to quantify the similarity of the non-numeric features than that of the numeric features, because features are sometimes hierarchical and sometimes not. Most of the recent research has been focusing on the semantic similarity among a set of documents or list of items, where the idea of distance between non-numeric terms is based on the likeliness of their meaning or semantic content. Therefore, it can be achieved by defining an ontology — a metric for terms arranged as nodes in a directed acyclic graph. However, given the situation where entries of cloud computing service are not as complicated as document data, the simple intersection of sets can be used to reflect the similarity of two entries. For instance, given two non-numeric sets $X = \{amazon, linux, centos, 5.7\}$ and $Y = \{amazon, linux, ubuntu, 10.04\}$, features both in X and Y is $X \cap Y = \{amazon, linux\}$ and $count(X \cap Y) = 2$. All the feature values in X and Y is $X \cup Y = \{amazon, linux, centos, ubuntu, 5.7, 10.04\}$ and $count(X \cup Y) = 6$. Then the similarity score $Sim_{X,Y}^{non}$ is as follows:

$$Sim_{X,Y}^{non} = 1 - \frac{count(X \cap Y)}{count(X \cup Y)} = 1 - \frac{2}{6} = \frac{2}{3}.$$

The value of $Sim_{X,Y}^{non}$ is in the range of [0,1]. The smaller $Sim_{X,Y}^{non}$ is, the more similar X and Y are.

C. Aggregating numeric and non-numeric scores

To take both measures $Sim_{X,Y}^{num}$ and $Sim_{X,Y}^{non}$ into consideration, a generalised similarity matrix can be defined by taking the weighted average of both measure methods as follows:

$$Sim_{X,Y} = \frac{1}{2}\lambda Sim_{X,Y}^{num} + (1 - \lambda) Sim_{X,Y}^{non},$$

ISBN: 978-988-19252-5-1
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

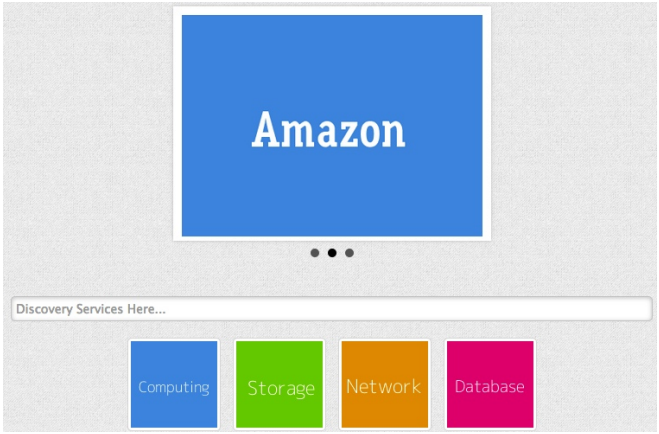


Fig. 4. CB-Cloudle's homepage

where $\lambda \in [0, 1]$. As shown in Table I, the number of numeric features equals 6 and the number of non-numeric features is equal to 4. Consequently, $\lambda = \frac{3}{5}$ is highly recommended. The rationale for dividing $Sim^{num}_{X,Y}$ by two is because the range of it is from 0 to 2 and equal weight should be put on each feature of an entry.

VII. Proof-of-concept example

This section gives an example to present how the users or customers use CB-Cloudle as a cloud service search engine. It only take several simple steps to use a search engine like Google, so does it with CB-Cloudle: the user enters the query first and then the search engine gives the final results based on the ranking algorithms in the way that the most relevant results present on the top. More details of steps are given as below:

Step 1: CB-Cloudle realises the search for cloud computing service, cloud storage service, cloud network service, etc. and the home page of CB-Cloudle is shown in Fig. 4. An example of searching for cloud computing service (a virtual machine) is used to demonstrate how to use CB-Cloudle;

Step 2: CB-Cloudle provides two approaches to search a cloud computing service. The first and easier way is to use CB-Cloudle-providing search templates. Optional choices are given for users to select. Fig. 5 shows a part of the template for searching cloud computing service. For instance, service providers can be filtered and by checking the checkbox of 'amazon', virtual machines provided by amazon may be listed in priority.

Alternatively, the user requirements can be transformed into the CB-Cloudle-defined formula and input as the query. Formula 1 follows the following rule with key-value pairs connected by the symbol of &&.

$$key_1 = value_1[\ \&\&\ key_2 = value_2\ \&\& \ldots] \qquad (1)$$

As shown in Table I , keys can be PROVIDER with corresponding value of amazon, gogrid, or rackspace; OS_TYPE can be linux or windows; OS_NAME can be RHEL (red hat enterprise linux) , windows, centos, debian or ubuntu server version; VCPU, RAM, INSTANCE_STORAGE, SSD_STORAGE can be any positive integer with the default unit of Gigabytes; PRICE_HOURLY shows how much it costs in dollars per hour.

TABLE II
KEY-VALUE PAIRS OF CLOUD-DEFINED FORMULA

key	value
PROVIDER	[amazon, gogrid, rackspace]
OS_TYPE	[linux, windows]
OS_NAME	[RHEL, windows_server, centos, debian, ubuntu]
SQL_SUPPORT	[True, False]
VCPU	Integer
RAM	Integer
INSTANCE_STORAGE	Integer
SSD_STORAGE	Integer
PRICE_HOURLY	Float

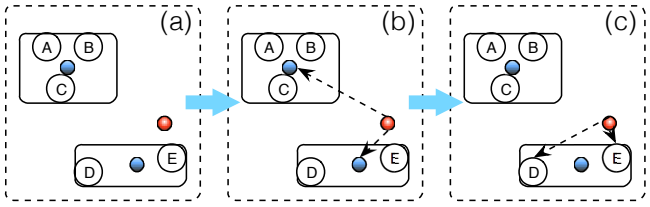


Fig. 6. Ranking process of query results (red point are the requirement entry; blue points are the centroids; A, B, C, D, E are the cloud service entries.)

The formula "PROVIDER=amazon && RAM = 2 && SQL_SUPPORT=True && Price_HOURLY=1", for example, can be used when a user wants to search virtual machines provided by amazon with sql support but costs no more than 1 dollar per hour.

Step 3: After receiving the user's query in either approach, the query string of user requirements is converted into the format of an entry. The requirement entry is then compared one by one with the cluster centroids obtained in the data normalisation step based on the similarity matrix as shown in Fig. 6 , and a cluster with entries that are most similar with the requirement entry is decided. In order to rank all the entries in this cluster, each entry is then compared with the requirement entry using the similarity matrix. CB-Cloudle presents a final list with the most related results present on the top as shown in Fig. 7 .

VIII. Conclusion and future work

This work presents CB-Cloudle: a new search engine specifically designed for searching cloud services. It consists of a newly designed web interface and uses a new measure to define the similarity matrix. In CB-Cloudle, centroids and the k-means clustering algorithm are adopted to classify the data to different group. This leads to more effective and efficient cloud service search which can potentially contribute to: 1) greater improvement in the effectiveness of search results and 2) significant increase in the efficiency of query performance. By introducing centroid-based approaches and the k-means clustering algorithm, CB-Cloudle significantly reduces the amount of work and the processing time, which is of great importance especially when the data set is dramatically huge.

The work present here is only the first step to demonstrate that the centroid-based method and clustering algorithm can potentially to be used to improve the performance of cloud search engines. In the future, the idea of a distributed cloud

ISBN: 978-988-19252-5-1
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

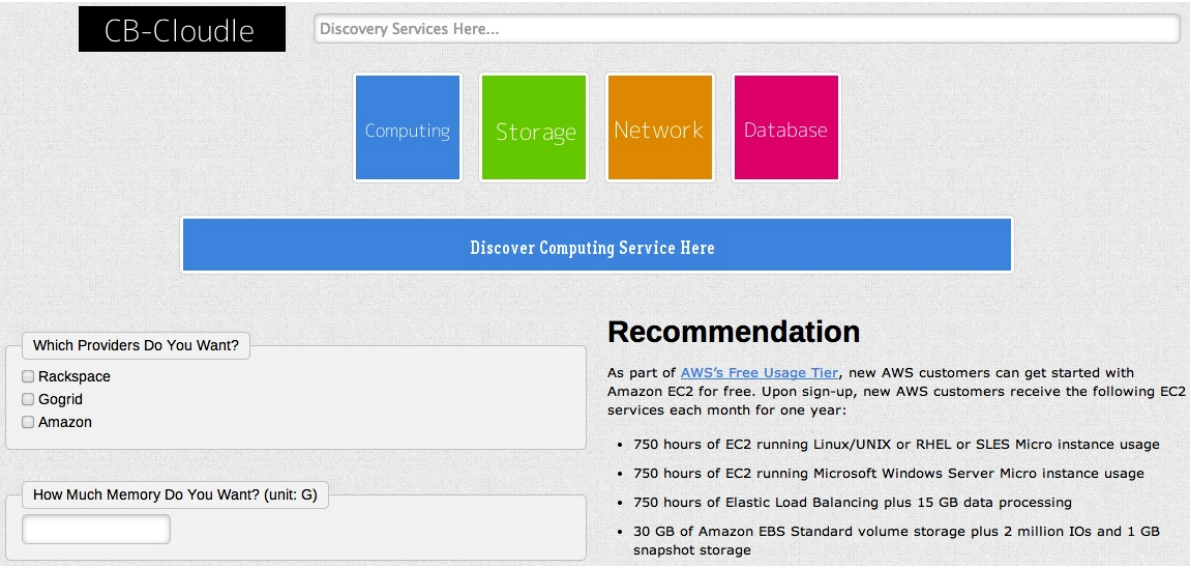
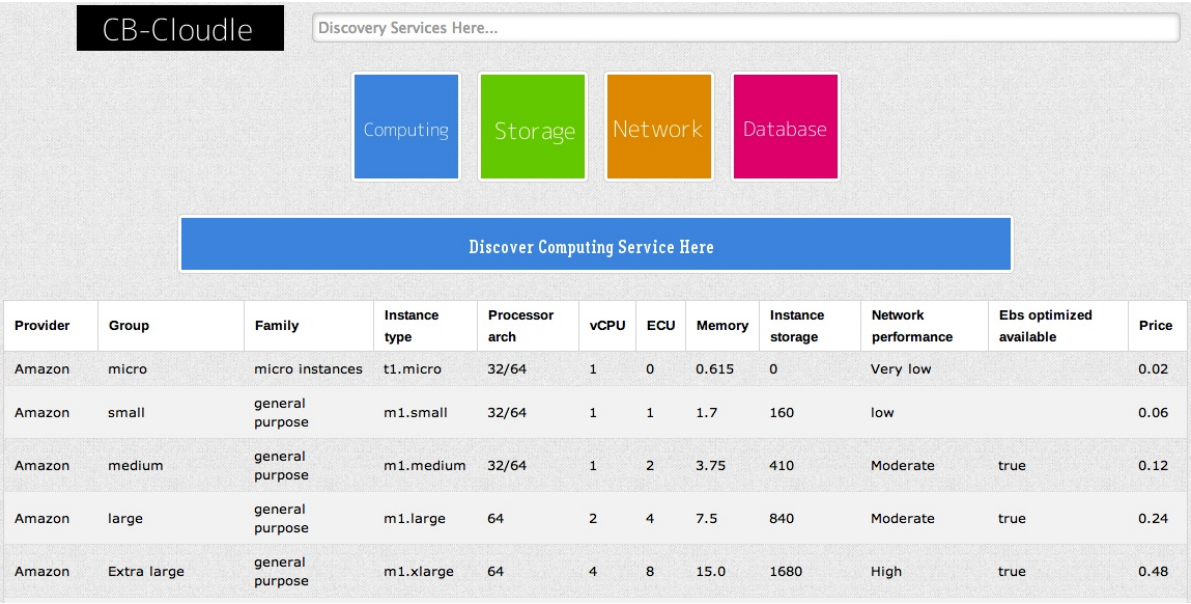


Fig. 5. CB-Cloudle-providing cloud computing service search template



Provider	Group	Family	Instance type	Processor arch	vCPU	ECU	Memory	Instance storage	Network performance	Ebs optimized available	Price
Amazon	micro	micro instances	t1.micro	32/64	1	0	0.615	0	Very low		0.02
Amazon	small	general purpose	m1.small	32/64	1	1	1.7	160	low		0.06
Amazon	medium	general purpose	m1.medium	32/64	1	2	3.75	410	Moderate	true	0.12
Amazon	large	general purpose	m1.large	64	2	4	7.5	840	Moderate	true	0.24
Amazon	Extra large	general purpose	m1.xlarge	64	4	8	15.0	1680	High	true	0.48

Fig. 7. Cloud computing service search results overview

service search engine architecture will be explored. This may include automated data crawlers, a distributed query processor that helps to split the query formula and merge the query results efficiently, and an automated testbed to improve the effectiveness of the search results. All these aim to achieve a higher searching capability of searching more kinds of cloud services and crawling more service providers.

REFERENCES

[1] P. Mell and T. Grance, "The nist definition of cloud computing (draft)," *NIST special publication*, vol. 800, no. 145, p. 7, 2011.
[2] G. Zhang, C. Li, C. Xing, and G. Zhang, "A Semantic++ Social Search Engine Framework in the Cloud," in *SKG '12: Proceedings of the Eighth International Conference on Semantics, Knowledge and Grids*. IEEE Computer Society, Oct. 2012, pp. 277–278.
[3] T. Akiyama, Y. Kawai, Y. Matsui, Y. Kubota, and T. Osaki, "A Proposal for Social Search System Design," *IEEE/IPSJ 11th International Symposium on Applications and the Internet (SAINT)*, pp. 110–117, 2011.
[4] J. Kang and K. M. Sim, "Cloudle: An agent-based cloud search engine that consults a cloud ontology," in *Cloud Computing and Virtualization Conference*, 2010, pp. 312–318.
[5] J. Kang and K. M. Sim, "Ontology and search engine for cloud computing system," in *International Conference on System Science and Engineering (ICSSE)*, 2011, pp. 276–281.
[6] T. Han and K. M. Sim, "An ontology-enhanced cloud service discovery system," in *Proceedings of the International MultiConference of Engineers and Computer Scientists*, vol. 1, 2010, pp. 17–19.
[7] K. M. Sim, "Cloud intelligence: agents within an intercloud," *Awareness Magazine. The official magazine for Future and Emerging Technologies Proactive Initiative, funded by the European Commission under FP7*, 2013.
[8] K. M. Sim, "Agent-based cloud computing," *IEEE Transactions on Services Computing*, vol. 5, no. 4, pp. 564–577, 2012.
[9] J. Kang and K. M. Sim, "A cloud portal with a cloud service search engine," in *International Conference on Information and Intelligent Computing (ICIIC)*, Hong Kong, China, Nov. 2011, pp. 1–8.
[10] J. Kang and K. M. Sim, "Cloudle: An ontology-enhanced cloud service search engine," in *Web Information Systems Engineering–WISE 2010 Workshops*, vol. 6724, 2011, pp. 416–427.
[11] J. Kang and K. M. Sim, "Cloudle: a multi-criteria cloud service search engine," in *IEEE Asia-Pacific Services Computing Conference (APSCC)*, Hangzhou, China, Dec. 2010, pp. 339–346.
[12] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

ISBN: 978-988-19252-5-1
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)