Dynamic Resource Provisioning in Cloud Computing: A Randomized Auction Approach

Linquan Zhang, Zongpeng Li[1] Chuan Wu[2]

[1]University of Calgary, [2]The University of Hong Kong

December 12, 2013

Outline

Introduction

Nowadays, virtualization technologies help cloud providers pack their resources such as CPU, RAM and storage into different types of virtual machines (VMs), for allocation to cloud users.

Introduction

Table : Amazon EC2 Virtual Machine Instance Types

VM type	CPU	Memory	Storage
m1.medium	2 EC2 Compute Units	3.75 GB	410 GB
m1.xlarge	8 EC2 Compute Units	15 GB	1680 GB
c1.medium	5 EC2 Compute Units	1.7 GB	350 GB
c1.xlarge	20 EC2 Compute Units	7 GB	1680 GB
m2.xlarge	6.5 EC2 Compute Units	17.1 GB	420 GB
hi1.4xlarge	35 EC2 Compute Units	60.5 GB	2048 GB

Introduction

- Because users have different demand
- VM heterogeneity
- More choices, more revenue

Introduction

Unfortunately, existing allocation mechanisms in cloud markets either are based on

- fixed pricing, which is economically inefficient, or
- resort to simple, static auctions that treat VMs as type-oblivious commodities. More specifically, it is usually assumed that either a single type of VMs exists in the cloud market, or VMs are substitutes in that a high-end VM is equivalent to a number of low-end VMs, *e.g.*, a Type II (2 $\times$ Core, 2 GB RAM, 40 GB Disk) VM equals two Type I (1 $\times$ Core, 1 GB RAM, 20 GB Disk) VMs

Introduction

This work further departs from the existing literature by explicitly modelling the dynamic provisioning of VM instances from cloud resources.

- Under *static provisioning*, the cloud assembles its available resources into different types of VMs based on simple heuristics or historical VM demand patterns, before the auction starts.
- Under *dynamic provisioning*, the cloud conducts VM assembling in an online fashion upon receiving VM bundle bids , targeting maximum possible social welfare given the current bid profile.

Our Contributions

- Allocation: an approximation algorithm, with ratio α, (≈ 4.4)
- Auction: randomized, computationally efficient, truthful in expectation, still with ratio α

System Model

- Provider, (auctioneer)
- User, (bidder)
- One round
- t types of resources (CPU, RAM, disk,...)
- Total amount of resource k: c_k
- VM_1, VM_2,....., VM_m
- VM_j consumes r_j^k units of resource k

System Model

- B_i: user i's bid, contains several optional bundles
- A bundle $\mathcal{S} \in B_i$ specifies amount of instances of each VM type
- An n-tuple: $(n_1^{\mathcal{S}}, \ldots, n_m^{\mathcal{S}})$, corresponding to each VM type
- Bidding price: $b_i(\mathcal{S})$
- $x_i(\mathcal{S}) \in \{0, 1\}$, wins bundle or not

System Constraints

We adopt the XOR bidding language, in which a user can win at most one bundle even if it submits multiple bundles , leading to the first constraint for VM allocation:

$$\sum_{\mathcal{S} \in B_i} x_i(\mathcal{S}) \leq 1, \forall i \in \mathcal{B} \tag{1}$$

System Constraints

The finite supply of each type of cloud resource translates into the capacity constraint at the cloud provider:

$$\sum_{i \in \mathcal{B}} \sum_{\mathcal{S} \in B_i} x_i(\mathcal{S}) n_j^{\mathcal{S}} \leq N_j, \forall 1 \leq j \leq m$$
$$\sum_{j=1}^{m} N_j r_j^k \leq c_k, \forall 1 \leq k \leq t \tag{2}$$

where N_j is the number of VM_j instances provisioned. The two groups of inequalities in (2) can be merged into an equivalent, more compact capacity constraint:

$$\sum_{i \in \mathcal{B}} \sum_{\mathcal{S} \in B_i} x_i(\mathcal{S}) (\sum_{j=1}^{m} n_j^{\mathcal{S}} r_j^k) \leq c_k, \forall 1 \leq k \leq t \tag{3}$$

Social Welfare Maximization Problem

The social welfare maximization problem can now be formulated:

$$\text{maximize} \quad DP(\mathcal{B}) = \sum_{i \in \mathcal{B}} \sum_{\mathcal{S} \in B_i} b_i(\mathcal{S}) x_i(\mathcal{S}) \tag{4}$$

subject to:

$$\sum_{\mathcal{S} \in B_i} x_i(\mathcal{S}) \leq 1, \qquad \forall i \in \mathcal{B} \tag{4a}$$

$$\sum_{i \in \mathcal{B}} \sum_{\mathcal{S} \in B_i} x_i(\mathcal{S}) \left(\sum_{j=1}^{m} n_j^{\mathcal{S}} r_j^k\right) \leq c_k, \qquad \forall 1 \leq k \leq t \tag{4b}$$

$$x_i(\mathcal{S}) \in \{0, 1\}, \quad \forall i \in \mathcal{B}, \mathcal{S} \in B_i \tag{4c}$$

Definition of Truthfulness

Definition 1

A (randomized) auction is *truthful* (in expectation) if for any bidder i, reporting its true valuation in the bid maximizes its (expected) utility, regardless of the bids submitted by other bidders.

That is, under a truthful auction, selfish buyers have no incentive to submit falsified and strategic bids.

The Celebrated VCG Auction

• A well-known type of truthful auctions is the celebrated VCG mechanism , which is proven to be the only type of auctions that can simultaneously guarantee **truthfulness** and **economic efficiency** (social welfare maximization).

The Celebrated VCG Auction

• A well-known type of truthful auctions is the celebrated VCG mechanism , which is proven to be the only type of auctions that can simultaneously guarantee **truthfulness** and **economic efficiency** (social welfare maximization).
• Unfortunately, a VCG auction requires solving the NP-hard problem of social welfare optimization multiple times for calculating externalities as user payments, and becomes computationally infeasible as the system size grows.

NP hardness

Theorem 2

The social welfare maximization problem defined in IP (4) is NP-hard.

Proof Sketch: The proof is based on a polynomial-time reduction from the knapsack problem, a classic combinatorial optimization problem that is proven NP-hard. □

NP hardness

Theorem 2

The social welfare maximization problem defined in IP (4) is NP-hard.

Proof Sketch: The proof is based on a polynomial-time reduction from the knapsack problem, a classic combinatorial optimization problem that is proven NP-hard. □

Theorem 2 reveals that solving IP (4) to optimal is NP-hard, implying that applying the VCG auction for truthfulness is computationally expensive.

A Primal-dual Cooperative Approximation Algorithm

Algorithm 2 The Primal-Dual Approximation Algorithm

1: // Initialization
2: $\Lambda = \min_{1 \le k \le t} c_k / R_k$; $p = 0$; $\mathcal{U} = \emptyset$;
3: $\forall i, \forall S : x_i(\mathcal{S}) = 0$; $\forall i : y_i = 0$; $\forall k : z_k = 1/c_k$;
4: // Iterative update of primal and dual variables:
5: **while** $\sum_{k=1}^{t} c_k z_k < t \exp(\Lambda - 1)$ AND $\mathcal{U} \neq \mathcal{B}$ **do**
6: **for all** $i \in \mathcal{B} \setminus \mathcal{U}$ **do**
7: $\mathcal{S}_i = \arg\max_{\mathcal{S} \in B_i} \{b_i(\mathcal{S})\}$;
8: **end for**
9: $\mu = \arg\max_{i \in \mathcal{B} \setminus \mathcal{U}} \left\{ \frac{b_i(\mathcal{S}_i)}{\sum_{k=1}^{t} \sum_{j=1}^{m} n_j^{\mathcal{S}_i} r_j^k z_k} \right\}$;
10: $x_\mu(\mathcal{S}_\mu) = 1$; $y_\mu = b_\mu(\mathcal{S}_\mu)$;
11: $p = p + b_\mu(\mathcal{S}_\mu)$; $\mathcal{U} = \mathcal{U} \cup \{\mu\}$;
12: **for all** $1 \le k \le t$ **do**
13: $z_k = z_k \cdot (t \exp(\Lambda - 1))^{(\sum_{j=1}^{m} n_j^{\mathcal{S}_\mu} r_j^k)/(c_k - R_k)}$;
14: **end for**
15: **end while**

Approximation Ratio

Algorithm 2 guarantees an α-approximation of social welfare, where $\alpha = 1 + \epsilon \frac{\Lambda}{\Lambda - 1}(et^{1/(\Lambda-1)} - 1)$.

Theorem 3

Algorithm 2 computes an α-approximate solution to IP (4) in polynomial-time, where $\alpha = 1 + \epsilon \frac{\Lambda}{\Lambda - 1}(et^{1/(\Lambda-1)} - 1)$.

In practice, the volume of a cloud provider's resource pool is substantially larger than a single user demand, *i.e.*, $\Lambda \gg 1$. The number of resource types t is a small constant (3 to 5). Consequently,

$$\lim_{\Lambda \to \infty} \alpha = \lim_{\Lambda \to \infty} (1 + \epsilon \frac{\Lambda}{\Lambda - 1}(et^{1/(\Lambda-1)} - 1)) = 1 + \epsilon(e - 1)$$

which suggests that the approximation ratio α is close to $1 + \epsilon(e - 1)$. When $\epsilon = 2$, $\alpha \approx 4.437$.

LP Relaxation

Theorem 2 reveals that solving IP (4) is NP-hard, and is computationally infeasible for a large input. Nonetheless, we may consider the *LP relaxation* of IP (4) by relaxing its last constraint (4c) to:

$$x_i(\mathcal{S}) \geq 0, \forall i \in \mathcal{B}, \mathcal{S} \in B_i \qquad \text{(4c')}$$

High-level Overview of the Structure of the Randomized VM Auction

- We first simulate a fractional VCG auction based on the linear programming relaxation (LPR) of the social welfare maximization IP.
- Then we utilize a pair of tailored primal and dual LPs to decompose the optimal fractional solution of the LPR into a weighted combination of integer solutions to the IP. This pair of LPs exploit the underlying packing nature of the social welfare maximization IP, and are solved using the ellipsoid algorithm with the cooperative α-approximation algorithm acting as a separation oracle.
- Each integer solution is selected randomly with probability equal to its corresponding weight calculated during the decomposition, and contains information for instructing the cloud provider to conduct both VM provisioning and VM allocation.

A Randomized Auction Mechanism

Algorithm 3 A Randomized Combinatorial VM Auction

1: **Simulating the fractional VCG auction.**
2: — Compute the fractional VCG allocation $\mathbf{x}^*$ and payment $\mathbf{\Pi}^F$, through solving the LPR of IP (4).

A Randomized Auction Mechanism

Algorithm 3 A Randomized Combinatorial VM Auction

1: **Simulating the fractional VCG auction.**
2: — Compute the fractional VCG allocation $\mathbf{x}^*$ and payment $\mathbf{\Pi}^F$, through solving the LPR of IP (4).
3: **Decomposing fractional solution into integer solutions**
4: — Decompose the scaled down fractional solution $\mathbf{x}^*/\alpha$ to a convex combination of integer solutions, *i.e.*, $\mathbf{x}^*/\alpha = \sum_{l\in\mathcal{I}} \beta_l \mathbf{x}(l)$, through solving a pair of primal-dual LPs in (6) and (7) using the ellipsoid method, leveraging the cooperative approximation algorithm as a separation oracle.
5: **Randomized VM allocation**
6: — Select each $\mathbf{x}(l)$ randomly with probability β_l.
7: **Charging scaled fractional VCG prices**
8: — for each winning cloud user $i \in \mathcal{B}$: charge a price $\Pi_i = \Pi_i^F/\alpha$.

The Fractional VCG Auction

The Fractional VCG Auction

We first resort to a fractional version of the VCG auction for achieving both computational efficiency (polynomial time complexity) and economic efficiency (social welfare maximization), by applying the VCG mechanism to the LPR instead of IP (4).The optimal solution $\mathbf{x}^*$ to the LPR constitutes the VM allocation solution in the fractional VCG auction.

The Fractional VCG payments

The fractional VCG payment for user i equals i's externality, or the difference in social welfare with and without i's bid

$$\Pi_i^F = V_i - \sum_{i' \neq i, i' \in \mathcal{B}} \sum_{\mathcal{S} \in B_{i'}} b_{i'}(\mathcal{S}) x_{i'}^*(\mathcal{S}) \tag{5}$$

where V_i is the optimal $DP^F(\mathcal{B})$ to the LPR when cloud user i bids zero.

Decomposing the Fractional Solutions

- $\mathbf{x}^* = (0.8, 0.6, 0.4)$
- Scale down $\mathbf{x}$ by 2: $(0.4, 0.3, 0.2)$
- $x(1) = (1, 0, 1), \beta_1 = 0.15$
- $x(2) = (1, 1, 0), \beta_2 = 0.25$
- $x(3) = (0, 1, 1), \beta_3 = 0.05$
- $x(4) = (0, 0, 0), \beta_4 = 0.55$

Decomposing the Fractional Solutions

We next decompose $\mathbf{x}^*$ into a *convex combination* of integer solutions, using a LP duality based decomposition technique for packing type of optimization problems due to Carr *et al.* and Lavi *et al.*

Our goal is to find β_I and $\mathbf{x}(I)$ such that $\mathbf{x}^*/\alpha = \sum_{I \in \mathcal{I}} \beta_I \mathbf{x}(I)$, where $\mathbb{Z}(DP) = \{\mathbf{x}(I)\}_{I \in \mathcal{I}}$ is the set of integer solutions to IP (4), $\mathcal{I}$ is the index set, and $\beta_I \geq 0, \sum_{I \in \mathcal{I}} \beta_I = 1$. Since the integrality gap is at most α, there exists at least one integer solution, *e.g.*, $DP(\mathcal{B})^*$, dominating the scaled down fractional solution. Consequently, scaling down the fractional solution $\mathbf{x}^*$ by α can guarantee the existence of such a decomposition.

The following primal and dual LPs are solved for decomposing $\mathbf{x}^*$:

A pair of tailored primal and dual LPs

$$\text{Primal:} \qquad \text{minimize} \quad \sum_{l \in \mathcal{I}} \beta_l \tag{6}$$

subject to:

$$\sum_{l \in \mathcal{I}} \beta_l x_i(\mathcal{S}, l) = x_i^*(\mathcal{S})/\alpha \quad \forall i \in \mathcal{B}, \mathcal{S} \in B_i \tag{6a}$$

$$\sum_{l \in \mathcal{I}} \beta_l \geq 1 \tag{6b}$$

$$\beta_l \geq 0 \qquad \forall l \in \mathcal{I} \tag{6c}$$

A pair of tailored primal and dual LPs

$$\text{Primal:} \quad \text{minimize} \quad \sum_{I \in \mathcal{I}} \beta_I \tag{6}$$

subject to:

$$\sum_{I \in \mathcal{I}} \beta_I x_i(\mathcal{S}, I) = x_i^*(\mathcal{S})/\alpha \quad \forall i \in \mathcal{B}, \mathcal{S} \in \mathcal{B}_i \tag{6a}$$

$$\sum_{I \in \mathcal{I}} \beta_I \geq 1 \tag{6b}$$

$$\beta_I \geq 0 \qquad \forall I \in \mathcal{I} \tag{6c}$$

$\Updownarrow$

Primal	(6a)	(6b)	β
Dual	ν	λ	(7a)

A pair of tailored primal and dual LPs

$$\text{Primal:} \quad \text{minimize} \quad \sum_{I \in \mathcal{I}} \beta_I \tag{6}$$

subject to:

$$\sum_{I \in \mathcal{I}} \beta_I x_i(\mathcal{S}, I) = x_i^*(\mathcal{S})/\alpha \quad \forall i \in \mathcal{B}, \mathcal{S} \in B_i \tag{6a}$$

$$\sum_{I \in \mathcal{I}} \beta_I \geq 1 \tag{6b}$$

$$\beta_I \geq 0 \qquad \forall I \in \mathcal{I} \tag{6c}$$

$\Updownarrow$

Primal	(6a)	(6b)	β
Dual	ν	λ	(7a)

$$\text{Dual:} \quad \text{maximize} \quad \frac{1}{\alpha} \sum_{i \in \mathcal{B}, \mathcal{S} \in B_i} x_i^*(\mathcal{S}) \nu_i(\mathcal{S}) + \lambda \tag{7}$$

subject to:

$$\sum_{i \in \mathcal{B}, \mathcal{S} \in B_i} x_i(\mathcal{S}, I) \nu_i(\mathcal{S}) + \lambda \leq 1 \qquad \forall I \in \mathcal{I} \tag{7a}$$

$$\lambda \geq 0 \tag{7b}$$

$$\nu_i(\mathcal{S}) \text{ unconstrained} \quad \forall i \in \mathcal{B}, \mathcal{S} \in B_i \tag{7c}$$

A pair of tailored primal and dual LPs

The primal decomposition LP has an exponential number of variables. We resort to the dual. Even though the dual (7) has an exponential number of constraints, the ellipsoid method can be applied to solve it in polynomial-time, with the cooperative algorithm acting as a separation oracle for generating separating hyperplanes for the dual.

Once an optimal dual solution is obtained, using a polynomial number of hyperplanes, the primal (6) can be converted to an optimization problem with a polynomial number of constraints corresponding to these hyperplanes. As a result, the convex decomposition can be solved within polynomial time.

A pair of tailored primal and dual LPs

However $\nu_i(\mathcal{S})$ may be negative, making the cooperative algorithm work improperly. Instead of using $\nu_i(\mathcal{S})$ directly, we set $\nu_i(\mathcal{S})^+ = \max(\nu_i(\mathcal{S}), 0)$ to circumvent this issue. IP (4) satisfies the nice *packing* property, *i.e.*, if $a \in \mathbb{Z}(DP), b \leq a$ then $b \in \mathbb{Z}(DP)$. Using the packing property, the following lemma ensures that using $\nu_i(\mathcal{S})^+$ does not violate the constraints in the dual (7).

Lemma 4

Given an integer solution $\mathbf{x}' \in \mathbb{Z}(DP)$*, we can obtain* $\mathbf{x}(I)$ *so that* $\sum_{i\in\mathcal{B},\mathcal{S}\in\mathcal{B}_i} x'_i(\mathcal{S}, I)\nu_i(\mathcal{S})^+ = \sum_{i\in\mathcal{B},\mathcal{S}\in\mathcal{B}_i} x_i(\mathcal{S}, I)\nu_i(\mathcal{S})$.

Lemma 5

If $\boldsymbol{\beta}^*$ *is an optimal solution to the primal (6), then* $\sum_{I\in\mathcal{I}} \beta^*_I = 1$.

The Randomized Auction Mechanism

Given the convex decomposition $\mathbf{x}^*/\alpha = \sum_{I \in \mathcal{I}} \beta_I \mathbf{x}(I)$, as shown in Algorithm 3, we select each valid integer solution $\mathbf{x}(I)$ randomly with probability β_I, and set the prices $\Pi_i = \Pi_i^F/\alpha$.

Theorem 6

The randomized auction in Algorithm 3 is truthful in expectation, and achieves the same α-approximation in social welfare as the the cooperative algorithm does.

Proof: The expected utility of a given bidder i is:

$$u_i(\sum_{I \in \mathcal{I}} \beta_I \mathbf{x}(I)) - \Pi_i = u_i(\mathbf{x}^*/\alpha) - \Pi_i^F/\alpha = (u_i(\mathbf{x}^*) - \Pi_i^F)/\alpha$$

The second equality is due to the linearity of $u_i(\mathbf{x})$. This means the expected utility is scaled down by α from the utility in the fractional VCG auction. Truthfulness of the randomized auction thus follows from that of the fractional VCG auction.

Conclusion

• Focusing on dynamic resource provisioning and heterogeneous types of VMs, we first propose a cooperative primal dual approximation algorithm with a small approximation ratio.

• Employing the cooperative approximation algorithm as a building block, we then design a novel randomized auction using a pair of tailed primal and dual LPs to decompose an optimal fractional solution into a summation of a series of weighted valid integer solutions.

• The randomized auction achieves the same approximation ratio in social welfare as the cooperative algorithm does.