DOI: 10.31534/engmod.2019.2-4.ri.02m
Original scientific paper
Received: 09.12.2019.

Discovery of resources over Cloud using MADM approaches

Mandeep Kaur(1), **Sanjay Kadam**(2)

(1) *Department of Computer Science, Savitribai Phule Pune University, Pune, INDIA*
e-mail: mandeep.gondara@gmail.com

(2) *Centre for Development of Advanced Computing, SPPU, Pune, INDIA*
e-mail: sskadam@cdac.in

SUMMARY

Cloud and P2P computing allows the sharing of resources among its users from diverse geographical locations. The resources are heterogeneous in nature and possess distinct attributes. The majority of the existing techniques rely on the static attributes during resource selection. However, the matching of resources based on static attributes may not be the best for the execution of user jobs. The resources could be very attractive as per the static configuration but might be heavily loaded with other multiple jobs since the working memory and storage space might have already been utilized. Hence, there is a need to consider the dynamic state of the resources in order to locate the most suitable resources for user jobs. In this paper, a 2-phased multi-attribute decision making (MADM) mechanism is proposed to locate the appropriate resources. The first phase explains a mechanism to discover all the matched resources and the second phase then applies the PROMETHEE MADM approach on the set of resources to find the most suitable resource for the user job. The proposed approach enables users to find the most suitable resource for their respective jobs.

KEY WORDS: *resource discovery; MADM, PROMETHEE; P2P; Cloud.*

1. INTRODUCTION

In cloud environment, the resource broker retrieves the resource information and provides matching resource(s) to the user job for execution [1]. However, the matched resource may not be the best one for the user job. Most of the existing matchmaking mechanisms make the decision only based on the static attributes during matching of available resources with user job requirements. It is inevitable to consider dynamic attributes in order to provide 'appropriate' resources [19]. The static attributes reveal the static configuration of the resources such as CPU Architecture, operating system, CPU speed, secondary memory storage, physical memory (RAM), and network bandwidth. The dynamic attributes project the actual availability, the queue length of jobs (job load) on particular resource, free RAM, free secondary storage, available bandwidth, and unutilized CPU of each resource [20].

In this paper, the research work focuses on MADM based approach using P2P formalism that takes care of both static and dynamic parameters in order to locate the suitable resources based on their ranking. The P2P network provides a scalable approach for resource discovery [2], where a peer in P2P environment can forward the request to the neighbouring peers for finding the resources. The requests could be forwarded till the end of time-to-live (TTL). The resources that match the user request are returned as a response to the peer (initiator of the request). The user can submit data or compute-intensive job in P2P environment. For a compute-intensive job, the processor speed, as well as the processor capacity, is more important whereas, for a data-intensive job, the storage space has more weightage.

In our proposed approach, the user job requirement is analysed to get the relative rankings of the resources. Both static and dynamic parameters are considered along with the predefined priorities of the parameters to locate the best resources. The relative importance of one parameter over another is expressed using pairwise comparisons. Finally, the resources are ranked and the best-ranked resource can be selected by the user to execute the job. This paper is organized into six sections. The first section provides an introduction, the second discusses existing work, while the third and fourth discuss our proposed approaches. The last two sections provide insights into the results and conclusion of the proposed research work.

2. RELATED WORK

There have been several attempts made by the researchers towards resource discovery. In [3], the authors proposed a method of Agent-based RD using Bloom filter. The resources information stored in a Bloom filter was sent to the related broker-agent which sent it to the related database. The users raised their resource requests using hash values and forwarded them to the related broker agent. The information needed for search and discovery was stored inside the Bloom filters and this reduced the amount of data transmitted significantly. In [4], a fast RD service and the balanced resource distribution on cloud designed and implemented. Secondly, a comprehensive service distribution approach was designed to analyse the performance of each node and load distribution. Finally, the outcome of the research work indicated that the RD mechanism could improve the efficiency of the resource and the utilization ratio of the service resource. In [5], a P2P-based distributed RD method was proposed based on spatial-awareness of cloud data-centres. This RD mechanism exploited location information of Data Centres and organized them into DHT peers for optimal communication. It allowed QoS-compliant resource provisioning across Cloud Service Providers (CSPs).

In [6], a pastry data hash table (DHT) for resource discovery was proposed. The resource ID key with 160-bit long and the first 128 bits were used for static attributes and rest 32 bits for dynamic attributes. In [7], a super peer model for RD was proposed. The job of super peer nodes was to find matching resources for a given user query and respond back to the peers who initiated the resource request. In [8], the MERCURY RD method was proposed. It created different attribute hubs for a specified resource parameter. Each attribute hub comprised of similar resource attributes. The resource request passed to the hubs containing resources with similar attributes. In [9], the proposed model consisted of 2 major components, namely, agents and aggregators. The agents published the resource-information and broadcasted it to the aggregators. The aggregators collected information from agents and forwarded the requests to other aggregators in a network for resource discovery.

In [10], an RD system was employed, based on a combination of DHTs (distributed hash tables). A dynamic query approach was used to search the required resources with the assistance of DHTs. In [11], a hybrid P2P RD mechanism was proposed with 3 disparate layers; cluster manager layer (CML), resource layer (RL), and super manager layer (SML). The RL contained resources. The SML checked user requests and the nodes in CML layer were responsible for RD. A decentralized RD method was proposed by Barati et al. [12].. The agent-based mechanism supported a semantic-based description of resources. This mechanism allowed the resource agents to interact with neighbour agents in a semantic way to form a resource chain for a specific task request. This model allowed the dynamic discovery of resources. The majority of the existing approaches for RD use only static attributes. There is a need for RD approach that considers both static and dynamic attributes during resource discovery and also ranks the matching resources according to the user requirement. Hence, we are proposing MADM approach for resource discovery that considers both static and dynamic attributes of the resources and also provides ranking to the matching resources.

3. PROPOSED WORK

The proposed RD mechanism using P2P formalism comprises a decentralized network of centralized clusters of resources. A resource can be any peer connected to a super-peer. The super-peers are connected through a virtual overlay network, which is built on top of a physical network. Each super-peer contains information about many other super-peers. Hence, in each cluster, there are peers and all peers are connected to a super-peer as depicted in Figure 1.

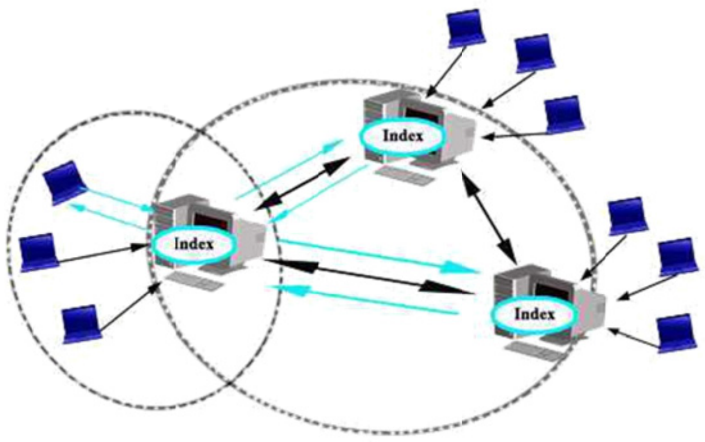

Fig. 1 *P2P formalism*

The proposed discovery mechanism is summarized as follows:

1. The user or peer sends a resource request query by the mentioning required resource attributes to execute the job.
2. The query is forwarded to the super-peer of the cluster, and it forwards the query to the neighbouring super-peers.
3. The neighbouring super-peers respond back to the requesting super-peer by applying the matchmaking between the requested resource parameters and the available ones.
4. The requesting super-peer combines the responses received from different super-peers and then applies MADM approaches to rank the resources.

5. The top-ranked resource is provided to the requesting peer.

3.1 MATCHING OF RESOURCES AT A SUPER-PEER

A super-peer node receives the resource request; the matchmaker algorithm finds resources by matching the attributes of the resources with the required ones. The matchmaker handles exact match queries and range queries. The 'must' criteria comprise static attributes of the resources that are absolutely necessary for the execution of the job. For example, operating system, CPU architecture and minimum working memory are considered as 'must' criteria. A job that requires Linux OS cannot be executed on Windows OS. A job that requires a *64-bit* processor may not be executed on a *32-bit* processor. The first phase of our RD system finds the resources with 'must' criteria. The user can also specify the range for 'must' criteria such as *RAM >= 8 GB*.

Other parameters such as available memory, available network bandwidth, job load on the resource are considered under the 'want' criteria. The 'want' criteria attributes are those that are desirable. The 'want' criteria can be scaled. For example, if a user requires storage space, and there are two resources with *500 GB* and *1 TB* of storage space, the resource with *1 TB* will be assigned more weightage.

An example of job submission with required resource attributes is as follows. *RAM > 512 MB*, *OS = 'LINUX RHEL 7.0'*, *CPU Arch = 'X86-64'*, *Secondary storage > 16GB*, *Bandwidth > 2 MBPS*.

Table 1 *Matched resources retrieved by requesting super-peer from other super-peers*

Responded Super-peers	*Resources*	*CPU*	*RAM*	*Secondary Storage*	*Network Bandwidth*	*CPU utilization*	*Available Storage*	*Available RAM*	*Available Bandwidth*	*Current Load*
Node2	*R1*	*4*	*16*	*1000*	*12*	*50*	*500*	*8*	*4*	*4850*
Node5	*R2*	*3.4*	*8*	*700*	*8*	*40*	*120*	*4*	*2*	*2900*
Node7	*R3*	*3.4*	*16*	*500*	*6*	*20*	*312*	*8*	*3*	*2644*
	R4	*3.9*	*4*	*900*	*8*	*5*	*500*	*3*	*4*	*2000*
	R5	*2.6*	*9*	*600*	*10*	*30*	*300*	*8*	*4*	*2350*

3.2 PROMETHEE APPROACH FOR RESOURCE RANKING

In this section, we propose PROMETHEE-II (Preference Ranking Organization Method for Enrichment of Evaluations) [15] for decision making. This approach enables the decision maker to select the most suitable resources. PROMETHEE-II does not provide any formal way to compute the weights of the attributes [16, 17] and hence, the AHP (Analytical hierarchy process) method is used to get the weights. The positive aspects of both MADM approaches are combined.

Normalize the matrix of matching resources: The first step is to normalize the matrix given in Table 1 of the matching resources to get entries as shown in Table 2. The values of the normalized decision matrix range from 0 to 1. Each column is summed up and each attribute value in that column is divided by the summation value of that column to get its normalized value.

Make the normalized decision matrix unidirectional: The normalized decision matrix columns are then made unidirectional if required. That is, each column in the matrix should represent either minimization or maximization to be performed but not a mixture of both. All the attribute values are converted into unidirectional units as shown in Table 3. In our resource discovery problem, 'CPU utilization' and 'Resource load' at-tribute values are subtracted from 1 to be made unidirectional and to simplify the procedure of pairwise comparison of resources.

Table 2 *Normalized decision matrix*

Resources	*CPU*	*RAM*	*Secondary Storage*	*Network Bandwidth*	*CPU utilization*	*Available Storage*	*Available RAM*	*Available Bandwidth*	*Current Load*
R1	*0.23*	*0.3*	*0.27*	*0.27*	*0.34*	*0.29*	*0.26*	*0.24*	*0.33*
R2	*0.2*	*0.15*	*0.19*	*0.18*	*0.28*	*0.07*	*0.13*	*0.12*	*0.2*
R3	*0.2*	*0.3*	*0.14*	*0.14*	*0.14*	*0.18*	*0.26*	*0.18*	*0.18*
R4	*0.23*	*0.08*	*0.24*	*0.18*	*0.03*	*0.29*	*0.1*	*0.24*	*0.14*
R5	*0.15*	*0.17*	*0.16*	*0.23*	*0.21*	*0.17*	*0.26*	*0.24*	*0.16*

Table 3 *Unidirectional decision matrix*

Resources	*CPU*	*RAM*	*Secondary Storage*	*Network Bandwidth*	*CPU utilization*	*Available Storage*	*Available RAM*	*Available Bandwidth*	*Current Load*
R1	*0.23*	*0.3*	*0.27*	*0.27*	*0.66*	*0.29*	*0.26*	*0.24*	*0.67*
R2	*0.2*	*0.15*	*0.19*	*0.18*	*0.72*	*0.07*	*0.13*	*0.12*	*0.8*
R3	*0.2*	*0.3*	*0.14*	*0.14*	*0.86*	*0.18*	*0.26*	*0.18*	*0.82*
R4	*0.23*	*0.08*	*0.24*	*0.18*	*0.97*	*0.29*	*0.1*	*0.24*	*0.86*
R5	*0.15*	*0.17*	*0.16*	*0.23*	*0.79*	*0.17*	*0.26*	*0.24*	*0.84*

3.2.1 SELECT PREFERNCE FUNCTION

We have selected Type 4 preference function, which is termed as Level-Criterion in the PROMETHEE-II method as shown in Eq. (1):

$$p(x)=\begin{cases}0 & x\le t\\ 1/2 & t<x\le t+s\\ 1 & x>t+s\end{cases} \quad\Rightarrow\quad p(x)=\begin{cases}0 & x\le 0.02\\ 1/2 & 0.02<x\le 0.02+0.12\\ 1 & x>0.02+0.12\end{cases} \tag{1}$$

In Equation (1), $t = 0.02$, and $s = 0.12$, so the specific preference equation is then driven by this equation. The preference of the alternatives a and b is considered as indifferent if the deviation x between $f(a)$ and $f(b)$ is not more than $t = 0.02$, the preference is considered to be weak if the deviation is between *0.02* and *0.14*, and the preference is considered to be strict if the deviation x is greater than *0.14*. Equation (1) is used to calculate all the entries in Table 4 using the alternative-attribute values from Table 3.

Table 4 *Pairwise comparison of resources w.r.t. preference indices*

Pairwise Comparison	*CPU Speed*	*RAM*	*Secondary Storage*	*Network Bandwidth*	*CPU Utilization*	*Available Storage*	*Available RAM*	*Available Bandwidth*	*Current Load*
(R1,R2)	*0.5*	*1*	*0.5*	*0.5*	*0*	*1*	*0.5*	*0.5*	*0*
(R1,R3)	*0.5*	*0*	*0.5*	*0.5*	*0*	*0.5*	*0*	*0.5*	*0*
(R1,R4)	*0*	*1*	*0.5*	*0.5*	*0*	*0*	*1*	*0*	*0*
(R1,R5)	*0.5*	*0.5*	*0.5*	*0.5*	*0*	*0.5*	*0*	*0*	*0*
(R2,R1)	*0*	*0*	*0*	*0*	*0.5*	*0*	*0*	*0*	*0.5*
(R2,R3)	*0*	*0*	*0.5*	*0.5*	*0*	*0*	*0*	*0*	*0*
(R2,R4)	*0*	*0.5*	*0*	*0*	*0*	*0*	*0.5*	*0*	*0*
(R2,R5)	*0.5*	*0*	*0.5*	*0*	*0*	*0*	*0*	*0*	*0*
(R3,R1)	*0*	*0*	*0*	*0*	*1*	*0*	*0*	*0*	*1*
(R3,R2)	*0*	*1*	*0*	*0*	*0.5*	*0.5*	*0.5*	*0.5*	*0*
(R3,R4)	*0*	*1*	*0*	*0*	*0*	*0*	*1*	*0*	*0*
(R3,R5)	*0.5*	*0.5*	*0*	*0*	*0.5*	*0*	*0*	*0*	*0*
(R4,R1)	*0*	*0*	*0*	*0*	*1*	*0*	*0*	*0*	*1*
(R4,R2)	*0.5*	*0*	*0.5*	*0*	*1*	*1*	*0*	*0.5*	*0.5*
(R4,R3)	*0.5*	*0*	*0.5*	*0.5*	*0.5*	*0.5*	*0*	*0.5*	*0.5*
(R4,R5)	*0.5*	*0*	*0.5*	*0*	*1*	*0.5*	*0*	*0*	*0.5*
(R5,R1)	*0*	*0*	*0*	*0*	*0.5*	*0*	*0*	*0*	*1*
(R5,R2)	*0*	*0*	*0*	*0.5*	*0.5*	*0.5*	*0.5*	*0.5*	*0.5*
(R5,R3)	*0*	*0*	*0.5*	*0.5*	*0*	*0*	*0*	*0.5*	*0*
(R5,R4)	*0*	*0.5*	*0*	*0.5*	*0*	*0*	*1*	*0*	*0*

3.2.2 APPLY AHP TO CALCULATE THE ATTRIBUTE WEIGHTS

One could use Saaty's scale [18] to provide the relative importance of one attribute over the other. Table 5 shows the importance of resource attributes w.r.t. compute-intensive job.

Table 5 *Relative importance of attributes for a compute-intensive job*

Attributes	*Importance*
CPU Speed	*8*
CPU utilization	*7*
Available storage	*2*
Available RAM	*6*
Available bandwidth	*3*
Current Load	*9*
RAM	*5*
Secondary Storage	*1*
Network Bandwidth	*4*

For example, if attribute 'A' is 2 times better than attribute 'B', then the relative importance of 'A' over 'B' would be 2/1 and that of 'B' over 'A' would be 1/2. After calculating Eigenvector values of the AHP MADM method, the relative importance of one attribute over another is obtained as shown in Table 6.

Table 6 *Relative importance of attributes based on principal eigenvector*

Resource Attributes	*Eigenvector (Normalized Values)*	*Relative Importance*
CPU Speed	*0.1764*	*The second most important criteria*
RAM	*0.1112*	*The fifth ranked criteria*
Secondary Storage	*0.0214*	*The sixth ranked criteria*
Network bandwidth	*0.0884*	*The seventh ranked criteria*
CPU Utilization	*0.1568*	*The third most important criteria*
Available storage	*0.0448*	*The least important criteria*
Available RAM	*0.1353*	*The fourth ranked criteria*
Available bandwidth	*0.0662*	*The eighth ranked criteria*
Current Load	*0.1996*	*The most important criteria*

3.2.3 AGGREGATED PREFERENCE FUNCTION

The aggregated preference function is applied by using Eq. (2) as shown below. The values obtained are shown in Table 7, after applying the aggregated preference function equation:

$$\pi(a,b) = \sum_{i=1}^{k} \omega_i * p_i(a,b) \tag{2}$$

Table 7 *Preference Index Matrix*

(a_i, a_j)	*R1*	*R2*	*R3*	*R4*	*R5*
R1	*1*	*0.4*	*0.2*	*0.3*	*0.22*
R2	*0.18*	*1*	*0.06*	*0.12*	*0.1*
R3	*0.36*	*0.31*	*1*	*0.24*	*0.22*
R4	*0.36*	*0.43*	*0.38*	*1*	*0.38*
R5	*0.28*	*0.34*	*0.09*	*0.23*	*1*

3.2.4 CALCULATE THE NET OUTRANKING FLOW FOR EACH ALTERNATIVE

The net flow of positive (also termed as leaving flow) and negative flow is defined by Eqs. (3) and (4), which provides net outranking of the alternatives. Table 8 displays both the positive and negative flows together with the net flow and the ranking of the resources. The more the net flow, the better is the resource (alternative).

The positive flow Φ^+ (row sum ratios in Table 8 given by Eq. (3)) depicts the measure of the dominance of a resource (alternative) in a given row over all other resources. The negative flow Φ^- (column sum ratios in Table 8 given by Eq. (4)) depicts the measure of the dominance of a resource (alternative) in a given column over all other resources. A high value of Φ^+ for any given resource or alternative implies that the given resource is better than the other resources. Similarly, a lower value of Φ^- for any given resource or alternative implies that the given resource is better than the other resources.

$$\Phi^+(a)=\frac{1}{n-1}\sum_{x\in A}\pi(a,x) \tag{3}$$

$$\Phi^-(a)=\frac{1}{n-1}\sum_{x\in A}\pi(x,a) \tag{4}$$

Table 8 *Ranking of the resources alternatives*

Resources Alternatives	*Leaving Flow Φ+(a)*	*Entering Flow Φ-(a)*	*Net Flow*	*Ranks*
R1	*0.531*	*0.542*	*0.011*	*4*
R2	*0.364*	*0.622*	*-0.258*	*5*
R3	*0.533*	*0.431*	*0.103*	*2*
R4	*0.636*	*0.475*	*0.161*	*1*
R5	*0.486*	*0.481*	*0.006*	*3*

3.2.5 RESULTS AND DISCUSSIONS

From Table 2 it can be seen that resource *R1* is the best resource as per its static configuration for compute-intensive job in comparison with other resources. However, it ranked 4th based on the scores computed using the PROMETHEE-II based MADM approach, as it is heavily loaded with other user jobs. Therefore, although R1 is the best resource for its static configuration it is not the most suitable one. In the cloud and PP environment, where many user jobs compute for the resources, it is indispensable to make the resources available to the user jobs at the earliest in order to suffice the purpose of optimum utilization of resources. If the user selects the resource with the best configuration but the resource is heavily loaded, then a job has to wait for a longer time in the queue. From Table 8, it can be inferred that resource *R4* is the top-ranked or the most suitable resource for a compute-intensive job. Similarly, *R2* is an attractive resource based on its static attributes but has the lowest rank due to heavy utilization of its CPU, primary memory, and secondary storage space. Hence, our proposed integrated multi-attribute decision making approach provides ranking of the discovered resources and allows the user to select the most suitable resource for the execution of user job based on their preferences. It also demonstrates the relevance of considering the dynamic status of the resource attributes along with the static attributes.

4. CONCLUSION

This paper provides insights into our MADM based proposed resource discovery mechanism in P2P environment. A peer raises a request for the resources to execute a given user job. The super-peer of that node receives the request and forwards it to other super-peers in the P2P-based network. Each super-peer finds the matching resources. After getting responses from all the peers, the requesting super-peer applies PROMETHEE-II based MADM technique to rank all the matched resources and allows the peer to select the best suitable resource for the user application. Using both static and dynamic attributes, it is observed that the proposed resource discovery mechanism selects the most suitable resource.

5. REFERENCES

[1] A.M. Arif, M. Mahmuddin, O. Dakkak, Peer to peer resource discovery mechanisms in grid computing: A critical review, in: *The 4th International Conference on Internet Applications, Protocols and Services*, pp. 295-300, 2015.

[2] A. Hameurlain, D. Cokuslu, K. Erciyes, Resource discovery in grid systems: a survey, *International Journal of Metadata, Semantics and Ontologies*, Vol. 5, No. 3, pp. 251-263, 2010. https://doi.org/10.1504/IJMSO.2010.034048

[3] R. Nikbazm, M. Ahmadi, Agent-based resource discovery in cloud computing using bloom lters, in: *2014 4th International Conference on Computer and Knowledge Engineering* (ICCKE), pp. 352-357, 2014. https://doi.org/10.1109/ICCKE.2014.6993399

[4] H.W. Zhao, Research on a resource discovery mechanism in cloud computing environment based on analysis of scientific materials, in: *Advanced Research on Material Engineering, Chemistry, Bioinformatics,* Vol. 282 of Advanced Materials Research, pp. 433-436, 2011. https://doi.org/10.4028/www.scientific.net/AMR.282-283.433

[5] L. Kappor, S. Bawa, A. Gupta, Peer clouds: A p2p-based resource discovery mechanism for the intercloud, *International Journal of Next-Generation Computing*, Vol. 6, No. 3, pp. 153-164, 2015.

[6] A.S. Cheema, M. Muhammad, I. Gupta., Peer-to-peer discovery of computational resources for grid applications, *The 6th IEEE and ACM International Workshop on Grid Computing*, pp. 179-185, 2005. https://doi.org/10.1109/GRID.2005.1542740

[7] B. Yang, H. Garcia-Molina, Designing a super-peer network, in: *19th IEEE International Conference on Data Engineering*, 2003.

[8] A. Bharambe, M. Agarwal, S. Seshan, Mercury: Supporting scalable multi-attribute range queries, in: *Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications* (SIGCOMM04), pp. 353-366, 2004. https://doi.org/10.1145/1015467.1015507

[9] D. Puppin, S. Moncelli, R. Baraglia, N. Tonelotto, F. Silvestri, A grid information service based on peer-to-peer, in: *11th Euro-Par Conference*, pp. 454-464, 2005.

https://doi.org/10.1007/11549468_52

[10] H. Papadakis, P. Trun o, D. Talia, P. Fragopoulou, Design and implementation of a hybrid p2p-based grid resource discovery system, *Making Grids Work*, pp. 89-101, 2008.

https://doi.org/10.1007/978-0-387-78448-9_7

[11] M.B. Bashir, M. Lati, A. Ahmed, Y. Coulibaly, A.H. Abdullah, A. Yousif, A hybrid resource discovery model for grid computing, *International Journal of Grid Computing & Applications*, Vol. 2, No. 3, p.p. 1-12, 2011. https://doi.org/10.5121/ijgca.2011.2301

[12] M. Barati, R. Alizadeh, A new method for resource discovery in a grid resource service, *International Journal of Grid and Distributed Computing*, Vol. 7, No. 2, pp. 21-32, 2014.

https://doi.org/10.14257/ijgdc.2014.7.2.03

[13] P.C. Fishburn, Additive Utilities with Incomplete Product Set: Applications to Priorities and Assignments, *Operations Research Society of America (ORSA) Publication*, Baltimore, MD, Vol. 15, No. 3, pp. 537-542, 1967. https://doi.org/10.1287/opre.15.3.537

[14] L. Wang, J. Tao, H. Marten, A. Streit, S. Khan, J. Kolodziej, D. Chen, MapReduce across distributed clusters for data-intensive applications., *2012 Proceedings of 26th IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 2004-2015, 2012. https://doi.org/10.1109/IPDPSW.2012.249

[15] J.P. Brans, P. Vincke, A preference ranking organisation method: (the PROMETHEE method for multiple criteria decision-making), *Management Science*, Vol. 31, No. 6, pp. 647-656, 1985. https://doi.org/10.1287/mnsc.31.6.647

[16] L. Turcksin, A. Bernardini, C. Macharis, A combined AHP-PROMETHEE approach for selecting the most appropriate policy scenario to stimulate a clean vehicle fleet, *Procedia Social and Behavioral Sciences*, Vol. 20, pp.954-965, 2011.

https://doi.org/10.1016/j.sbspro.2011.08.104

[17] D. Bogdanović, D. Nikolić, I. Ilić, Mining method selection by integrated AHP and PROMETHEE method, *Annals of the Brazilian Academy of Sciences*, Vol. 84, No. 1, pp. 219-233, 2012. https://doi.org/10.1590/S0001-37652012005000013

[18] T. Saaty, Decision making with the analytic hierarchy process, *International Journal of Services Sciences*, Vol. 1, No. 1, pp. 83-98, 2008.

https://doi.org/10.1504/IJSSCI.2008.017590

[19] Sandhya, Rakesh Garg and Ramesh Kumar, Computational MADM evaluation and ranking of cloud service providers using distance-based approach, *International Journal of Information and Decision Sciences*, Vol. 10, No. 3, pp. 222–234, 2018.

https://doi.org/10.1504/IJIDS.2018.093930

[20] Chao Li, Xiaofen Zhang, Hongfeng Geng and Jinghai Yan, Robustness Analysis Model for MADM Methods Based on Cloud Model, In proceedings of International Congress of Information and Communication Technology (ICICT 2017), Vol. 107, pp. 84-90, 2017.

https://doi.org/10.1016/j.procs.2017.03.061