2012 IEEE Fifth International Conference on Cloud Computing

Minimum Cost Maximum Flow Algorithm for Dynamic Resource Allocation in Clouds

Makhlouf Hadji, Djamal Zeghlache

Institut Telecom, Telecom SudParis
UMR CNRS 5157
9, Rue Charles Fourier
91011, Evry, France
Email: {makhlouf.hadji, djamal.zeghlache}@it-sudparis.eu

***Abstract*—A minimum cost maximum flow algorithm is proposed for resources (e.g. virtual machines) placement in clouds confronted to dynamic workloads and flows variations. The algorithm is compared to an exact method generalizing the classical Bin-Packing formulation using a linear integer program. A directed graph is used to model the allocation problem for cloud resources organized in a finite number of resource types; a common practice in cloud services. Providers can use the minimum cost maximum flow algorithm to opportunistically select the most appropriate physical resources to serve applications or to ensure elastic platform provisioning. The modified Bin-Packing algorithm is used to benchmark the minimum cost maximum flow solution. The latter combined with a prediction mechanism to handle dynamic variations achieves near optimal performance.**

***Keywords*-Cloud Computing, Resource Allocation, Linear Integer Programming, Minimum Cost Maximum Flow.**

I. Introduction

Optimal virtual machine placement in clouds has received considerable attention recently to facilitate cloud resources and services provisioning on an on demand basis to users and multiple tenants. The dynamic variations of workloads, jobs and application flows have nevertheless not been addressed with the same focus. The actual demand is typically not known in advance. This makes reservation based and on-demand cloud services provisioning difficult since it is not possible to reserve the right amount of resources a priori. The dynamic variations are typically observed during operations at the infrastructure or platform levels to scale cloud services (see for example: [1]). The idea is to predict these variations based on past observations to anticipate future demand by extending or adapting previous allocations probatively.

Authors in [1] use a demand forecaster to ensure evolutionary machine placement into multiple cloud providers. They use simulations and numerical studies to evaluate the performance of their algorithm and show that their algorithm is close to the optimal solution given by the stochastic integer program. They also indicate that the evolutionary algorithm can reach the minimum cost when the resource in the reservation plan is accurately provisioned compared with the actual demand, otherwise, additional costs are incurred.

The approach in our contribution also consists of predicting future loads and variations. To avoid the cost and complexity of [1] using a demand forecaster combining several predictors (Kalman filtering, double exponential smoothing and Markov prediction), we just resort to an autoregressive process to anticipate variations in incoming flows and workload dynamics. Instead of using a separate and external predictor, we embed the prediction and adaptation process in the algorithm itself and seek a near optimal solution that can achieve performance close to Bin-Packing algorithms that are optimal for virtual machine placement in virtualization enabled physical resources when the demand is known in advance. Our goal is also to derive a quasi exact algorithm instead of using meta-heuristics or evolutionary algorithms.

The use of optimal dynamic resource allocation leads to significant increase in providers' revenues as stated in [2] and gives the opportunity to adjust cloud resources prices to foster uptake as well as achieve access and congestion control. These aspects were addressed in [3] for cloud federations, using insourcing and outsourcing to increase revenues, using an exact mathematical model based on linear integer programming, to set prices dynamically and achieve optimal resource allocation and management.

In this paper we focus on optimal dynamic placement of virtual resources in data centers and cloud infrastructures to serve multiple users and tenants with time varying demands and workloads. As mentioned earlier static allocation policies and pricing lead to inefficient resource sharing, poor utilisation, waste of resources and revenue loss. The paper derives a low complexity and effective algorithm by modeling the optimal resource placement in clouds as a maximum flow problem leading to a viable implementation for cloud providers. Without any loss in generality, we consider that resources are offered as Virtual Machines (VMs) to focus on the model instead of overemphasizing complex resources in the study. Details on complex resources can be found in

[4]. Since our algorithm applies to complex resources with marginal adaptation, resources are limited to Virtual Machines to be placed in physical resources acting as containers.

Our objective is to allocate demands, expressed as a vector of VM instances $D = (d^{(small)}, d^{(medium)}, d^{(large)}, \ldots)$, to different data centers (or Physical Resources) to maximize cloud provider's revenues. We resort to time-series analysis to predict or forecast demands based on past and current observations and use dynamic prices to achieve near optimal placement and revenue gains.

Two placement algorithms are investigated. The first is a modified exact Bin-Packing algorithm serving as a reference to benchmark a second algorithm proposed by our work, the minimum cost maximum flow algorithm (MCMF). The problem at hand is NP-Hard [5] [6] and the Bin-Packing approach encounters scalability issues when the number of instances and the number of events grow; it exhibits exponential explosion and leads to unacceptable delays. The MCMF is an alternative that performs very well and matches the global optimum most of the time. The MCMF will also be shown to convergence to quasi optimal solutions in much shorter times and does not suffer from state explosion.

Section II of this paper introduces the time-series model used to predict future demands. An auto-regressive method $(AR(k))$ is used to forecast demands for a historical period k. The exact modified Bin-Packing model that uses a linear integer programming formulation and the proposed Minimum Cost Maximum Flow (MCMF) are presented in Section III while section IV addresses their performance evaluation. Related work on cloud computing optimization and dynamic resources allocation in cloud computing is presented for completeness in Section V. Conclusions and future research are presented in Section VI.

II. The System Model

The model considers a cloud provider offering virtual resources (CPU, Memory, I/O Bandwidth, ...) (seen as VMs) from physical resources (seen as Physical Machines (PMs)). A PM can be switched ON or OFF. Providers shut down or put PMs into sleep mode to save energy and reduce cost whenever appropriate. When a PM is power ON a cost is associated to the objective function of the models to include the energy bill according to workload. Different cost functions are considered for each PM to capture this energy saving practices by the providers. These costs are assessed, known and provided by the providers themselves.

Figure 1 depicts an exemple where physical machine 1 (PM$_1$) can offer 2 small VMs instances and large 1 VM instance while PM$_2$ and PM$_n$ offer respectively during a given allocation window or cycle, (1 Medium, 1 Large) VM set and (1 Small, 1 Large) set. These offered resources will vary dynamically with completed tasks, new requests and even when applications migrate across physical machines or cloud providers. During an allocation cycle Δt, a cloud provider controls these free capacities, turns physical machines ON or OFF and simultaneously aims at assigning optimally the demands to the PMs. The provider will turn PMs ON when the demand exceed the activated fraction of the overall capacity to avoid rejecting user requests and thus maintain good reputation. The provider will aim at maximizing their return on investment at all time to balance cost and revenue.

The proposed dynamic resource allocation and pricing framework is illustrated in figure 1. As already mentioned small, medium and large VM instances are offered by the provider that sets prices dynamically during the allocation cycles Δt according to predicted future VMs demands by a forecaster. Per unit of resource prices or costs incurred to users should decrease as a function of their demand and without loss of generality we start off with a simple price function in line with this usual practice for a demand d_j of VM instance $j \in \mathcal{I}$:

$$P_j = \begin{cases} \frac{1}{d_j}, & \text{if } d_j > 0; \\ 0, & \text{if } d_j = 0. \end{cases}$$

Given the prediction by the forecaster, providers with knowledge of their free resources and adapted prices per instance type can rely on their VM scheduler to allocate demands to PMs with the objective of maximizing their revenues. The *Forecasting* and *VM Scheduler* work in tandem to enable providers to have the right amount of physical resources ready to fulfill the current and future demands.

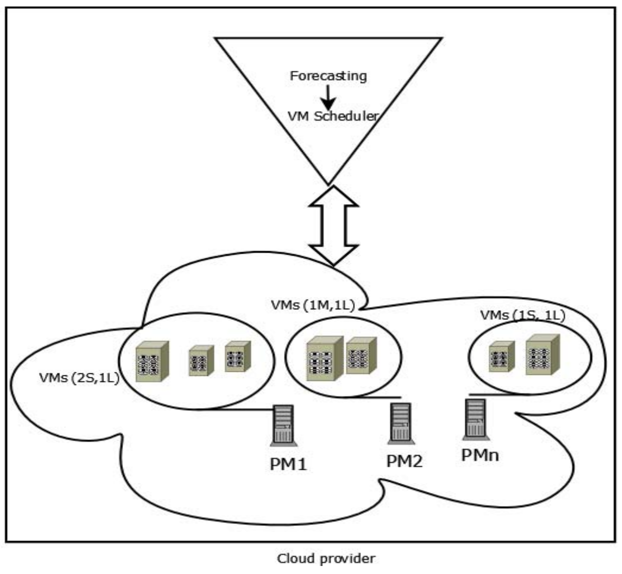

Fig. 1. The system model

A. Forecasting future demands

The prediction of future demands in virtual machines is essential to reserve the right amount of physical resources and achieve durable optimal placement in physical hosts at initial provisioning. Even if cloud services include elasticity services that can react to such variations, the ability to predict demands can considerably reduce the need for frequent and disrupting adaptation, migration and consolidation changes to absorb fluctuations.

Different forecasting models can be found in the literature (see [2], [7] for example). The objective of forecasting demands in clouds is essentially to minimize costs, achieve better and longer term resource allocations and distribute the demand optimally across data centres and physical resources.

There exist several methods to predict requests as exemplified in [1] and time series models [8] that we have adopted for our minimum cost maximum flow algorithm to achieve dynamic optimal placement in clouds. Since our objective is to assess the efficiency and complexity of our algorithm compared to Bin-Packing, we adopt a simple auto-regressive model (AR(k)) where k is the historical size. We limit ourselves to a simple AR prediction as we intuit that using moving averages (ARMA) and extensions (ARIMA) will enhance the prediction outcomes and performance.

The AR(k) model estimates the future request d_t for a time t given by :

$$d_t = \sum_{i=1}^{k} \varphi_i d_{t-i} + \varepsilon_t \tag{1}$$

where the parameters φ_i can be easily estimated using the historical values of the demands, and ε_t is uncorrelated white noise with mean 0 and a variance σ^2.

B. Modified Bin-Packing algorithm

Basically, packing problems consider sets of items and item-holding objects called bins, and aim to group items in such a way that they all (or a maximum number of them) fit into the minimum number of bins [6]. In our case, bins are the different Physical Machines noted by PMs, and the items are the Virtual Machines Instances (for example, we consider small, medium and large instances) to fit into PMs.

In this paper, we introduce a modified and extended Bin-Packing algorithm that is more appropriate for the dynamic placement problem at hand to have a reliable, exact and well established method for benchmarking the minimum cost maximum flow algorithm. We derive the Bin-Packing model for a finite set of VM instances and with no loss of generality specialize the instances to a set often used in cloud computing services (Amazon, Azure, others ...): $\mathcal{I} = \{Small, Medium, Large, \ldots\}$. The problem is characterized by a number of items (the VMs), a profit (price per resource unit) and a set of bins (the PMs that will host the VMs) with their associated free (left over) capacity and costs (hosting and PM usage costs that include energy consumption costs). The aim is to select the subsets of profitable items (the VMs) and appropriate bins (the PMs) to optimize the objective function combining the cost of using the PMs and the profit obtained by hosting the selected VMs.

Let us now derive the model by defining n as the number of available PMs from a cloud provider and noting by d_j the demand in VM instances of type j at time t. Variable C_{ij} is used to represent the available hosting capability of PM i with respect to VMs instances of type j. A gain (price per unit), P_j is also associated to each instance $j \in \mathcal{I}$. This price is set after forecasting the future demand.

Assuming full knowledge by the provider of all lumped costs (including energy bill), the composite hosting cost, when assigning VM j to PM i is represented by γ_{ij}. Decision variable x_{ij} is introduced to reflect that VM instances of type j are assigned to physical machine i.

In order to take into account the need to power physical machines ON if the currently activated resources are insufficient to satisfy the predicted demand, binary variable y_{ij} is introduced. This variable indicates if PM i is powered ON to host a VM instance j.

The cloud provider using a *VM Scheduler* finds the best resource allocation to maximize revenue expressed by: $\sum_{i=1}^{n}\sum_{j=1}^{|\mathcal{I}|} P_j x_{ij}$ and to minimize the corresponding costs $\sum_{i=1}^{n}\sum_{j=1}^{|\mathcal{I}|} \gamma_{ij} y_{ij}$. This leads to the use of the objective function in equation (2) to achieve optimal placement.

This optimisation is subject to a number of constraints expressed by inequality (3). The cloud providers new allocations can not exceed the available free resources in the data-centers. An additional family of constraints is given by equality (4) to satisfy the current demand and the integrity constraints expressed by (5) and (6).

This leads to an exact model extending the Bin-Packing problem when heterogeneous items are involved as in the addressed dynamic cloud resource allocation in this paper:

$$\min Z = \sum_{i=1}^{n}\sum_{j=1}^{|\mathcal{I}|} \gamma_{ij} y_{ij} - \sum_{i=1}^{n}\sum_{j=1}^{|\mathcal{I}|} P_j x_{ij} \tag{2}$$

Subject To:

$$x_{ij} \le C_{ij} y_{ij}, \forall j \in \mathcal{I}, \forall i = 1, \ldots, n \tag{3}$$

$$\sum_{i=1}^{n} x_{ij} = d_j, \forall j \in \mathcal{I} \tag{4}$$

$$y_{ij} = \begin{cases} 1, & \text{if the PM } i \text{ is ON to assign the VM } j \text{ to it;} \\ 0, & \text{elsewhere.} \end{cases} \tag{5}$$

$$x_{ij} \in \mathbb{N}, \forall j \in \mathcal{I}, \forall i = 1, \ldots, n \tag{6}$$

For completeness all used variables and constraints are listed below:

- n is the number of available Physical Machines.
- $\mathcal{I}$ is the set of all available VM instances types. $|\mathcal{I}|$ represents the number of considered instances. For example, for $\mathcal{I} = \{small, medium, large\}$ this gives $|\mathcal{I}| = 3$.
- d_j represents the demand in VMs of instance j.
- C_{ij} represents the number of available (free) VMs of instance j offered by the Physical Machine i.
- x_{ij} indicates if a VM of instance j is assigned to Physical Machine i.
- y_{ij} is the associated variable when we power ON a PM i to assign a VM j to it.
- P_j is the price (gain) of one VM of instance j.
- γ_{ij} is the associated hosting cost when a VM of instance j is assigned to a Physical Machine i.

C. Minimum Cost Maximum Flow algorithm

The exact formulation based on the modified Bin-Packing model suffers from scalability problems with large instances and increasing number of PMs as well as the length of the requests. This has been the motivation for seeking an alternate approach to the dynamic resource placement problem and this led to the Minimum Cost Maximum Flow (MCMF) algorithm proposed in this work. The MCMF on the contrary is not subject to the scalability issues of Bin-Packing.

The MCMF is based on a directed graph representation of the dynamic resource allocation problem. The directed graph is represented by $G = (V, E)$ with V corresponding to the set of vertices and E is the set of arcs.

The number of vertices for the graph G is fixed to $|V| = 2 + |\mathcal{I}|(2+|PM|)$ and the number of arcs to $|E| = 2|\mathcal{I}|(1+|PM|)$ where $|PM|$ represents the number of available PMs in the cloud provider. We consider two fictitious vertices S and T representing the source and the destination respectively. For each type of instance, we create $|PM|$ vertices linked with two other fictitious vertices. One vertex, u_i, is located at the entry of this set, and the second one, v_i is located at the exit. This overall set is noted A_i with $i \in \mathcal{I}$. For example, if the request asks for three instances (small, medium, large), three sets of vertices are created. Figure 3 depicts the created graph and the position of all its vertices (PMs and fictitious vertices). The corresponding arcs are created using the steps listed below:

- We associate an arc from S to each entry vertex of A_i (noted by u_i) for all $i \in \mathcal{I}$. We also associate a capacity to this arc (S, u_i) equal to the requested demand in instance i noted by d_i. A cost equal to zero is associated to this arc as it just represents the initial resource request.
- From each exit vertex v_i of a component A_i (for all $i \in \mathcal{I}$), to the destination vertex T (we create the arc (v_i, T)) to which we associate a capacity equal to the demand in instance i (i.e. d_i) and a cost equal to zero.
- For each component A_i (for all $i \in \mathcal{I}$), we repeat the following steps:
  - We associate an arc from the entry vertex u_i of A_i to all intermediate vertices representing PMs and noted by $j_1^i, j_2^i, j_{|PM|}^i$. On each constituted arc (u_i, j_k^i) for $k = 1, \ldots, |PM|$, we associate a capacity representing the available (free) resources (VMs in our case) on the PM$_k$ and a cost function g_{cost} witch will be detailed bellow.
  - We add an arc from each intermediate vertex j_k^i for $k = 1, \ldots, |PM|$ to the exit vertex v_i of A_i. This arc is represented by (j_k^i, v_i). The same capacity as on the arc (u_i, j_k^i) will be associated to (j_k^i, v_i) and a cost equal to zero is assigned to this arc.
  - A last case consists in assigning a zero capacity value and an infinite cost to the arcs (u_i, j_k^i) and (j_k^i, v_i) to all PMs j_k^i ($k = 1, \ldots, |PM|$) that are either turned OFF or have no resources left to allocate.

Figure 2 illustrates graph G for a one dimensional request. Generalizing this graph construction to multidimensional requests leads to the graph G depicted in Figure 3 for the small, medium and large resource instances.

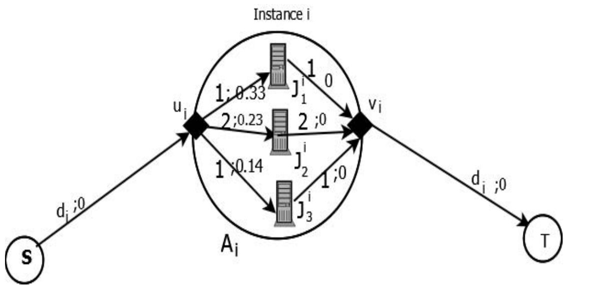

Fig. 2. The graph G for a dynamic demand of one instance type

Each arc on the graph is labeled with a tuple that consists of its available capacity and proposed cost. For example, the arc (u_i, j_1^i) has a capacity of 1 and a cost of 0.33 as can be observed in Figure 2.

To compute a minimum cost maximum flow in graph $G = (V, E)$ from S to T, we use the Edmonds-Karp algorithm [9] to return the best (optimal) PMs that will dynamically provide resources and host the requests (VMs or virtual resources). The complexity of this algorithm is the minimum of $O(|V|^2 flow)$ and $(|V|^3 fcost)$ where $flow$ is the obtained flow on G and $fcost$ is its corresponding minimum cost. The output of the algorithms are the PMs that will handle the demands and convey the associated flow from the source S to the destination

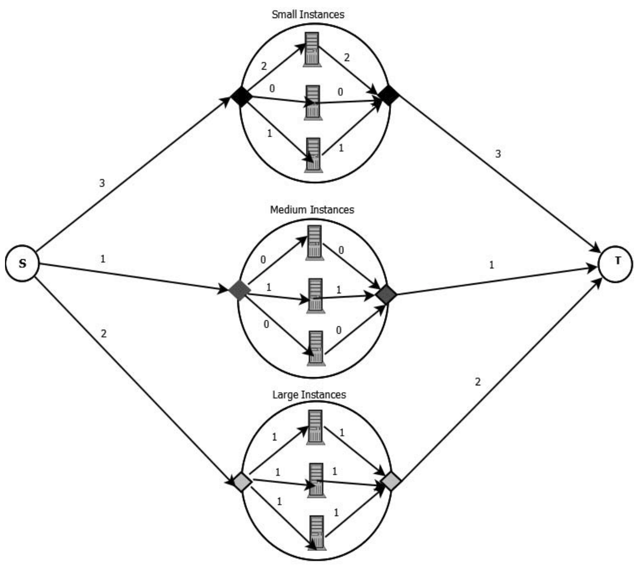

Fig. 3. The generalized directed graph G (costs are missed for clarity)

T. If the outgoing flow form S is equal to the incoming flow in T, the demand is accepted. Otherwise it is rejected. Note that when a physical machine is selected to carry a flow for more than one instance it is counted only once in the cost function. Indeed, when a physical machine is turned ON for the first time it is counted and obviously not counted for all subsequent requests as long as it stays ON.

III. Numerical results

The MCMF algorithm and the modified Bin-Packing algorithm are evaluated using numerical analysis. The objective is to assess the performance of the MCMF algorithm combined with the AR predictor using the modified Bin-Packing as a benchmark. The exact model of the modified Bin-Packing problem is evaluated through a C++ language implementation and the linear programming solver CPLEX [10]. The proposed MCMF is evaluated through the Edmonds-Karp [9] algorithm. The assessment scenarios considers a cloud provider using different physical machines or nodes to host virtual resources offering multiple virtual resource types.

Without loss of generality, we specialize the scenario to three cases corresponding to 3, 5 and 8 instances and report the results for the dynamic resource allocation problem addressed in this paper. The requests for virtual resources or demands D towards a cloud provider are generated as an independent Poisson process with rate λ.

Figure 4 depicts the evolution of the cloud provider costs for fixed and dynamic prices and indicates that it is essential to use dynamic prices to achieve stable and reliable revenues. The use of fixed prices leads to unstable and often high costs. For the dynamic pricing based on future demands prediction a stable gain can be obtained and maintained. For the fixed price approach the losses can be significantly high when low prices are applied for high demands. When cloud providers forecast the demand and adjust prices accordingly, they are capable of minimizing cost more accurately by adapting prices and can serve more customers.

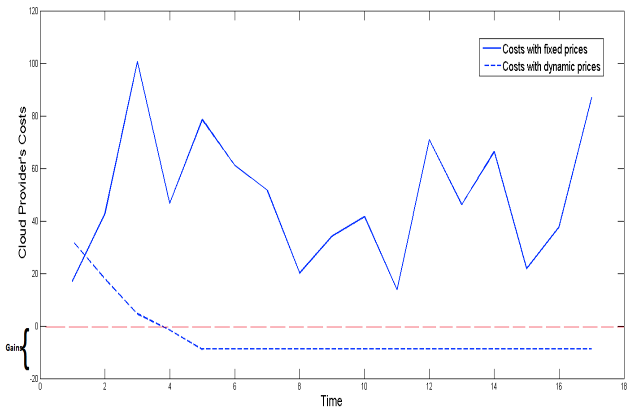

Fig. 4. Cloud provider's costs behaviour with fixed and dynamic prices

To conduct the performance evaluation, 100 random and independent simulations for each case are used with 20 to 200 PMs and a number of cost functions g_{cost} related to the hosting costs γ. The selected costs for the studies reflect the intuitive expectation that the cost should be inversely proportional to the available capacity so as to foster demand when resources are unused and tamper user requests when resources become too busy.

1) Inverse hosting costs: In this experiment, for each arc (u_i, j_k^i) in the graph G (see figure 3), we will consider the hosting cost function g_{cost} given as follows:

$$g_{cost}(i,k) = \begin{cases} \frac{1}{C_{ik}}, & \text{if } C_{ik} > 0; \\ \infty, & \text{elsewhere.} \end{cases} \quad (7)$$

where C_{ik} is the available capacity on arc (u_i, j_k^i) $(k = 1, \ldots, |PM|)$.

For this specific scenario and the cost function in (7), the difference in costs between the two algorithms is depicted in Figure 5. The figure contains in fact six curves, three for the modified Bin-Packing for the 3, 5 and 8 instances and three entirely superposed to these corresponding to the MCMF and the same sets of instances. The MCMF for this cost always finds the optimal Bin-Packing placement. We have also tested alternate cost functions such as $\frac{1}{log(C_{ik}+1)}$ and $\left(\frac{1}{C_{ik}}\right)^a$ (with $a \geq 1$) and observed that the MCMF achieved the same optimal performance as the modified Bin-Packing. The MCMF algorithm with $\frac{1}{f(Cj)}$ hosting costs was observed

to always lead to the optimal Bin-Packing solutions for other than Poisson demand arrival distributions. Additional assessments using arbitrary bursty arrivals and very different distributions all lead to optimal Bin-Packing.
This motivates us to conjecture that the MCMF is optimal for all costs of type $\frac{1}{f(C_j)}$. Proof of this optimality is out of scope of the current paper, however, as we seek in priority to understand how the MCMF behaves with cost and with different cost models. In order to get a clear answer on how close MCMF with AR prediction performs compared to optimal, when faced with important variations in demands, we resort to a test using a random cost to reveal potential weaknesses.

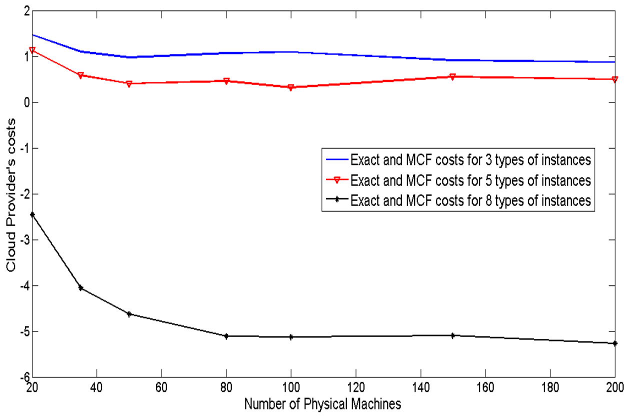

Fig. 5. Cloud provider's costs for three, five and eight type of instances: case of inverse hosting costs

2) Random (0,1) hosting costs : The second scenario considers consequently a cost function g_{cost} for all arcs (u_i, j_k^i) $(k = 1, \ldots, |PM|)$ taking random values in 0 and 1. Figure 6 summarizes the gaps in performance with respect to the exact and optimal modified Bin-Packing algorithm.

The deviation from optimal are consistently small (for less than 80 PMs in figure 6) and tend to vanish when the number of PMs is high (above 80 PMs). This means that the MCMF algorithm will be very close to optimal for a large number of physical machines for large cloud providers with many data centers a sector where actually the Bin-Packing algorithm encounters scalability problems and takes longer times to find the optimal solution.

From figure 6, we conclude that using more resource instances in the provider service offer leads to higher revenues (negative losses on the curves). Using 5 and 8 instances produces gains with a clear advantage for 8 instances. Providers will need however to strike a balance between increasing the number of instances for higher revenues with increasing complexity for customers or users.

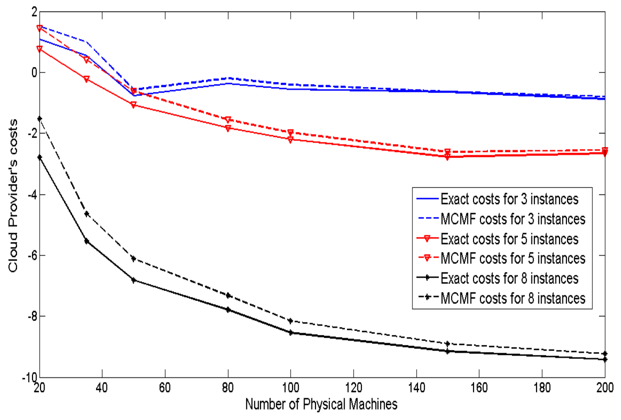

Fig. 6. Cloud provider's costs for three, five and eight type of instances: case of random hosting costs between 0 and 1

IV. Related work

In [11], authors describe multiple problems and open challenges concerning resource allocation in cloud computing environment. They noted the importance of tacking into account the consumption of huge amounts of energy caused by hosting applications from consumers. Thus, they presented vision, challenges and architectural elements for energy-efficient management of cloud computing environments based on dynamic resource provisioning and allocation algorithms considering synergy between various data center infrastructures.

Authors in [2] investigate the combination of workload prediction algorithms and switching policies to solve the dynamic resource allocation for enterprise applications. They presented two switching policies: proportional switching policy (PSP) and bottleneck aware switching policy (BSP). They illustrated via simulation results that combining of BSP with different workload predictions can clearly improve system revenues.

Reference [12] addresses an analytical model of virtual machine migration that provides improvement in response time due to a migration decision. The presented model accounts for predictability of resource requests and characteristics of the virtualization of the infrastructure. To validate their model, they used simulations on data center resource utilization traces.

The closest work to our model appears in [7]. The approach in this case consists in two parts: (1) a market analysis for forecasting the demand of each spot market and (2) a dynamic scheduling and consolidation mechanism that allocate resource to each spot market to maximize total revenue. They showed that the proposed algorithm can approximate the optimal solution of the NP-Hard problem of resource allocations in both fixed and variable pricing schemes. In our approach, we aim at a general solution using a Minimum

Cost Maximum Flow algorithm that achieves optimal virtual resource or service placement in polynomial time and staying very close to optimal for all inverse hosting cost functions of type $\frac{1}{f(C)}$. As mentioned earlier, we avoid resorting to meta-heuristics and evolutionary algorithms and search for solutions that can apply to any market and adapt to demand variations. MCMF is proposed as a scalable and viable solution to reach this objective.

V. Conclusions

An exact modified Bin-Packing problem and a Minimum Cost Maximum Flow (MCMF) are proposed and evaluated to address the dynamic resource allocation problem in cloud computing environments. A simple autoregressive process is used to predict fluctuating demands to set prices dynamically to ensure gains instead of experiencing losses and hence increase revenues for providers. The proposed MCMF algorithm uses this time-series model to forecast future requests and exhibits very good performance and scalability properties as opposed to the modified Bin-Packing algorithm that is NP-hard and subject to exponentially increasing delays in finding the optimal solution for large clouds. The $O(\min\{n^2 flow; n^3 fcost\})$ complexity of MCMF can be considered as low and negligible compared to the modified Bin-Packing solution. The MCMF also finds the optimal solutions for cost functions that lower prices as resources become more available and act inversely when resources are too busy. A test using a random cost, having no business or market sense, is used to assess the robustness of the MCMF to errors in matching prices with dynamic demands. The MCMF resists especially well to such scenarios and stays close to the optimal exact Bin-Packing algorithm while taking much less time to find a viable solution to the dynamic resource allocation problem in clouds.

Our future research will address solutions to the dynamic resource allocation problem taking into account the costs of networking and of migration of applications between data centers.

Acknowledgment

We would like to thank Professor Thierry-Duvillard Cabanal from the university of Paris 5 for his constructive comments.

References

[1] C. T. Ching, D. Niyato, and C.-K. Tham, "Evolutionary optimal virtual machine placement and demand forecaster for cloud computing," *In Proceedings of IEEE International Conference on Advanced Information Networking and Applications (AINA)*, March 2011.

[2] M. Al-Ghamdi, A. Chester, and S. Jarvis, "Predictive and dynamic resource allocation for enterprise applications," *IEEE International Conference on Computer and Information Technology (CIT 2010)*, pp. 2776–2783, 2010.

[3] M. Hadji, D. Zeghlache, and W. Louati, "Resource allocation in cloud federation: Exact algorithm," *Submitted*, December 2011.

[4] J. Rao, X. Bu, K. Wang, and C. Xu, "Self-adaptive provisioning of virtualized resources in cloud computing," *Proceedings of the ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems*, p. 13, June 2011.

[5] M. Garey, and D.S. Johnson, *Computers and Intractability.* W.H. Freeman and Company, 1979.

[6] M. Maria-Baldi, T. Crainic, G. Perboli, and R. Tadei, "The generalized bin packing problem," *Technical Report, CIRRELT*, July 2011.

[7] Q. Zhang, E. Gurses, R. Boutaba, and J. Xiao, "Dynamic resource allocation for spot market in clouds," *Proceedings of the 11th USENIX conference on Hot topics in management of internet, cloud, and enterprise networks and services*, 2011.

[8] P.J. Brockwell, and R. A.Davis, *Introduction to time series and forecasting*, ser. 0-387-95351-5. Springer 2nd edition, 2002.

[9] J. Edmonds and R. Karp, "Theoretical improvements in algorithmic efficiency for network flow problems," *Journal of the Association for Computing Machinery*, vol. 19, no. 02, pp. 248–264, April 1972.

[10] http://www-01.ibm.com/software/integration/optimization/cplex optimizer/.

[11] R. buyya, A. Beloglazov, and J. Abawajy, "Energy-efficient management of data center resources for cloud computing: A vision, architecture elements, and open chalenges," *Proceedings of the 2010 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2010)*, p. 12, July 2010.

[12] A. Kochut and K. Beaty, "On strategies for dynamic resource management in virtualized server environments," *Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, MASCOTS'2007*, pp. 193–200, 2007.