Error-Tolerant Resource Allocation And Payment Minimization For Cloud System

R.S.Prabakaran, T.R.Abinaya, T.Aarthi

ABSTRACT: Virtual machine (VM) technology being greater and fully developed, compute resources in cloud systems can be partitioned in fine granularity and allocated on demand, which contributes three technologies such as, Formulating a deadline-driven resource allocation problem based on the cloud environment facilitated with VM resource isolation technology, and also to minimize users' payment. Analyzing the upper bound of task execution length based on the possibly inaccurate workload prediction, it further proposed an error-tolerant method to guarantee task's completion within its deadline. Validating its effectiveness over a real VM-facilitated cluster environment under different levels of competition.

Keywords: VM multiplexing, Resource allocation, Error tolerance, Payment minimization.

I. INTRODUCTION

Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (software, data storage, network, etc.) that can be rapidly provisioned and released with minimal management effort or service provider interaction. Cloud computing [1], [14] is a comprehensive solution that delivers IT as a service. It is an Internet-based computing solution where shared resources are provided like electricity distributed on the electrical grid. Computers in the cloud are configured to work together and the various applications use the collective computing power as if they are running on a single system. The characteristics of cloud computing are: user friendliness, virtualization, Internet centric, variety of resources, automatic adaptation, scalability, resource optimization, pay-per-use, service SLAs (Service-Level Agreements)[15] and infrastructure SLAs.

- **Reduced cost:** Cloud computing can reduce both capital expense (CapEx) and operating expense (OpEx) costs because resources are only acquired when needed and are only paid for when used.
- **Refined usage of personnel:** Using cloud computing frees valuable personnel allowing them to focus on delivering value rather than maintaining hardware and software.
- **Robust scalability:** Cloud computing allows for immediate scaling, either up or down, at any time without long-term commitment.

- *R.S.Prabakaran, PG Scholar (CSE), Sri Ramakrishna Engineering College, Coimbatore*
- *T.R.Abinaya, PG Scholar (CSE), Sri Ramakrishna Engineering College, Coimbatore*
- *T.Aarthi, PG Scholar (CSE), Sri Ramakrishna Engineering College, Coimbatore*

II. PROPOSED SYSTEM

This scheme proposed a novel resource allocation algorithm for cloud system that supports VM-multiplexing technology, aiming to minimize user's payment of task within its deadline. It can prove that the output of the algorithm is optimal based on the KKT condition[2], which means any other solutions would definitely cause larger payment cost. In addition, this paper analyze the approximation ratio for the expanded execution time generated by the algorithm to the user-expected deadline, under the possibly inaccurate task property prediction. When the resources provisioned are relatively sufficient, It can guarantee task's execution time[11] always within its deadline even under the wrong prediction about task's workload characteristic.

1) It formulate a deadline-driven resource allocation problem based on the cloud environment facilitated with VM resource isolation technology, and also proposed a novel solution with polynomial time, which could minimize users' payment in terms of their expected deadlines.
2) By analyzing the upper bound of task execution length based on the possibly inaccurate workload prediction, it further proposed an error-tolerant method to guarantee task's completion within its deadline.
3) It validate its effectiveness over a real VM-facilitated cluster environment under different levels of competition.

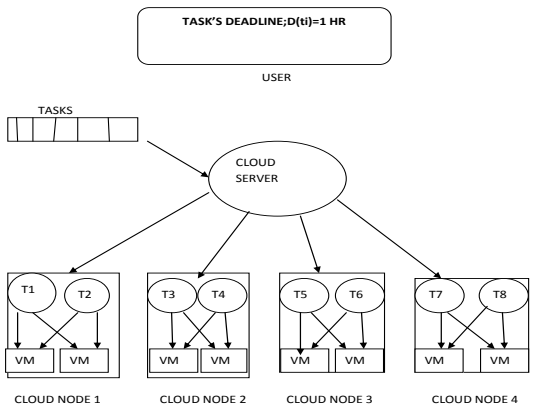

Fig. 2 Resource allocation in cloud system

1. ADVANTAGES OF PROPOSED SYSTEM

- Based on the elastic resource usage model, this paper aims to design a resource allocation algorithm with high prediction error tolerance ability, also minimizing users' payment subject to their expected deadlines.
- The idle physical resources can be arbitrarily partitioned and allocated to new tasks; the VM-based divisible resource allocation could be very flexible.

III. SYSTEM IMPLEMENTATION

3.1 Optimal Resource Allocation

Based on the convex optimization theory[2],[3], the Lagrangian function of the problem could be formulated as (1), where λ and $\mu_1, \mu_2, \ldots, \mu_R$ are corresponding Lagrangian multipliers. Note that θ is a constant defined in (1) and r is the abbreviation of r (t_i) as stated above

$$F1(R) = \frac{1}{R}(\sum_{k=1}^{R} b_k r_k)(\theta \sum_{K=1}^{R} \frac{l_k}{r_k}) + \lambda(\theta \sum_{K=1}^{R} \frac{l_k}{r_k} - D) + \sum_{K=1}^{R} \mu_k (r_k - a_k)$$

(1) Where,

R=Execution Dimension,

B_k=Price Vector,

R_k=Resource Vector,

L_k=Workload Vector,

D=Deadline,

A_k=Available Vector

According to Karush-Kuhn-Tucker conditions (i.e., the necessary and sufficient condition of the optimization).

$$\lambda \geq 0, \mu_K \geq 0, K = 1, 2 \ldots \mathrm{R}$$

$$\sum_{I=1}^{R} \theta \frac{l_I}{r_I} \leq D$$

$$\lambda(\sum_{I=1}^{R} \theta \frac{l_I}{r_I} - D) = 0$$

$$r_K \leq a_K(P_S), K = 1, 2 \ldots . R; S = 1, 2, \ldots . \mathrm{n}$$

$$\mu_K(\mathrm{r}_K - a_K(P_S)) = 0, K = 1, 2 \ldots . R; S = 1, 2 \ldots \mathrm{n}$$

$$\frac{dF1}{dr_K} = \frac{1}{R}((\sum_{i=1}^{R} b_i r_i).\frac{-l_k}{r_k^2} + b_k.\sum_{i=1}^{R} \frac{l_i}{r_i} + -\lambda \frac{l_k}{r_k^2} + \mu_k) = 0$$

….2)

K=1,2….R

Where,

R=Resource,

B_k=Price Vector,

R_k=Resource Vector,

Lk=Workload Vector,

D=Deadline,

A_k=Available Vector

Optimal allocation algorithm

- Input: D(t_i); Output:execution node p_s , r*(t_i)
- Γ =Π, C=D (t_i), r*=ϕ (empty set);
- Repeat
- $r_Γ$* (t_i , p_s) = CO-STEP (Γ,c);
- on Γ*
- Ω = d_k/ d_k Є Γ & $r_k^{(*)}$ (t_i , p_s) > $a_k(p_s)$};
- Γ =Γ\ Ω/*Γ take away Ω* /
- C= C –θ $\sum_{d_k \in \Omega} \frac{l_k}{a_k}$ /* Update C* /
- r* (t_i , p_s) = r* (t_i , p_s) U ($r_k^{(*)}$ = $a_k(p_s)$|d_k Є Ω& $a_k(p_s)$
- is d_k 's upper bound};
- until (Ω =ϕ);
- `r* (t_i , p_s) = r* (t_i , p_s) U $r_Γ$* (t_i , p_s)
- end for
- Select the smallest p(ti) by traversing the candidate solution set;
- Output the selected node ps and resource allocation r*(ti,ps);

Based on the Algorithm 1, it is obvious that the local optimal resource allocation for t_i to be executed on a specified node p_s is the most crucial part. In fact, the final outputted resource allocation solution of the whole algorithm will be globally optimal around the whole system as long as each local process on a specified node (line 2-10) can be proved as optimal resource allocation. Consequently, this will intensively discuss the local divisible-resource allocation by specifying a particular execution node, in the following text. Although Algorithm 1 is proved optimal for minimizing the payment cost within user-defined deadline of task still may not be guaranteed due to two factors, either bounded available resources or inaccurate workload vector information about the task. IT proposed the following lemma, which provides a necessary and sufficient condition of guaranteeing[8] the task's deadline given accurate prediction and relatively sufficient resources.

3.2 Optimality Analysis with Inaccurate Information

$$r^{*}E(ti) = r^{*}E(ti)....(3)$$

$$r^{*}E(ti) \neq r^{*}E(ti).....(4)$$

The first situation indicates that in terms of the skewed estimation of workload ratios, all the resource shares calculated by the initial CO-STEP in Algorithm 1 are always no greater than the corresponding capacities. That is, it is equal to the situation with the assumption that Inequality holds:

$$\boldsymbol{r}_{E}^{(*)}(t_i) \preceq \boldsymbol{a}(p_s). \quad .....(5)$$

In contrast, the second one means that the initial CO-STEP cannot fulfill the above condition, and the optimal allocation cannot be found unless a few more adjustment steps (line 5-8 of Algorithm 1).

IV.RESULTS AND DISCUSSION

4.1 RESULTS: OAA and DER

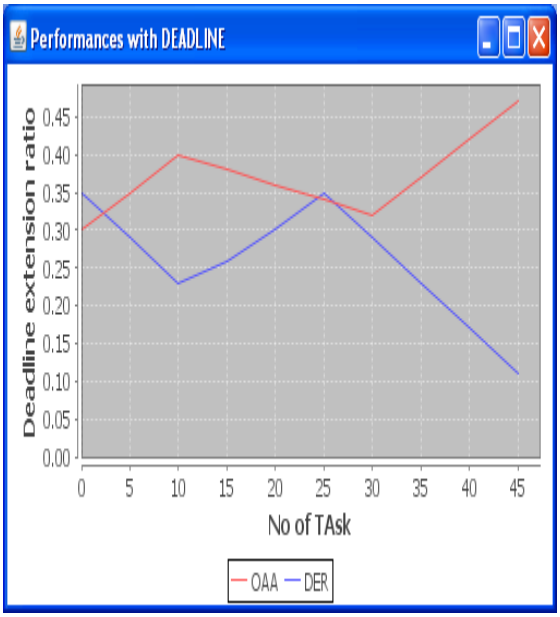

Fig 5.1 Performance With Deadline D'=D

DISCUSSIONS:
In this graph to measure the performance of the OAA and DER(Deadline Extension Ratio). The number of input tasks are shown in the X axis and the deadline extension ratio results are measured in Y axis. The experimental results are shown by using the original deadline D (i.e., D'=D) in the algorithm. From Fig. 1, the tasks' execution times cannot be always guaranteed to be executed within their deadlines in the worst case, no matter how many tasks (1-45) are submitted. Specifically, even though the system availability is relatively high (e.g., there are only several tasks submitted), the average value of deadline extension ratio is nearly to 0.15.

4.2 Performance With Deadline D= α*D

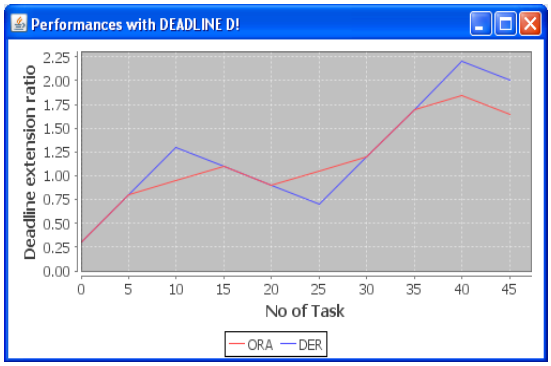

Fig 5.2 Performance With Deadline D= α*D

Discussions:
Deadline extension ratio when the deadline D' is set to a stricter deadline (α*D). When the number (denoted by m) of tasks submitted scales up to 45, all tasks' execution times can be kept nearly to about only 2.00 times as high as their preset deadlines (D) at the worst situation (i.e., the highest level shown in the figure). With further increasing number of the submitted tasks, tasks' execution times cannot be always guaranteed because of the limited resource capacities (or the higher level of competitions on resources), but the mean level is still kept remarkably lower than 2.25, which means that most of the tasks can still meet the QoS (i.e., large majority can be finished before deadlines).

4.3 Performance With Lower Bound

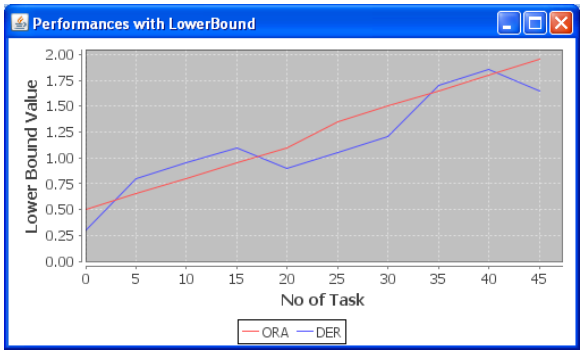

Fig 5.3 Performance With Lower Bound

Disscussions:
Optimal algorithm is based on the inaccuracy of the workload predicted, according to the analysis. From this figure can clearly observe that the prediction method we used can make sure that the lower bound of the workload predicted (i.e., α value will be set close 0 to 2, where α is defined is always lower than the real workload that is calculated after its execution.

4.4 Performance With Upper Bound

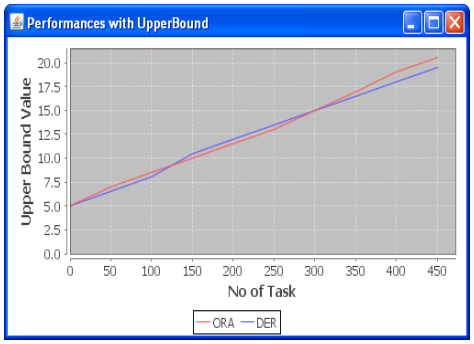

Fig 4.4 Performance With Upper Bound

Disscussions:
Optimal algorithm is based on the inaccuracy of the workload predicted, according to the analysis. From this figure, we can clearly observe that the prediction method used can make sure that the lower bound of the workload predicted (i.e., β value will be set close 0 to 20, where β is defined is always lower than the real workload that is calculated after its execution.

V.CONCLUSION AND FUTURE WORK

5.1 CONCLUSION:
This paper proposes a novel resource allocation algorithm for cloud system that supports VM-multiplexing technology, aiming to minimize[6] user's payment of task and also endeavor to guarantee its execution deadline meanwhile. This is proven that the output of this algorithm is optimal based on the KKT condition, which means any other solutions would definitely cause larger payment cost.

5.2 FUTURE ENHANCEMENT:
In the future, the plan to integrate this algorithms with stricter/original deadlines into some excellent management tools, for maximizing the system-wide performance. To improve the resource utilization and reduce the user payment the future work to fix the KKT based on dual parameters like deadline, cost or bandwidth.

REFERENCES:

[1]. Armbrust.M, Fox.A, Griffith.R, Joseph.A.D, Katz.R.H, Konwinski.A, Lee.G, Patterson.D.A, Rabkin.A, Stoica.I, and Zaharia.M,(2009), "Above the Clouds: A Berkeley View of Cloud Computing," Technical Report UCB/EECS-2009-28, EECS Dept., Univ. California, Berkeley.

[2]. Boyd.S and Vandenberghe.L,(2009), "Convex Optimization," Cambridge Univ. Press.

[3]. Chang.F, Ren.J, and Viswanathan.R, (2010). "Optimal Resource Allocation in Clouds," Proc. IEEE Int'l Conf. Cloud Computing, pp. 418-425.

[4]. Gupta.D, Cherkasova.L, Gardner.R, and Vahdat.A, (2006), "Enforcing Performance Isolation across Virtual Machines in Xen," Proc. ACM/IFIP/USENIX Int'l Conf. Middleware (Middleware '06), pp. 342-362.

[5]. Huang.L, Jia.J, Yu.B, Chun.B.G, Maniatis.P, and Naik.M,(2010), "Predicting Execution Time of Computer Programs Using SparsePolynomial Regression," Proc. 24th Conf. Neural Information Processing Systems (NIPS '10), pp. 1-9.

[6]. Mao.M and Humphrey.M,(2011), "Auto-Scaling to Minimize Cost and Meet Application Deadlines in Cloud Workflows," Proc. Int'l Conf. High Performance Computing, Networking, Storage & Analysis (SC '11), pp. 49:1-49:12.

[7]. Matthews.J.N, Hu.W, Hapuarachchi.M, Deshane.T, Dimatos.D, Hamilton.G, McCabe.M, and Owens.J, (2007), "Quantifying the Performance Isolation Properties of Virtualization Systems," Proc. Workshop Experimental Computer Science (ExpCS '07).

[8]. McElvany.M.C and Stotts.P.D, (1991), "Guaranteed Task Deadlines for Fault-Tolerant Workloads with Conditional Branches," Real-Time Systems, vol. 3, no. 3, pp. 275-30.

[9]. Meng.X, Isci.C, Kephart.J, Zhang.L, Bouillet.E, and Pendarakis.D, (2010), "Efficient Resource Provisioning in Compute Clouds via VM Multiplexing," Proc. Seventh Int'l Conf. Autonomic Computing (ICAC '10), pp. 11-20.

[10]. Nathuji.R, Kansal.A, and Ghaffarkhah.A,(2010), "Q-Clouds : Managing Performance Interference Effects for Qos-Aware Clouds," Proc. European Conf. Computer Systems (EuroSys '10), pp. 237-250.