A solution based on prediction for cloud resources allocation

Djelid raouf

Eloued university.

Constantine 2 university

Algeria

Djelid-raouf@univ-eloued.dz

Djelid.raouf@gmail.com

Abstract. The phenomenon of the clouds makes it possible to have a dynamic resource management. The effectiveness of this management becomes an important issue with the increasing size of the number of resources and the number of users. Because of a non-negligible initialization cost, a precise prediction needs must be made. In this article, we offer an approach to the prediction of the Identification based on the use of resources in the history of a platform. We represent the algorithm of resources allocation witch uses this method to assist the system to use resource allocation decisions as well as results of utilization expertise of cloud resources historical load.

Key words: cloud computing, resources allocation, prediction algorithm.

1. Introduction

The evolution of software services to the clouds has taken a new growth with the effective use of virtualized resources. These platforms allow the dynamic addition or withdrawal of such resources during the execution of an application hosted on a cloud. By going further, it is possible to dynamically manage these resources at the resource provider or the customer itself by way of the provider API. When cloud customer optimizes its resource management through elastic management, it saves expenditures.

The main problem of dynamic resources management is that new resources are not obtained instantly and the start-up latency cannot be neglected. In order to solve this problem, the strategies currently in place use predictive approaches hoping provision of the resources in advance, reducing this latency. The idea of self-similarity was used in other areas such as web traffic [3]. From these ideas, a new customer resources management strategy at the customer level can be developed that identify reasons of use of resources that took place in the past and have an important degree of similarity with current motive.

The prediction system that comes from this work uses the historical of resources used by a client and tries to guess intensively the short-term load. This is not an entire system of dynamic resources management as there are other elements to be taken into account. In our work, we focus only on the prediction of resource use. The impact of the other factors on residential

decisions is an important research subject that is exceeding the boundaries of this work. the main contribution of this paper is the elaboration of an algorithm of resources prediction based on the past use of resources. This allocation is realized by modifying the algorithm proposed in [1].

The rest of this article is organized as follows. The following section presents an existing horizon of the related works. Then section 3 lists the problem definition and shows our algorithm and its principal concepts. Finally, a conclusion and a description of future work.

2. Related works

At present, two main approaches are needed to implement self-resizing decisions at the client level based on past use of resources witch is a mechanism that permits the adjustment of resources allocation by adding additional CPUs, network interfaces, or memory. The first approach takes as input the past use of the servers and deduces a mathematical model that models it. The next value of the source request is obtained by instantiating the model at the next time. The second approach is reactive, based on the current server load and self-resizing rules that are implemented by a human operator (typically aCloud client). This approach is often referred to as the elastic rule approach or the SLA Service Level Agreement approach [5].

In [8], a comparative description of three self-resizing algorithms is given: the self-regression 1 (AR1), the linear regression and the RightScale algorithm. The two first algorithms are belonging to the first category of self-resizing algorithms. They consist to identify a recurrent sequence (or respectively a polynomial) with a sliding window and using it to predict the following value. the RightScale algorithm belongs to the second category. Its approach is to use a democratic vote system in which each virtual machine owned by the cloud vote either to increase or reduce depending on its load. The majority decides the resizing decision for the comprehensive platform.

A more complex form of dynamic resource management with rules, Service Level Agreement (SLA) can be described using the elastic rules that dictate which part of the client must evolve, in which direction and for which volume. In [5], one can find such an example with threshold rules. This is performed by extending OVF (Open Virtualization Format), an open and interoperable format independent of the platforms or suppliers that is used to describe virtual applications (VAs). This approach is reactive. The extensibility rules have the interest of combining the good performance of the threshold algorithms with the adaptation. They have been widely used in the currently available clouds.

Studies around the prediction of load and its modeling are not new and there have been many recent work in particular of the calculation grids. In [6], a fine study of synthetic loads several production grids and research. Performance metrics useful for the evaluation of grids environments are described. This article describes the diagrams to submit these grid environments and presents some of the current approaches to modeling them.

A non-linear model of grid load prediction is given in [4]. The authors offer a prediction model as a series of known functional components generally token from a sigmoid function class. The experimental prediction error is less than 14% with an average of 7.5%.

In [7], one can find a real-time resource management system for on line multiplayer game based on predictive use model.. The authors offer a predictive model based on a neurons network of. Through their experiences, the neuron networks proves its best Precision compared to other prediction methods tested: average, mobile average, last value and exponential smoothing. The error of prediction obtained during the experiments has a maximum male value of 33% and a minimum value of 4.94%.

One of the defects of these reactive approaches is the fact that they are insensitive to the overall form of the load they are predicting since they do not consider recent states of load for their results. On the contrary, predictive approaches consider trends, but the above presented approaches, which use mathematics models, do not consider self-similarities which have not repetitive periodic behavior. This is the reason that has motivated our work.

3. Problem definition

The migration of virtual instance to another physical hypervisor is needed when there is a non equilibrate load between hypervisors, this migration can be done randomly to any hypervisor of the infrastructure which is not recommended because it may cause inefficiencies. Which is suggested is resource migration to the appropriate hypervisor. The following graphics shows inefficiency in the use of two hypervisors. At time t4 to t7 occur inefficiencies on hypervisor 1 in with the need of physical resource is 100% or more however the hypervisor 2 is only used at 40 and 50% respectively at the same time. [1]

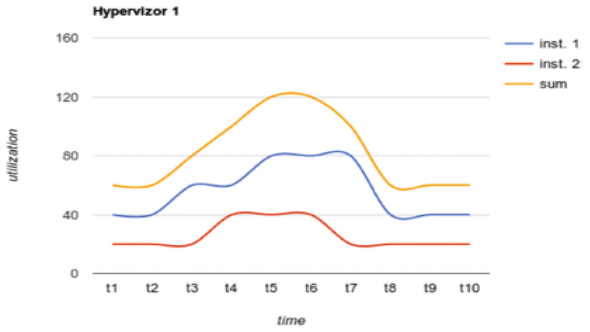

Fig. 2. Utilization of hypervizor 1

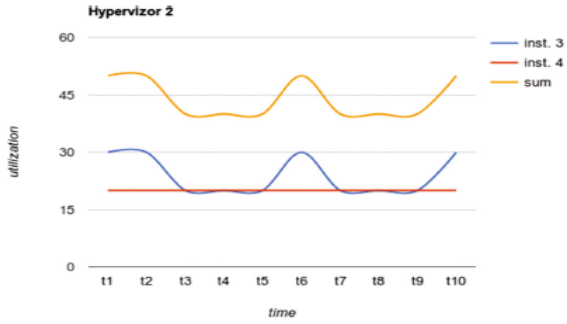

Fig. 3. Utilization of hypervizor 2

Table 1. Table of load (in % of hypervisor performance)

Utilization (%)	t1	t2	t3	t4	t5	t6	t7	t8	t9	t10
Instance 1	40	40	60	60	80	80	80	40	40	40
Instance 2	20	20	20	40	40	40	20	20	20	20
Sum (hypervizor 1)	**60**	**60**	**80**	**100**	**120**	**120**	**100**	**60**	**60**	**60**
Instance 3	30	30	20	20	20	30	20	20	20	30
Instance 4	20	20	20	20	20	20	20	20	20	20
Sum (hypervizor 2)	**50**	**50**	**40**	**40**	**40**	**50**	**40**	**40**	**40**	**50**

A solution to eliminate this inefficiency can be done by moving instance 4 to hypervisor 1 and instance 2 to hypervisor 2. Which will reduces the inefficiency only at time t5 to t7 but only at the level of allocation of 100% and that is more acceptable.

4. Solution design

The algorithm realizing such migration can be designed as follow:

If we consider a visualization interval of N times so we declare an array T of t1 to tN.

T[N] :integer.

The number of resources instances is K we designate a resource by Ri such as i is varying from 1 to K.

So let' s calculate the average of the resources load use for each hypervisor as follow,

$$AVG = \sum_{i=1}^{n} Li \ /N$$

Where Li is the load of hypervisor at time ti.

Time	t1	t2	…..	……	…..	tN
Resource 1	L1R1	L2R1				LnR1
Resource 2	L1R2	L2R2				LnR2
Hypervisor A	L1	L2				Ln
TL A	Good	Good	Bad			Good
Resource 3	L1R3	L2R3				LnR3
Resource 4	L1R4	L2R4				LnR4
Hypervisor B	L1	L2				Ln
TL B	Bad	Good	Bad			Bad

L1R1 is the load 1 of resource 1.

Now we calculate for each time if the load is over or under the average.

1: TL[n]: array of real.

/* Phase 1

2: For i: =1 **to** N **do**

3: {

4: TL[i]: =Li-AVG

5: **If** TL[i]>0 **then**

6: The load of hypervisor at time ti is BAD

7: **Else**

8: The load of hypervisor at time ti is GOOD

9 : }

So the hypervisor A is overloaded than the hypervisor B at time ti if TL[i] = bad at hypervisor A and TL[i]= good at hypervisor B.

In this case we move the resource i from hypervisor A to hypervosor B as follow

/* Phase 2

10: For i:=1 **to** n **do**

11: **IF** TLA[i] = "BAD" **then**

12: **IF** LiR2 > LiR1 **then**

13: R2TM:=" yes" /* instance 2 to move = yes

14: **Else**

15: **IF** LiR2 < LiR1

16: R1TM:=" yes"

17: **Else**

18: R2TM:=" yes" ;

19: R1TM:=" yes" ;

countbad:=countbad+1;

20: **ELSE**

21: **IF** LiR3 < LiR4 **then**

22: R3TM:=" yes"

23: **Else**

24: **IF** LiR3 > LiR4

15: R4TM:=" yes"

26: **Else**

27: R3TM:=" yes"

28: R4TM:=" yes"

Countgood:=countgood+1

IF countbad >= (N/2)

The resource with maximum number of RiTM=" yes" will be migrated

ELSE

IF countgood >= (N/2)

The resource with minimum number of RiTM=" yes" will be migrated

ELSE

The resource with maximum number of RiTM=" yes" will be migrated

In the case of the equality between the load values of the two resources instances at time ti we mark the two instance as to be migrated and the final decision of what ressource to migrate is differed to the end when the total migration status is calculated as:

when the hypervisor load status is bad according to the law proposed above showing that it is over the global load average, so we migrate the resource of maximum high load values at all ti time intervals to minimize the load. In contrary, if the load of the two resources is equal at the most ti time intervals when the hypervisor load status at these times is good, we migrate the resource of minimum hight load values at all ti time intervals. In remaining ordinary cases when there are random resources load values we migrate the most loaded when the load status is bad lowest loaded if the load is good.

The algorithme has two phases.

The phase 1 is to determinate the load status (good or bad) at time ti for each hypervisor. While phase 2 allow to know witch resource instance from the overloaded hypervisor is to move.

if we execute the algorithm on table 1 we get the following.

Utilisation (%)	t1	t2	t3	t4	t5	t6	t7	t8	t9	t10	
Instance 1	30	30	40	60	80	80	80	40	40	40	**hypervisor 1**
R1TM				yes	yes	yes	yes				
Instance 2	30	30	40	40	40	40	20	20	20	20	
R2TM	yes	yes	yes					yes	yes	yes	
Sum (hypervisor 1)	60	60	80	100	120	120	100	60	60	60	AVG = 82
TL 1 (Sum-AVG)	-22	-22	-2	18	38	38	18	-22	-22	-22	
TL 1 (good or bad)	good	good	good	bad	bad	bad	bad	good	good	good	
Instance 3	30	30	20	20	20	30	20	20	20	30	**hypervisor 2**
R3TM	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	
Instance 4	20	20	20	20	20	20	20	20	20	20	
R4TM			yes	yes	yes		yes	yes	yes		
Sum (hypervisor 2)	50	50	40	40	40	50	40	40	40	50	AVG = 44
TL 2 (Sum-AVG)	6	6	-4	-4	-4	6	-4	-4	-4	6	
TL 2 (good or bad)	bad	bad	good	good	good	bad	good	good	good	bad	

The ressource 2 from the hypervisor 1 is to migrate and the resource 4 from the hypervisor 2 is to migrate.

The load before the execution of the algorithme was :

Sum (hypervisor 1)	60	60	80	100	120	120	100	60	60	60	AVG = 82
Sum (hypervisor 2)	50	50	40	40	40	50	40	40	40	50	AVG = 44

After execution of the algorithme.

Sum (hypervisor 1)	60	60	80	80	100	100	100	60	60	60	AVG = 76
Sum (hypervisor 2)	50	50	40	60	60	70	40	40	40	50	AVG = 50

So the load average between the two hypervisor is balanced and the overload occured at time t4 is eliminated, also the overload still at time t5 to t7 but with butter values.

The table bellow show another example

Utilisation (%)	t1	t2	t3	t4	t5	t6	t7	t8	t9	t10	
Instance 1	80	40	50	20	90	100	110	60	90	40	**hypervisor 1**
R1TM		yes	yes	yes	yes	yes	yes		yes		
Instance 2	20	80	70	50	60	40	20	80	40	90	
R2TM	yes							yes		yes	
Sum (hypervisor 1)	100	120	120	70	150	140	130	140	130	130	AVG = 123
TL 1 (Sum-AVG)	-23	-3	-3	-53	27	17	7	17	7	7	
TL 1 (good or bad)	good	good	good	good	bad	bad	bad	bad	bad	bad	
Instance 3	30	50	60	10	44	80	10	60	30	40	**hypervisor 2**
R3TM			yes	yes		yes	yes				
Instance 4	20	60	30	50	60	80	10	30	20	30	
R4TM	yes	yes			yes	yes	yes	yes	yes	yes	
Sum (hypervisor 2)	50	110	90	60	104	160	20	70	50	70	AVG =78,4
TL 2 (Sum-AVG)	-28,4	31,6	90	-18,4	60	82	-58	-8,4	-28,4	-8,4	
TL 2 (good or bad)	good	bad	bad	good	bad	bad	good	good	good	good	

The load comparaison before and after the algorithm is executed is as follow :

Sum (hypervisor 1)	100	120	120	70	150	140	130	140	130	130	AVG = 123
Sum (hypervisor 2)	50	110	90	60	104	160	20	70	50	70	AVG =78,4

Sum (hypervisor 1)	40	140	100	100	120	120	30	110	60	120	AVG = 94
Sum (hypervisor 2)	110	90	110	30	134	180	120	70	120	80	AVG =104,4

The overload occured seven times before and four times after the execution of the algorithm. And the global load average of the two hypervisors is balanced.

The algorithm is operational only in the case of two resource instance hosted by hypervisor. In the case of more resources instances we need only to add as many as conditions to the algorithm inorder to migrate the lowest load value resource from the hypevisor of load GOOD and the highest load value resource from the hypervisor of load BAD at time ti.

The algorithm is based on prediction methode , it took as entry the past monitoring use of resources. We will get better results if we execute the algorithm on the future ressource need table, if none, so the result of the algorithm in this case is witch resource to allocate to avoid overload before it occurs.

5. Future works

A future prediction algorithm of resource allocation should be based on new prediction approaches listed in [2] and should take in consideration the following challenges.

- Considering all dimensions of the application to making the appropriate learning model to enhance the prediction accuracy.
- Based on resource adaptation method to improve the dynamic characteristic of cloud application.
- It should not be support assumption of workload behavior. Because, this behavior could be violate the entire resource management system.
- Support interpretable behavior by the cloud resource manager.
- Combining reactive approaches and proactive approaches to improve the accuracy.
- Considering correlation among resources to extract important features from dataset.[2]

6. Conclusion

Cloud computing is designed in such a way, so as to provide various services to the external users. But without knowing about the future needs of the clients it is tedious to provide the resources to them. This paper presents dynamic resource allocation in cloud computing using load prediction algorithm based on the previous resources use monitoring.

The testing but also the monitoring and operation of this algorithm has been done and has shown a good results.[9].

Bibliography

1. Lubos Mercl, An Algorithm for Migration and Resource Planning in Cloud Technologies. Advances on P2P, Parallel, Grid, Cloud and Internet Computing proceeding of The 12th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC-2017). Springer International Publishing AG 2018.

2. K Dinesh Kumar and E Umamaheswari, Resource Provisioning in Cloud Computing Using Prediction Models: A Survey. International Journal of Pure and Applied Mathematics Volume 119 No. 9 2018, 333-342

3. Mark Crovella and Azer Bestavros. Explaining World Wide Web Traffic Self-Similarity. Technical Report TR-95-015, Boston, MA, USA, 1995.

4. Nikolaos Doulamis, Anastasios Doulamis, Antonios Litke, Athanasios Panagakis, Theodora Varvarigou, and Emmanuel Varvarigos. Adjusted fair scheduling and non-linear workload prediction for QoS guarantees in grid computing. Computer Communications, 30(3) :499 – 515, 2007. Special Issue : Emerging Middleware for Next Generation Networks.

5. Fermín Galán, Americo Sampaio, Luis Rodero-Merino, Irit Loy, Victor Gil, and Luis M. Vaquero. Service specification in cloud environments based on extensions to open standards. In COMSWARE '09 : Proceedings of the Fourth International ICST Conference on COMmunication System softWAre and middlewaRE, pages 1–12, New York, NY, USA, 2009. ACM.

6. AlexandruIosup,DickH.J.Epema,CarstenFranke,AlexanderPapaspyrou,LarsSchley,BaiyiSong, and Ramin Yahyapour. On grid performance evaluation using synthetic workloads. In JSSPP' 06 : Proceedings of the 12th international conference on Job scheduling strategies for parallel processing, pages 232– 255, Berlin, Heidelberg, 2007. Springer-Verlag.

7. Radu Prodan and Vlad Nae. Prediction-based real-time resource provisioning for massively multiplayer online games. Future Generation Computer Systems, 25(7) :785 – 793, 2009.

8. Jonathan Kupferman, Jeff Silverman, Patricio Jara, and Jeff Browne. Scaling Into The Cloud. http: //cs.ucsb.edu/~jkupferman/docs/ScalingIntoTheClouds.pdf, 2009.

9.The official site of CloudSim http://www.cloudsimapp.com.