When Cloud Meets eBay: Towards Effective Pricing for Cloud Computing

Shi Weijie

November 9, 2012

Contributions

- A computationally efficient and truthful auction mechanism
- A computationally efficient and truthful auction mechanism against a small-size collusion group

Why auction?

- A good pricing mechanism can increase the revenue of both the cloud provider and the users
- Differential pricing can further expand the market size and the revenue
- Auction is the fastest and most direct way to reflect the market trend

The Weakness of Spot Instance

- Not truthful
- User cannot bid for a bundle of computing resources
- Uniform pricing

Vickrey auction

- One item, several bidders
- Each user has a valuation for the item (secret)
- Highest bidder wins, and pay the second-highest bid.
- Truthful and efficient

User	Bid	Pay	Utility
A	\$10	\$8	\$2
B	\$8	\$0	\$0
C	\$6	\$0	\$0

VCG auction

- Multiple items, several bidders bid on a bundle of items
- Allocation: Maximize the sum of winners' bidding. (Social optimal)
- Payment: Charge each winner the opportunity cost that he brings to other bidders.
- Efficient and Truthful
- But not time efficient! (NP-Complete)

User	Bid	Pay	Utility
A	\$5 for Item1	6-2 = \$4	5-4 = \$1
B	\$2 for Item2	6-5 = \$1	2-1 = \$1
C	\$6 for both	\$0	\$0

Greedy (approximate) allocation in VCG auction

- Greedy standard: Bid/$\sqrt{\text{Weight}}$. Here we suppose each item has the same weight.
- Time efficient.
- But...

User	Bid	Win item	Pay	Utility
A	\$13 for I_1 & I_2	Nothing	\$0	\$0
B	\$10 for I_1	I_1	21-8 = \$13	\$-3
C	\$8 for I_3	I_3	10-10 = \$0	\$8

B becomes a liar and gains utility

- B now bids \$9
- VCG payment scheme together with a non-optimal allocation is not a truthful auction.

User	Bid	Win item	Pay	Utility
A	\$13 for I_1 & I_2	I_1 & I_2	17-8=\$9	\$4
B	\$9 for I_1	Nothing	\$0	\$0
C	\$8 for I_3	I_3	13-13 = \$0	\$8

Model

- m types of instance
- k_i available units for instance type i
- n users
- User j bid for a bundle $S_j = \{k_j^1, k_j^2, \ldots, k_j^m\} \in S = \{0, \ldots, k_1\} \times \{0, \ldots, k_2\} \times \{0, \ldots, k_m\}$
- User j bid at per unit price $B_j = \{b_j^1, \ldots, b_j^m\}$
- User j's total bid $\overline{B_j} = \sum_i k_j^i b_j^i$
- User j's total valuation $V_j = \sum_i k_j^i v_j^i$
- Payment P_j
- Utility $U_j = V_j - P_j$

Weight

- Different types of instance have different importance: w_i
- VM with more CPU and more memory has the higher weight.
- Input by the cloud provider to make the approximation more accurate

Rank

- Rank by average price per weighted instance
- $$\frac{\overline{B_j}}{\sqrt{\sum_i k_j^i w_i}}, \text{ for } j \in 1, \ldots, n$$
- Select winner in ranked order

Allocation Algorithm

Algorithm 1 An computationally efficient and truthful mechanism for cloud service auction

Input: $\mathcal{S}, \mathcal{B}, \mathcal{W}$

Output: $\mathcal{X}, \mathcal{P}$

```
Calculate $\overline{\mathcal{B}}$ based on $\mathcal{B}$
Sort bids in $\overline{\mathcal{B}}$ such that $\frac{\overline{\mathcal{B}}_1}{\sqrt{\sum_{i=1}^{m} k_1^i w_i}} \geq \ldots \geq \frac{\overline{\mathcal{B}}_n}{\sqrt{\sum_{i=1}^{m} k_n^i w_i}}$
Initialize $x_1 = \cdots = x_n = 0$   %greedy allocation
for $j = 1, \ldots, n$ do
    if $\forall q \in \{1, \ldots, m\}, \sum_{t=1}^{j-1} k_t^q x_t + k_j^q \leq k_q$ then
        $x_j = 1$
    end if
```

Allocation Approximation Ratio

Theorem 2: Algorithm 1 with criterion $\overline{\mathcal{B}}_j/\sqrt{\sum_{i=1}^{m} k_j^i w_i}$ approximates the optimal solution with a factor of $\sqrt{\frac{w_{max}}{w_{min}} \sum_{i=1}^{m} k_i}$.

Critical value

- The minimum price v_c that bidder j should bid in order to win.
- If $\overline{B_j} < v_c$, then bidder j loses.
- Critical bidder: The highest-rank bidder who is eliminated by bidder j

Pay the critical value

- Find the critical bidder, and pay his bid.
- Necessary condition for a truthful auction
- Losers will not lie
- Winners' pay is decided by a critical loser

Compute the payment

- According to the order of the rank, after the bidder j, the first bidder eliminated by bidder j. (Say bidder s)
- Compute the payment :

$$P_j = \frac{\overline{B_s}\sqrt{\sum_i k_j^i w_i}}{\sqrt{\sum_i k_s^i w_i}}$$

Payment Scheme

Initialize $\mathcal{P}_1 = \mathcal{P}_2 = \ldots = \mathcal{P}_n = 0$ *%payment*

for $j = 1, \ldots, n$ **do**

  if $x_j = 1$ **then**

    $\overline{\mathrm{INS}_1} = \sum_{t=1}^{j-1} k_t^1 x_t, \ldots, \overline{\mathrm{INS}_m} = \sum_{t=1}^{j-1} k_t^m x_t$

    for $s = j+1, \ldots, n$ **do**

      if $\forall q \in \{1, \ldots, m\}, \overline{\mathrm{INS}_q} + k_s^q \leq k_q$ **then**

        $\overline{\mathrm{INS}_1} = \overline{\mathrm{INS}_1} + k_s^1, \ldots, \overline{\mathrm{INS}_m} = \overline{\mathrm{INS}_m} + k_s^m$

        if for some $q \in \{1, \ldots, m\}$, $\overline{\mathrm{INS}_q} + k_j^q > k_q$ **then**

          $\mathcal{P}_j = \dfrac{\overline{\mathcal{B}}_s \sqrt{\sum_{i=1}^m k_j^i w_i}}{\sqrt{\sum_{i=1}^m k_s^i w_i}}$

          break;

        end if

      end if

Figure A & B

- Fix the number of different instances ($m = 20$)
- Revenue: the total income of the cloud provider
- User satisfaction: the percentage of winning users

Figure A & B

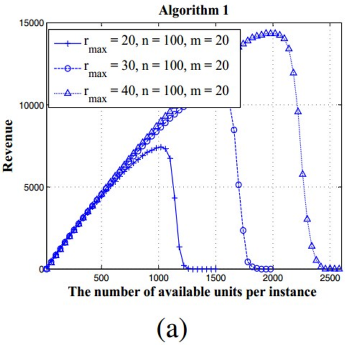
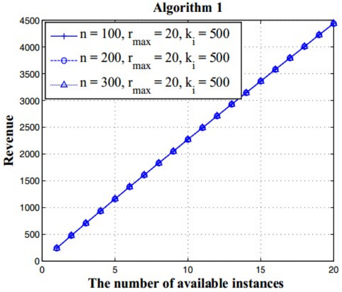

Algorithm 1
Revenue
15000
10000
5000
0
500
1000
1500
2000
2500
The number of available units per instance
r max = 20, n = 100, m = 20
r max = 30, n = 100, m = 20
r max = 40, n = 100, m = 20
(a)
Algorithm 1
User satisfaction
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
500
1000
1500
2000
2500
The number of available units per instance
r max = 20, n = 100, m = 20
r max = 30, n = 100, m = 20
r max = 40, n = 100, m = 20
(b)

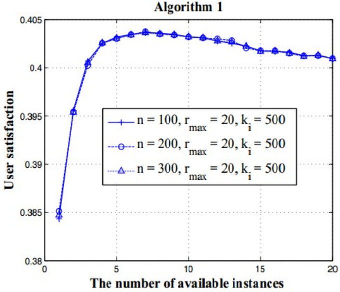
Figure A & B

- Fix the number of total units ($\sum_i k = 500$)
- Revenue: the total income of the cloud provider
- User satisfaction: the percentage of winning users

Figure C & D


Algorithm 1
Revenue
The number of available instances
n = 100, r max = 20, k i = 500
n = 200, r max = 20, k i = 500
n = 300, r max = 20, k i = 500

(c)


Algorithm 1
User satisfaction
The number of available instances
n = 100, r max = 20, k i = 500
n = 200, r max = 20, k i = 500
n = 300, r max = 20, k i = 500

(d)