# Ex-post Efficient Resource Allocation for Self-organizing Cloud

Sheng Di[a], Cho-Li Wang[a], Ling Chen[b]

[a] *Department of Computer Science, The University of Hong Kong, Pokfulam Road, Hong Kong*
[b] *Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China*

---

**Abstract**

Self-organizing Cloud (SoC) is a very scalable model, which aims to make full use of the geographically distributed computers to provide powerful aggregated computing ability. The resources are provisioned elastically according to user's specific demand, by leveraging virtual machine (VM) resource isolation technology. The resource allocation atop SoC is very challenging in that it is not only an optimization problem over multi-dimensional divisible resources but closely related to economic analysis for coordinating the social competitions. Existing economy-based approaches, however, are commonly ex-ante efficient, such that the participants' decisions may rely on their subjective judgments about their competitors' decisions, leading to the unguaranteed payoffs. Instead, we propose an ex-post efficient resource allocation scheme, which owns three features. (1) Ex-post win-win effect: under such a scheme, each participant (including consumers and suppliers) will always feel satisfied with their ex-post payoffs. (2) Ex-post incentive compatibility: we can theoretically prove each rational consumer and contributor will always get the optimal payoff if and only if their resource demands and expected prices are truthfully declared. (3) Ex-post maximized efficiency: truly advanced (or more powerful) resources will be allocated and consumed with higher priority, such that the whole system will stay very efficient with maximized resource utilization. Via simulation, our approach can significantly improve the resource contributor's payoffs, also with a high quality of service throughout the global system (including user task's execution efficiency, overall utility, payment level, and fairness of treatment, etc.).

*Keywords:* Cloud Computing, Convex optimization, Divisible Resource Allocation, Vickrey Auction, Ex-post Optimality

---

## 1. Introduction

Cloud computing offers scalable on-demand virtualized resources as a flexible service over the Internet with bypassed inter-operability constraints. With VM's resource isolation technology [1, 2], computing resources such as CPU and memory could be partitioned and reassembled to meet end-users' specific needs, achieving elastic and convenient access to the virtualized computational resources [3, 4]. On the other hand, volunteer computing (or P2P desktop Grid) has been studied for years especially for its great potential in millions of computers throughout the world. Such platforms (e.g., BOINC [5], XtremWeb [6]) have made great contributions to scientific researches since 2000.

Our extended Cloud framework (a.k.a., Self-organizing Cloud) combines resource isolation technology of Cloud computing and self-organizing architecture in volunteer computing together. In the Self-organizing Cloud, each host (either a volunteer desktop computer or a dedicated computing node behind cluster) is deployed with an autonomous resource state collector and a virtual machine monitor (VMM), being able to act as both task scheduler and resource contributor (or supplier). A task could be a user request to customize a particular execution environment with specified resource demand, which is expressed as a least-qualified vector (e.g., including CPU, memory and network bandwidth). In such a framework, the participants will have high motivations to contribute their resources. This is because any resource contributor will earn an amount of payment/income, which benefit them in return (e.g., getting more resources later) based on the relation between motivation and market [7].

Recently, there already exist many projects being designed based on this framework. A typical example is an on-going project (namely Cloud@Home [8]) undertaken by INRIA. It mainly aims to devise a set of strategies for checkpointing applications using VMs and guarantee high data durability, availability, and access performance. Another example is Community Cloud [9], in which each participating host is also considered the resource supplier and the whole system is organized based on the principle of digital ecosystem. Wuala Cloud [10] is a fully distributed online storage system that can make use of the disk space and network bandwidth from the distributed volunteer desktop computers, to provision flexible and scalable management of large-capacity storage.

Based on the above real-world cases, designing a Self-organizing Cloud (SoC) system can benefit a lot. (1) The resource utilization could be improved with finer-granularity resource allocation over VM technology. (2) More geographically distributed idle resources can be used while the substrate details are still transparent to users as if in a single-point-of-access manner. (3) High robustness and reliability can be guaranteed by minimizing the impact of single node's failure or malicious user's DDoS attack; (4) There would be less cost on management and maintenance than that of traditional vender clouds, by leveraging the autonomous marketing mechanism between resource consumers and contributors.

In addition to the opportunities mentioned above, however, the self-organizing architecture will introduce some challenges, especially when aiming to achieve the ex-post efficiency (a.k.a., ex-post optimality) [11, 12, 13]. In a nut shell, ex-post efficiency means that the consumers (provider) would never be regretful about their previous decisions on their resource declaration. That is, the participants' decisions will not depend on other competitors' decisions, so that each of them will always be satisfied with the results, even after finding out the winning bids of other bidders. In this paper, we strictly define the ex-post efficiency as the situation with the following features:

- Ex-post Win-win Effect: After task execution and resource consumption, each participant (including consumers and suppliers) will always feel satisfied with their ex-post payoffs. Ex-post win-win effect is significant to the long-term high availability and high resource utilization of the system. However, since there are no central-controlled servers to coordinate the global resource states and prices and each autonomous participant cannot derive other participants' decisions, it is non-trivial to guarantee each autonomous consumer to get satisfied on the task schedule or contributor to be satisfactory on the payoff of resource contribution.

- Ex-post Incentive Compatibility: After task execution and resource consumption, each rational consumer and contributor will always get the optimal payoff if and only if their resource demands and expected prices are truthfully declared. Ex-post incentive compatibility is critical to the guarantee of the Quality of Service (QoS) as well as the system resource utilization. Without a robust pricing policy, rational participants may tend to lie on their real demands in order to maximize their selfish gains, finally weakening other ones' motivations. What is most serious is that the whole system resource availability may be prominently degraded eventually.

- Ex-post Maximized Execution Efficiency: After task execution and resource consumption, powerful resources will be allocated and consumed with a higher priority, such that the whole system will stay efficient with the maximized resource utilization.

The remainder of the paper is organized as follows. In Section 2, we formulate the fully-distributed cloud resource allocation problem. In Section 3, we formally propose our solution, which contains three key steps: (1) how self-organizing nodes discover resources and collects volatile states in brief; (2) how to design the Double-sided Vickrey Auction algorithm, by combining the ex-post win-win effect and incentive compatibility together; (3) how to split the resources using the VM resource isolation technology by running a task on a specific execution node. Section 4 shows our simulation result. Our work is carefully compared to the related works, by discussing the traditional economy-based allocation adopted in Grid/Cloud systems in Section 5. Finally, we conclude the paper with the future work in Section 6.

## 2. Problem Formulation and Analysis

Without loss of generality, we assume that there are $n$ peer nodes, denoted by $p_i$ ($1 \leq i \leq n$). Each node serves as both resource consumer (or user) and resource provider (or contributor). Each node's resource is multi-dimensional along $D$ different attribute types (such as CPU, disk IO, network bandwidth, etc.). All the tasks submitted to $p_i$ are marked as $t_{ij}$ ($1 \leq j \leq n_i$), where $n_i$ indicates the total number of tasks submitted to $p_i$. We use $t_{ij}^{p_e}$ to indicate node $p_e$ is $t_{ij}$'s execution node. Let $c_k(p_e)$ ($1 \leq e \leq n$) denote the capacity of the $k$th resource attribute on node $p_e$ and $r(t_{ij}) = (r_1(t_{ij}), r_2(t_{ij}), \cdots, r_D(t_{ij}))^T$ denote the actual amounts of resource allocated to $t_{ij}$ when it is executed. Users need to specify an expected resource vector $e(t_{ij})$=$(e_1(t_{ij}), e_2(t_{ij}), \cdots, e_D(t_{ij}))^T$, where $e_k(t_{ij})$ is the least qualified amount of resource at $k$th attribute demanded by task $t_{ij}$, to complete its execution within an expected execution time $t_e$. Hence, two necessary conditions for the successful completion of any task $t_{ij}$ are Inequality (1) and Inequality (2).

$$\textstyle\sum_{i,j} e_k(t_{ij}^{p_e}) \leq c_k(p_e), \quad k = 1, 2, \cdots, D \tag{1}$$

$$e_k(t_{ij}) \leq r_k(t_{ij}^{p_e}), \quad k = 1, 2, \cdots, D \tag{2}$$

Moreover, each task $t_{ij}$ has multi-dimensional workloads on different resource attributes (such as computational workload over CPU, I/O data to read/write, etc.) to process, and its workload vector is denoted as $l(t_{ij})$=$(l_1(t_{ij}), l_2(t_{ij}), \cdots, l_D(t_{ij}))^T$. By considering the worst case for the task's execution, the execution along different dimensions will not overlap, thus the execution time (denoted $t(\boldsymbol{r}(t_{ij}))$) can be formulated as Equation (3).

$$t_e(\boldsymbol{r}(t_{ij}, p_e)) = \sum_{k=1}^{D} \frac{l_k(t_{ij})}{r_k(t_{ij})} \tag{3}$$

A node $p_e$'s availability state is denoted as a vector $a(p_e) = (a_1(p_e), a_2(p_e), \cdots, a_D(p_e))^T$, where $a_k(p_e){=}c_k(p_e){-}\sum_{\forall i,j} r_k(t_{ij}^{p_e})$. Nodes' availability states will be dynamically propagated using the multi-dimensional resource discovery protocol, namely Proactive Index Diffusion CAN (PID-CAN), which appears in our previous work [14].

It is likely that different tasks own various characteristics, e.g. CPU-bound or IO-bound properties. We use the workload ratio (denoted $l_1(t_{ij}){:}l_2(t_{ij}){:}\cdots{:}l_D(t_{ij}))^T$) to describe the execution property for each task, which can be predicted based on historical or statistical execution records [15, 16] or analysis of their intrinsic programming structures [17] in practice.

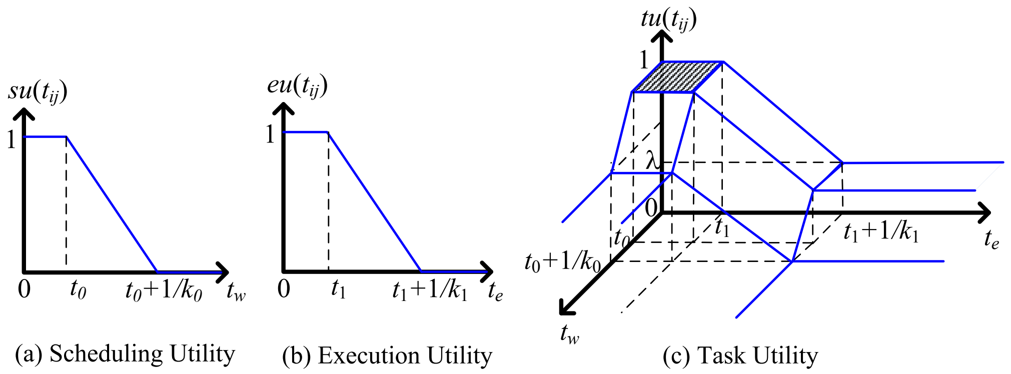
In this paper, we not only focus on how to optimize the task's execution efficiency by constructing the virtual machine (VM) with the resource shares split from execution node, but also study how to cope with the social competition relations among resource consumers and contributors, such that each of them will be satisfied with its *payoff*. Since a task's turnaround time could be split to two phases, scheduling period (or waiting/queuing time) and execution period (or running time), we define the task utility as Equation (4), where $su(t_{ij})$ and $eu(t_{ij})$ refer to the scheduling utility and execution utility respectively and $\lambda_{ij}$ is a coefficient customized based on user's expectation.

$$tu(t_{ij}) = \lambda_{ij} \cdot su(t_{ij}) + (1 - \lambda_{ij}) \cdot eu(t_{ij}) \tag{4}$$

Without loss of generality, we define $su(t_{ij})$ and $eu(t_{ij})$ as two piecewise linear functions which decay linearly over time (either waiting/queuing time $t_w$ or execution time $t_e$), as shown in Formula (5) and Formula (6) respectively, which are also shown in Figure 1 (a) and Figure 1 (b) graphically. Figure 1 (c) clearly illustrates the synthetic task utility: the shaded area represents a completely content status, and it starts declining at different rates along different planes, when increasing the expected waiting time $t_w$ or the expected execution time $t_e$.

$$su(t_{ij}) = \begin{cases} 1 & t_w \leq t_0 \\ 1 - k_0(t_w - t_0) & t_0 < t_w \leq t_0 + 1/k_0 \\ 0 & t_w > t_0 + 1/k_0 \end{cases} \tag{5}$$

$$eu(t_{ij}) = \begin{cases} 1 & t_e \le t_1 \\ 1 - k_1(t_e - t_1) & t_1 < t_e \le t_1 + 1/k_1 \\ 0 & t_e > t_1 + 1/k_1 \end{cases} \tag{6}$$



(a) Scheduling Utility (b) Execution Utility (c) Task Utility

Figure 1: Graphical Illustration of Utility Functions

In the two equations, $t_0$ and $t_1$ are users' expected queuing time and expected execution time respectively. The slopes of both linear functions (either $k_0$ or $k_1$) reflect user's patience on task scheduling and task's execution time. Obviously, $t_0+\frac{1}{k_0}$ and $t_1+\frac{1}{k_1}$ can be considered *least tolerable queuing time* and *least tolerable execution time* respectively. Moreover, we define $t_1$ in Definition 1, which will be used later to prove resource consumer's incentive compatibility. According to the definition of expectation vector $\boldsymbol{e}(t_{ij})$, we can get Equation (7).

$$t_1 + \frac{1}{k_1} = \sum_{k=1}^{D} \frac{l_k(t_{ij})}{e_k(t_{ij})} \tag{7}$$

**Definition 1.** Without loss of generality, $t_1$ is set as the shortest execution time by running task $t_{ij}$ over any of the candidate resource nodes found by PID-CAN protocol[1], i.e. $\min_{p_e \in QSET(t_{ij})} t_e(r^*(t_{ij}^{p_e}))$, where $QSET(t_{ij})$ is the candidate qualified nodes queried based on $\boldsymbol{e}(t_{ij})$.

When a task is finished by an execution node, its user should pay an amount of money for this node. Any resource provider would like to maximize its revenues,

[1]PID-CAN [14] will be discussed later in Section 3.2.

measured as the sum of the consumers' payments. Different nodes may regard their owned resources to be of different values based on their various needs on money and nodes' various properties or available states, thus they may assign various prices for their multiple types of resources, or via a pricing function of the demanded resource amounts. We use $b(p_e)$ to denote the price vector assigned by the owner of the resource node $p_e$.

From provider's point of view, the expected per-time-unit payment (denoted as $EPU(t_{ij})$) for a task's execution by its user is denoted by Formula (8).

$$EPU(t_{ij}^{p_e}) = r^*(t_{ij})^T \cdot b(p_e) \tag{8}$$

Then, the provider's expected payment (or income, denoted as *EP*$(t_{ij}^{p_e})$) in executing this task can be calculated by Formula (9), where $t_e(t_{ij})$ is referred to as task $t_{ij}$'s execution time. That is, the real payment amount (denoted by $RP(t_{ij}^{p_e})$ by the consumer should be no less than $EP(t_{ij}^{p_e})$.

$$EP(t_{ij}^{p_e}) = EPU(t_{ij}) \times t_e(t_{ij}) \tag{9}$$

From the perspective of resource consumers, they also have expected budgets for their tasks' execution, to make sure that the cost is affordable to them. Based on the expectation vector $e(t_{ij})$ and their expected price vector $\beta(t_{ij})$ on different resource attributes, they could easily estimate their own least-required per-time-unit payment (a.k.a., demanded per-time-unit budget, denoted as *DB*$(t_{ij})$) based on their evaluation, as shown in Formula (10).

$$DB(t_{ij}) = r^*(t_{ij})^T \cdot \beta(t_{ij}) \tag{10}$$

Then, each consumer needs to set a firm budget (denoted as $B(t_{ij})$) for its task $t_{ij}$, which means that the real per-time-unit payment amount on the task must be no greater than this budget. Obviously, $B(t_{ij}) \geq db(t_{ij}^{p_e})$ at any time. In our design, the firm budget $B(t_{ij})$ will also serve as the scheduling bid of users, to compete for the scheduling priority among submitted tasks on the same scheduler node. That is, higher budget implies the cost the user is willing to pay more on its task's execution and on being scheduled with higher priority, resulting in shorter queuing and execution time.

The final real per-time-unit price (denoted *RPU*$(t_{ij}^{p_e})$) will be synthetically settled based on the auction bid (a.k.a., auction-based budget, denoted as *AB*$(t_{ij})$) among consumers and the auction payment unit (denoted as *APU*$(t_{ij})$) that is dependent upon candidate resource owners' declared prices (i.e., $EPU(t_{ij}^{p_e})$). The auction bid is determined by different auction policies. For example, assuming $t_{ij}$ wins the bid (i.e., it gives the highest firm budget among all other competitors), *AB*($t_{ij}$) is equal to *B*($t_{ij}$) under the first-price sealed-bid policy, while *AB*($t_{ij}$)=$\max_{\{x,y\}\neq\{i,j\}}(B(t_{xy}))$ in the second-price sealed-bid policy. The details of how to determine *RPU*($t_{ij}^{p_e}$ will be described in Section 3.1.

Once the task is scheduled and completed successfully, its user needs to pay the execution node's owner the amount of currencies calculated based on the *RPU*($t_{ij}^{p_e}$. The main objective of such a definition on user's payment is to guarantee the satisfaction on the payment from both sides (resource consumers and suppliers), which serves as one condition of the ex-post win-win effect of our system.

The total payment (denoted as *TP*($t_{ij}$)) of this user on this task is the product of the real per-time-unit price and the execution time, as shown in Formula (11).

$$TP(t_{ij}) = RPU(t_{ij}) \times t_e(t_{ij}) \tag{11}$$

In practice, either the system or the users themselves may not accurately predict the execution times for the tasks. For example, any user may pay more than the original payment amount previously estimated on their own due to the inevitable error-prone prediction on the task's execution time. In this situation, the users would still feel worthy as long as the per-time-unit payments are still under their agreement. That is, the execution of each task $t_{ij}^{p_e}$ is acceptable as long as its real payment cost per time unit (i.e., $RPU(t_{ij}^{p_e})$) is in accordance with Inequality (12), where $B(t_{ij})$ is the user's defined firm per-time-unit budget.

$$RPU(t_{ij}) \leq B(t_{ij}) \tag{12}$$

We define $P_{cs}^{sch}(t_{ij})$ (Equation (13)) and $P_{cs}^{exe}(t_{ij})$ (Equation (14)) as the payoffs of $t_{ij}$'s user (i.e., resource consumer), respectively at task scheduling phase and task's execution duration. In these two equations, $\nu_{cs}^{sch}(t_{ij})$ refers to the true valuation (or private valuation) of $t_{ij}$'s scheduling priority and $\nu_{cs}^{exe}(t_{ij})$ refers to the final valuation of task's execution, regarded by the resource consumer. The reason why $P_{cs}^{sch}(t_{ij})$ is 0 whenever $sp(t_{ij}) \geq \nu_{cs}^{sch}(t_{ij})$ is that $\nu_{cs}^{sch}(t_{ij})$ refers to the true valuation of scheduling priority regarded by its user. In other words, if the user cannot win the scheduling priority among all other tasks in queue based on his/her own scheduling bid, he/she should not suffer any loss from his/her point of view. Moreover, without loss of generality, $\nu_{cs}^{exe}(t_{ij})$ could be considered a function whose value is proportional to the execution utility (i.e., *eu*($t_{ij}$)), and equal to 0 when *eu*($t_{ij}$)=0.

$$P_{cs}^{sch}(t_{ij}) = \begin{cases} \nu_{cs}^{sch}(t_{ij}) - AB(t_{ij}) \cdot t_e(t_{ij}), & AB(t_{ij}) \cdot t_e(t_{ij}) < \nu_{cs}^{sch}(t_{ij}) \\ 0, & otherwise \end{cases} \tag{13}$$

$$P_{cs}^{exe}(t_{ij}) = \begin{cases} \nu_{cs}(t_{ij}) - EP(t_{ij}^{p_e}), & eu(t_{ij})\ is\ maximized\ with\ B(t_{ij}) \\ & budget\ constraint\ \&\ EP(t_{ij}^{p_e}) < \nu_{cs}^{exe}(t_{ij}) \\ 0 & ,\ otherwise \end{cases} \tag{14}$$

**Definition 2.** Rational consumer's expected result is that the payoffs of his task scheduling and execution are both maximized, based on its unilateral bidding.

We denote $P_{ct}(t_{ij}^{p_e})$ as the contributor's payoff of running $t_{ij}$ on the node $p_e$, which conforms to the Formula (15).

$$P_{ct}(t_{ij}^{p_e}) = \begin{cases} TP(t_{ij}) - \nu_{ct}(t_{ij}), & TP(t_{ij}) > \nu_{ct}(t_{ij}) \\ 0, & otherwise \end{cases} \tag{15}$$

The total payoff of the contributor $p_e$ can be expressed as Formula (16), where $t_{ij}^{p_e}$ denotes the tasks that are executed on $p_e$.

$$P_{ct}(p_e) = \sum_{t_{ij}^{p_e}\ executed\ by\ p_e} P_{ct}(t_{ij}^{p_e}) \tag{16}$$

**Definition 3.** Rational resource contributor's expected result is that the payoff of its resource contribution is maximized, based on its unilateral bidding.

The ultimate objective of our design is to achieve the ex-post efficiency (a.k.a., ex-post optimality) of the whole system, through the fully decentralized self-organizing resource allocation. Specifically, the three Inequalities (17), (18), and (19) should always hold for each participant, where $P_{cs}^{sch}(t_{ij})$, $P_{cs}^{exe}(t_{ij})$, and $P_{ct}(p_e)$ refer to the payoffs under our designed resource allocation scheme (also with participants' true valuations declared) and $P_{cs}^{sch'}(t_{ij})$, $P_{cs}^{exe'}(t_{ij})$, and $P'_{ct}(p_e)$ denotes the payoffs gained under any other approaches (also including the situation with participants' fake/lying valuations declared).

$$P_{cs}^{sch'}(t_{ij}) \leq P_{cs}^{sch}(t_{ij}) \tag{17}$$

$$P_{cs}^{exe'}(t_{ij}) \leq P_{cs}^{exe}(t_{ij}) \tag{18}$$

$$P'_{ct}(p_e) \leq P_{ct}(p_e) \tag{19}$$

For readability, we summarize the key notations used in the problem formulation and the following analysis in Appendix A (Table A.3).

## 3. Ex-post Efficient Resource Allocation for Self-organizing Cloud

Our goal is to design a distributed cloud model that can achieve the ex-post efficiency, i.e. the three Inequalities (17) and (19) for each participant. We will first present our core design skeleton of the ex-post efficient allocation algorithm, and then briefly describe the fully decentralized resource discovery protocol and the local optimal VM resource allocation (LOVRA) algorithm. Finally, we will prove the ex-post efficiency of our design and the incentive compatibility feature in theory.

### *3.1. Design Skeleton*

We show our core design (the skeleton algorithm) in Algorithm 1 (called Double-sided Vickrey Auction algorithm).

**Algorithm 1** DOUBLE-SIDED VICKREY AUCTION ALGORITHM

```
1: while (true) do
2:     Sort $q_e$'s tasks in non-increasing order of $B(t_{ij})$;
3:     for (each task $t_{ij}$ in $q_e$) do
4:         $\max_{B(t_{xy}) \leq B(t_{ij})}(B(t_{xy})) \rightarrow AB(t_{ij})$; /*$t_{xy}$ refers to other tasks in $q_e$ other than $t_{ij}$*/
5:         Perform PID-CAN to construct qualified node set $QSET(t_{ij})$ for $t_{ij}$;
6:         for (each item $p^*_{(k)}$ in $QSET(t_{ij})$) do
7:             Perform LOVRA algorithm to calculate $t_{ij}$'s optimal allocation ($r^*_{(k)}(t_{ij})$) on $p^*_{(k)}$;
8:             Estimate $p^*_{(k)}$'s expected payment (denoted as $EP^*_{(k)}$);
9:         end for
10:        Sort $QSET(t_{ij})$ in non-decreasing order of $t_{ij}$'s expected payment based on $r^*_{(k)}(t_{ij})$;
11:        for (each item $p^*_{(k)}$ in $QSET(t_{ij})$) do
12:            Connect $p^*_{(k)}$ to reconfirm its availability state; /*to avoid conflict of contention.*/
13:            if ($p^*_{(k)}$ is qualified for $t_{ij}$ on Formula (1) and (12)) then
14:                Determine $APU(t_{ij})$ based on Formula (20);
15:                Determine $RPU(t_{ij})$ with $AB(t_{ij})$ and $APU(t_{ij})$, based on Formula (21);
16:                Update $p^{(k)}_*$'s status;
17:                Execute $t_{ij}$ in VM atop $p^*_{(k)}$ based on $r^*_{(k)}$ outputted by LOVRA algorithm;
18:                break;
19:            end if
20:        end for
21:    end for
22:    Sleep a tiny cycle; /*to receive more tasks*/
23: end while
```

This algorithm should run on each individual nodes in the Self-organizing Cloud system. In this algorithm, $p^*_{(k)}$, $r^*_{(k)}$ and *RPU*$(t_{ij})$ stand for the $k$th qualified node record in *QSET*$(t_{ij})$ (qualified node set), the corresponding optimal resource share vector calculated by the Local Optimal VM Resource Allocation (LOVRA) algorithm (to be introduced in Section 3.3) and task $t_{ij}$'s final per-time-unit payment.

Without loss of generality, suppose it is running on a node $p_e$ as a scheduler. As mentioned previously, the node $p_e$ receives multiple tasks submitted by users over time, and all of them are put in the queue $q_e$, which also contains the old tasks that still have not found matched resources yet. The scheduler of $p_e$ will process $q_e$'s tasks according to the non-increasing order of $B(t_{ij})$ (i.e., firm budget), periodically (line 2).

Basically, there are three phases for processing each task (as shown in Figure 2 (a)): resource discovery, auction-based task scheduling, and resource allocation.

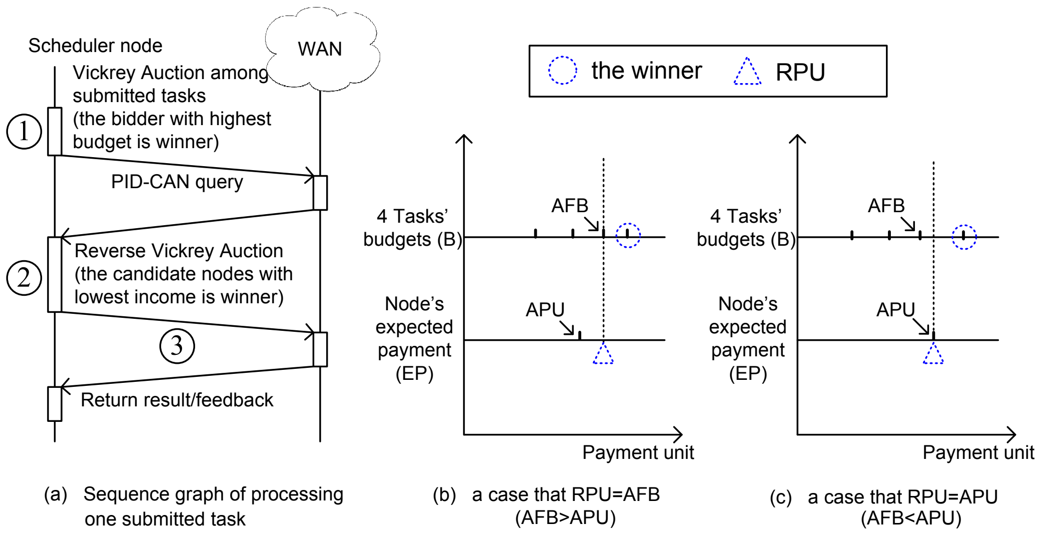

Scheduler node
WAN
Vickrey Auction among submitted tasks (the bidder with highest budget is winner)
1
PID-CAN query
Reverse Vickrey Auction (the candidate nodes with lowest income is winner)
2
3
Return result/feedback
the winner
RPU
4 Tasks' budgets (B)
AFB
Node's expected payment (EP)
APU
Payment unit
(a) Sequence graph of processing one submitted task
(b) a case that RPU=AFB (AFB>APU)
(c) a case that RPU=APU (AFB<APU)

Figure 2: Illustration for task scheduling phase and calculation of RPU

In the first phase (line 4∼5), we first adopt the classical Vickrey auction [18] (line 4) to decide $AB(t_{ij})$ as the second highest budget declared by the submitted tasks, which can discipline the truth bidding behaviors of rational resource consumers. And then, we use our previously designed Proactive Index Diffusion CAN (PID-CAN) [14] to find available resources (line 5) for any task based on its demand $e(t_{ij})$ around the global systems, with mitigated query contention among requesters. Specifically, a query message that contains $e(t_{ij})$ will be routed via the PID-CAN overlay, until it finds the specific number of qualified resource nodes through the passed duty nodes or the number of hops surpasses the time-to-live (TTL) threshold. The response messages received by the scheduler node $p_s$ will contain the queried resource nodes' identifiers (say IP address), their resource availability states, and the corresponding prices (or a pricing function).

At the second phase (line 6$\sim$15), one qualified resource node $p^*_{(k)}$ will be selected as the execution node. line 7 is used to perform our LOVRA algorithm to find the local optimal solution, which will be described in Section 3.3 in details. We We calculate *APU*$(t_{ij})$ and *RPU*$(t_{ij})$ at line 14 and line 15 respectively, based on Formula (20) and Formula (21). We use Figure 2 (b) and (c) to illustrate the calculation of *RPU*$(t_{ij})$, assuming there are 4 tasks submitted with different budgets.

$$APU(t_{ij}) = \frac{EP^*_{(k+1)}}{EP^*_{(k)}} \cdot EPU(t_{ij}^{p^*_{(k)}}) \tag{20}$$

$$RPU(t_{ij}) = \max(AB(t_{ij}), APU(t_{ij})) \tag{21}$$

In addition to the classical Vickrey auction used at0 line 4, we also exploit a reverse Vickrey auction to make sure that the resource contributors are also better off truthfully revealing their resources' prices (line 10). The basic rationale is that the resource nodes with lower prices will have higher priority to be selected (line 10$\sim$11), while $t_{ij}$'s payment will be settled based on the next lowest payment from among the set of candidate resources (i.e., reverse second-price policy). Hence, the resource contributors can always receive more payments than their original expectations since the auction-based per-time payment unit (APU) is calculated based on higher prices. On the other hand, the tasks can be finished within their preferred firm budgets ($B(t_{ij})$) and expected execution performance, so the consumers will also be satisfied. For the last phase (line 16$\sim$17), we use the $r^*_{(k)}$ calculated at line 7 (i.e., the optimal resource share based on LOVRA algorithm) as the resource vector for the task to be run on the selected node. In addition, through line 5 and line 7, it is obvious that more powerful resources with higher availability will be selected as the candidate resource nodes with higher likelihood, implying a higher working efficiency of the whole system.

## *3.2. Dynamic Decentralized Resource Discovery Protocol*

We first briefly introduce the resource discovery protocol, namely Proactive Index Diffusion CAN (PID-CAN) [14], which will be responsible for distributively aggregating state information on every node.

Like traditional CAN [19], each node (a.k.a., duty node) under PID-CAN is responsible for a globally unique *multi-dimensional range* zone randomly selected when it joins the overlay; nodes' update-states containing the availability vector and resource price vector will be periodically propagated to the duty node whose zone encloses the availability vector.

Unlike CAN, every node in PID-CAN connects a few more neighbors whose distances are $2^k$ ($k$=0,1,$\cdots$) hops; the identifer (a.k.a., index) of the duty node that has non-empty cache will be proactively diffused to a few randomly selected $2^k$-hop negative-direction neighbors. As a user submits a task $t_{ij}$, the corresponding scheduler node will perform range query for it, and the query message will be first routed to the zone-matched node (i.e., its first duty node). This duty node will then issue a multi-dimensional range query moving towards other duty nodes along the positive indexes hop by hop, to find more available resource records. Finally, several qualified nodes (e.g., $p_e$) satisfying the task's demanding vector and budget (i.e., Inequality (22) and (23)) will be returned to the scheduler node. Note that this design can effectively control the query traffic overhead because each query just launches single query message instead of multiple parallel ones.

$$e(t_{ij}) \leq a(p_e) \tag{22}$$

$$EPU(t_{ij}^{p_e}) \leq B(t_{ij}) \tag{23}$$

## *3.3. Local Optimal VM Resource Allocation (LOVRA)*

### *3.3.1. Problem Analysis*

As a task $t_{ij}$ is scheduled onto one resource node (denoted $p_e$), the local optimal VM resource allocation (LOVRA) will be performed: a vector of resource shares on multi-dimensional resource types will be constructed for this task. Then, the corresponding resource shares will be split from the physical available idle resources of $p_e$, by leveraging VM resource isolation technology. Considering user's budget demand, task's characteristic, and the limited availability of the idle resource, it can be formulated as a convex-optimization problem, as shown in Formula (24).

$$\begin{array}{ll} \min & t_e(\boldsymbol{r}(t_{ij}, p_e)) = \sum\limits_{k=1}^{D} \frac{l_k(t_{ij})}{r_k(t_{ij})} \\ s.t. & \boldsymbol{b}(p_e)^T \cdot \boldsymbol{r}(t_{ij}) \leq B(t_{ij}) \\ & \boldsymbol{r}(t_{ij}) \preceq \boldsymbol{a}(p_e) \end{array} \tag{24}$$

The Karush-Kuhn-Tucker (KKT) condition [20] has been proved the necessary and sufficient condition of the optimal solution to the above convex problem. As such, as long as we can solve the corresponding KKT conditions (as shown in

Formula (25)), we can solve the problem optimally. Whereas, it is non-trivial to directly solve the Formula (25).

$$\begin{cases} \boldsymbol{b}(t_{ij})^T \cdot \boldsymbol{r}^*(t_{ij}) \leq B(t_{ij}) & \\ r_k^*(t_{ij}) - a_k(p_e) \leq 0 & k = 1, 2, \cdots, D \\ \lambda \geq 0, \nu \succeq 0 & \\ \lambda \cdot (\boldsymbol{b}(t_{ij})^T \cdot \boldsymbol{r}^*(t_{ij}) - B(t_{ij})) = 0 & \\ \nu_k \cdot (r_k^*(t_{ij}) - a_k(p_e)) = 0 & k = 1, 2, \cdots, D \\ -\frac{l_k(t_{ij})}{r_k^{*2}(t_{ij})} + \lambda \cdot b_k(t_{ij}) + \nu_k = 0 & k = 1, 2, \cdots, D \end{cases} \tag{25}$$

In the next section, we will propose an algorithm with polynomial time complexity ($D^2$, where $D$ is the number of dimensions), to find the solution of the Formula (25).

*3.3.2. LOVRA algorithm*

The basic idea of *LOVRA* algorithm is temporally removing the last condition (resource availability constraint) in the problem formulated in Formula (24) at the beginning, and then recursively tuning the solution by taking into account the resource availability constraint until finding an allocation case satisfying this constraint. Then, the output of the algorithm will be the feasible case satisfying the above KKT condition, which can also be proved in theory. Consequently, in the following text, we will first study the problem without the resource availability constraint, i.e., Formula (26).

$$\begin{aligned} \min \quad & t_e(\boldsymbol{r}(t_{ij}, p_e)) = \sum_{k=1}^{D} \frac{l_k(t_{ij})}{r_k(t_{ij})} \\ s.t. \quad & \boldsymbol{b}(p_e)^T \cdot \boldsymbol{r}(t_{ij}) \leq B(t_{ij}) \end{aligned} \tag{26}$$

**Theorem 1.** *The optimal resource share vector* $\boldsymbol{r}^{(*)}(t_{ij})$ *is shown in Equation (27), where k=1, 2,* $\cdots$*, D. (Note that* $\boldsymbol{r}^{(*)}(t_{ij})$ *is not calculated without resource availability constraint, unlike the notation* $\boldsymbol{r}^*(t_{ij})$*.)*

$$r_k^{(*)}(t_{ij}) = \frac{\sqrt{l_k(t_{ij})/b_k(p_e)}}{\sum_{k=1}^{D} \sqrt{1_k(t_{ij}) b_k(p_e)}} \cdot B(t_{ij}) \tag{27}$$

PROOF. The proof is simple. In fact, we could derive Equation (28) based on convex optimization theory.

$$r_1 : r_2 : \cdots : r_D = \sqrt{\frac{l_1}{b_1}} : \sqrt{\frac{l_2}{b_2}} : \cdots : \sqrt{\frac{l_D}{b_D}} \tag{28}$$

In order to minimize $t_e(\boldsymbol{r})$, the optimal resource vector $\boldsymbol{r}^{(*)}$ should make $\boldsymbol{b}^T \cdot \boldsymbol{r}$ equal to $B$. So, we could get the conclusion, Equation (27).

We devise Algorithm 2 (namely *LOVRA*) for finding the optimal solution to the problem with resource availability constraint (Formula (24)), as shown below.

**Algorithm 2** LOCAL OPTIMAL VM RESOURCE ALLOCATION (LOVRA)

```
function name: LOVRA($\Pi$, $B(t_{ij})$, $\boldsymbol{w}(t_{ij})$, $\boldsymbol{b}(p_i)$, $\boldsymbol{a}(p_i)$);
Input:  $\Pi$: the execution dimension set
        $B(t_{ij})$: $t_{ij}$'s budget
        $\boldsymbol{l}(t_{ij})$: $t_{ij}$'s preferential weight vector
        $\boldsymbol{b}(p_e)$: execution node $p_e$'s price vector
        $\boldsymbol{a}(p_e)$: execution node $p_e$'s availability vector
Output: $\boldsymbol{r}^*(t_{ij})$: $t_{ij}$'s optimal resource allocation vector on $p_e$
1: $\Gamma = \Pi$, $C = B(t_{ij})$, $\boldsymbol{r}^* = \Phi$ (empty set);
2: repeat
3:     $\boldsymbol{r}_\Gamma^{(*)}(t_{ij})$ = CO-STEP($\Gamma$,$C$); /*Compute optimal $r^{(*)}$ based on $\Gamma$ with unbounded capacity assumption*/
4:     $\Omega$=$\{d_k | d_k \in \Gamma$ & $r_k^{(*)}(t_{ij}) > a_k(p_e)\}$;/*Select elements violating the resource availability constraint)*/
5:     $\Gamma = \Gamma \backslash \Omega$; /*$\Gamma$ takes away $\Omega$*/
6:     $C = C - \sum_{d_k \in \Omega} (b_k(p_e) \cdot a_k(p_e))$; /*Update $C$*/
7:     $\boldsymbol{r}^* = \boldsymbol{r}^*(t_{ij}) \cup \{r_k^*(t_{ij}) = a_k(p_e) \mid d_k \in \Omega$ & $a_k(p_e)$ is $d_k$'s upper bound$\}$;
8: until ($\Omega = \Phi$);
9: $\boldsymbol{r}^*(t_{ij}) = \boldsymbol{r}^*(t_{ij}) \cup \boldsymbol{r}_\Gamma^{(*)}(t_{ij})$;
```

We can prove that the output of Algorithm 2 is the optimal solution that satisfies the KKT condition (25), in Theorem 2.

**Theorem 2.** *Given a submitted task $t_{ij}$ with its workload vector $\boldsymbol{l}(t_{ij})$ and a budget $B(t_{ij})$ and a qualified node $p_e$ with its resource price vector $\boldsymbol{b}(p_e)$, Algorithm 2's output $\boldsymbol{r}^*$ is the optimal solution to the problem formulated in Formula (24).*

The basic idea of the proof to Theorem 2 is confirming the fact that the output of Algorithm 2 must satisfy the Condition (25), which is not a hard work, so we will omit the details of the proof here.

Based on the Theorem 2, an interesting result is that we can further derive Theorem 3, which will be used later in the proof of the ex-post efficiency of our design in next section.

**Theorem 3.** *Suppose under Algorithm 2 (without considering the reverse vickrey auction used in Algorithm 2), the final real execution time of task $t_{ij}$ is denoted as $T_f(t_{ij})$. Given $T_f(t_{ij})$ ($t_{ij}$'s deadline set by its user), $t_{ij}$'s execution property ratio, and execution node $p_e$'s price vector $\boldsymbol{b}(p_e)$, Algorithm 2's output $\boldsymbol{r}^*(t_{ij})$ is the optimal solution that minimizes the payment based on the node's price vector.*

PROOF. We denote task $t_{ij}$'s allocated resource vector as $\boldsymbol{r}^*(t_{ij})$, then it must satisfy Equation (29) and Equation (30), where $B(t_{ij})$ is task's budget used in Algorithm 2.

$$t_e^*(\boldsymbol{r}(t_{ij}^{p_e})) = \sum\nolimits_{i=1}^{D} \frac{l_i(t_{ij})}{r_i^*(t_{ij})} = T_f(t_{ij}) \tag{29}$$

$$\sum\nolimits_{i=1}^{D} b_i(p_e) r_i^*(t_{ij}) = B_f(t_{ij}) \leq B(t_{ij}) \tag{30}$$

Since we do not consider the impact of the reverse vickrey auction in Algorithm 1, the user's payment could be estimated by Equation (31), which is calculated using the execution node's price vector.

$$EP_{r^*}(t_{ij}) = B_f(t_{ij}) \cdot T_f(t_{ij}) \tag{31}$$

If Theorem 3 does not hold, there must exist a resource allocation $\boldsymbol{r}'(t_{ij})(\neq \boldsymbol{r}^*(t_{ij}))$ for running $t_{ij}$ on $p_e$, satisfying $\sum\limits_{k=1}^{D} \frac{l_k(t_{ij})}{r'_k(t_{ij})} = T_f(t_{ij})$ and Inequality (32).

$$EP_{r'}(t_{ij}) = (\sum\nolimits_{k=1}^{D} b_k(t_{ij}) r'_k(t_{ij})) \cdot T_f(t_{ij}) < B_f(t_{ij}) \cdot T_f(t_{ij}) = EP_{r^*}(t_{ij}) \tag{32}$$

Inequality (32) implies that there must exist a $\Delta r$>0, such that the new resource allocation $\{r'_1(t_{ij})$+$\Delta r$, $r'_2(t_{ij})$, $r'_3(t_{ij})$, $\cdots$, $r'_D(t_{ij})\}$ also satisfies Inequality (32), yet its execution time (i.e., $\frac{l_1(t_{ij})}{r'_1(t_{ij})+\Delta r} + \sum_{i=2}^{D} \frac{l_i(t_{ij})}{r'_i(t_{ij})}$)will be smaller than $t_e^*(\boldsymbol{r}(t_{ij}^{p_e}))$, which contradicts to the fact that $t_e^*(\boldsymbol{r}(t_{ij}^{p_e}))$ is minimized proved by Theorem 2.

## *3.4. Discussion of Ex-post Efficiency*

In this section, we will theoretically prove the ex-post efficiency (a.k.a., ex-post optimality) of our design, which is mainly composed of Theorem 4 and Theorem 5. Specifically, we will first prove the ***ex-ante*** efficiency of the task execution (i.e., our resource allocation is optimal from the perspective of consumers based on their prior knowledge about resource states). After that, we will prove the incentive compatibility via Theorem 5 (i.e., all rational suppliers will truthfully declare their valuations on their owned resources). Finally, we will derive the overall ***ex-post*** efficiency for our design in Corollary 1.

**Theorem 4.** *Under the DVA algorithm (i.e., Algorithm 1), task's execution will always achieve ex-ante efficiency based on the prior collected information, with guaranteed Inequality (18).*

PROOF. Based on the Equation (14) and the definition that $\nu_{cs}^{exe}(t_{ij})$ is proportional to $t_{ij}$'s execution utility, it is easy to see that Theorem 4 holds is and only if task $t_{ij}$'s execution utility can get the maximized value under our DVA algorithm.

Based on the Definition 1, $t_1$ is the time point such that $t_{ij}$'s execution time $t_e(t_{ij})$ is shortest among all the candidate choices (i.e., the execution times by using the nodes in *QSET*$(t_{ij})$). That is, $t_{ij}$'s practical execution time must be no less than the time point $t_1$. On the other hand, according to the definition of $\nu_{cs}^{exe}(t_{ij})$ in the Formula (14), when *su*$(t_{ij})$=0, then $\nu_{cs}^{exe}(t_{ij})$=0 due to $P_{cs}^{exe}(t_{ij})$=0$\leq ep(t_{ij})$. That is, $t_{ij}$'s execution time must be in the range of [$t_1$, $t_1 + \frac{1}{k_1}$], as the shaded area shown in the Figure 3.

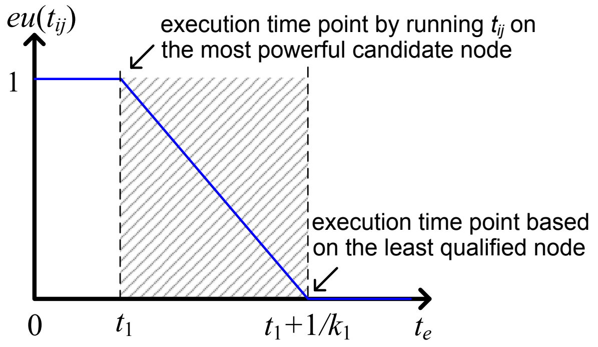


Figure 3: The viable range (shaded area) of task's execution utility for each user's task execution

In addition, according to the definition of $P_{cs}^{exe}(t_{ij})$, maximizing the execution utility *eu*$(t_{ij})$ based on user's budget constraint is a basic condition (otherwise, the user will be unsatisfied (i.e., $P_{cs}^{exe}(t_{ij})$=0)). As follows, we will prove that $P_{cs}^{exe}(t_{ij})$ is already maximized, provided that the task's information (including its resource demand, the weight vector, etc.) is honestly declared. That is, any other skewed resource allocation rather than the allocation calculated by Line 6$\sim$9 of Algorithm 1 will get inferior payoff (i.e., $P_{cs}^{exe'}(t_{ij}) \leq P_{cs}^{exe}(t_{ij})$).

If $t_e(t_{ij})$=$t_1$, according to Theorem 3, the payment *EP*$(t_{ij}^{p_e})$ is already minimized based on the imagined given deadline $t_1$. Hence, $P_{cs}^{exe}(t_{ij})$(=$\nu_{cs}^{exe}(t_{ij}) - EP(t_{ij}^{p_e})$) must be maximized for any resource allocation subject to the budget constraint $B(t_{ij})$ (note that $\nu_{cs}^{exe}(t_{ij})$ is supposed to be a constant here because $eu(t_{ij})$=1).

If $t_1 < t_e(t_{ij}) < t_1 + \frac{1}{k_1}$, there are three situations to discuss:

- If the execution time (denoted as $t'_e(t_{ij})$) by executing $t_{ij}$ using another resource share vector $r'(t_{ij})(\neq r^*_{(k)}(t_{ij}))$ on the selected execution node is less than that the execution time $t_e(t_{ij})$ with $r^*_{(k)}(t_{ij})$, then, this contradicts to the fact that $t_e(t_{ij})$ should be minimized based on $t_{ij}$'s budget, which was proved by Theorem 2. That is, $t'_e(t_{ij}) \geq t_e(t_{ij})$ will definitely hold.

- If $t'_e(t_{ij})$=$t_e(t_{ij})$, the situation is similar to that with $t_e(t_{ij}) = t_1$. That is, according to Theorem 3, the payment *EP*$(t_{ij}^{p_e})$ is already minimized, so $P_{cs}^{exe}(t_{ij})$=$\nu_{cs}^{exe}(t_{ij}) - EP(t_{ij}^{p_e})$ must be the maximized payment.

- If $t'_e(t_{ij}) > t_e(t_{ij})$, we use $eu'(t_{ij})$, $\nu_{cs}^{exe'}(t_{ij})$, and $EP'(t_{ij}^{p_e})$ to denote the corresponding execution utility, consumer's valuation on $t_{ij}$'s execution, and execution payment respectively. It is obvious that $eu'(t_{ij}) < eu(t_{ij})$ according to definition of the execution utility, thus, $\nu_{cs}^{exe'}(t_{ij}) < \nu_{cs}^{exe}(t_{ij})$. On the other hand, according to Theorem 3, we know that $EP'(t_{ij}^{p_e}) > EP(t_{ij}^{p_e})$. Accordingly, it is obvious that $P_{cs}^{exe'}(t_{ij})$ = $\nu_{cs}^{exe'}(t_{ij}) - EP'(t_{ij}^{p_e})$ < $\nu_{cs}^{exe}(t_{ij})$ $- EP(t_{ij}^{p_e})$ = $P_{cs}^{exe}(t_{ij})$.

All in all, $P_{cs}^{exe'}(t_{ij}) \leq P_{cs}^{exe}(t_{ij})$.

**Theorem 5.** *DVA algorithm (i.e., Algorithm 1) can achieve the ex-post incentive compatibility, by guaranteeing Inequality (17) and (19) for any participant.*

PROOF. *Proof of Inequality (17)*:

Basically, the problem can be converted to discussing whether $t_{ij}$'s user can further improve its payoff by overbidding or underbidding its scheduling bid (i.e., when $B(t_{ij}) \neq \nu_{cs}^{sch}(t_{ij})$). According to Algorithm 1, $t_{ij}$ will be scheduled if and only if its declared budget is the largest among all the bids. Consequently,

$AB(t_{ij})$ = $\max_{B(t_{xy})\leq B(t_{ij})}(B(t_{xy}))$ = $\max_{\{x,y\}\neq\{i,j\}}(B(t_{xy}))$ according to the line 4 of Algorithm 1, which means that the winner task $t_{ij}$ will just need to pay the payment based on the second lowest budget. Hence, we can get the Formula (33) according to Formula (13).

$$P_{cs}^{sch}(t_{ij}) = \begin{cases} \upsilon^{sch}(t_{ij}) - \max\limits_{\{x,y\}\neq\{i,j\}}(B(t_{xy})), & B(t_{ij}) > \max\limits_{\{x,y\}\neq\{i,j\}}(B(t_{xy})) \\ 0 & , \ otherwise \end{cases} \tag{33}$$

As follow, we will prove that Inequality (17) always holds in the two situations (overbidding or underbidding).

- If $\nu_{cs}^{sch}(t_{ij}) < B(t_{ij})$ (i.e., over-bidding), there are three cases: If $\max\limits_{\{x,y\}\neq\{i,j\}}(B(t_{xy})) < \nu_{cs}^{sch}(t_{ij})$ or $\max\limits_{\{x,y\}\neq\{i,j\}}(B(t_{xy})) > B(t_{ij})$, the two strategies of the user (being honest or lying) have equal payoffs based on Inequality (33). If $\nu_{cs}^{sch}(t_{ij}) \leq \max\limits_{\{x,y\}\neq\{i,j\}}(B(t_{xy})) \leq B(t_{ij})$, then the payoff (=0) under honest bidding would be no less than that ($\leq$0) with overbidding.
- If $\nu_{cs}^{sch}(t_{ij}) > B(t_{ij})$ (i.e., under-bidding), there are also three cases to discuss: If $\max\limits_{\{x,y\}\neq\{i,j\}}(B(t_{xy})) > \nu_{cs}^{sch}(t_{ij})$ or $\max\limits_{\{x,y\}\neq\{i,j\}}(B(t_{xy})) < B(t_{ij})$, the two strategies (being honest or lying) have equal payoffs based on Inequality (33). If $B(t_{ij}) \leq \max\limits_{\{x,y\}\neq\{i,j\}}(B(t_{xy})) \leq \nu_{cs}^{sch}(t_{ij})$, then the payoff ($\geq$0) under honest bidding will be no less than that (=0) under overbidding.

*Proof of Inequality (19)*:

Basically, this inequality can be converted to discussing whether the contributor can improve its payoff by over-declaring or under-declaring the valuations of their resources. We take into account such two situations from the perspective of resource contributor (node $p^*_{(k)}$): $p^*_{(k)}$ winning bid and $p^*_{(k)}$ losing bid.

If $p^*_{(k)}$ wins the bid, then $p^*_{(k)}$ must be the qualified resource node (in *QSET*$(t_{ij})$) with the lowest payment required, and the payment calculated based on its price vector is denoted as $EP^*_{(k)}$. If $EP^*_{(k)} > \nu_{ct}(t_{ij})$ (deliberately over-declaration), since $EP^*_{(k)} \leq EP^*_{(k+1)}$, we can get $\upsilon_{ct}(t_{ij}) < EP^*_{(k)} \leq EP^*_{(k+1)}$. According to Algorithm 1, $P_{ct}(t_{ij}^{p_e})$(=$EP^*_{(k+1)} - \nu_{ct}(t_{ij})$) will be the same in the two strategies (over-declaration and honest-declaration), which implies that over-declaration (lying behavior) cannot improve the payoff at all.

If $p^*_{(k)}$ loses the bid, then there must exist one qualified resource node (say $p^*_{(l)}$ where $l < k$) in *QSET*$(t_{ij})$ such that $t_{ij}$'s payment is lower on $p^*_{(l)}$ than that on $p^*_{(k)}$.

- If $EP^*_{(l)} < v_{ct}(t_{ij})$, the fact that $p^*_{(k)}$ loses the bid will not be changed no matter $EP^*_{(k)} = v_{ct}(t_{ij})$ or not, which means that the payoff will not change for the two strategies (under-declaration and honest-declaration).
- If $EP^*_{(l)} \geq v_{ct}(t_{ij})$, honest contributor (with $p^*_{(k)} = v_{ct}(t_{ij})$) will get payoff $=EP^*_{(l)} - v_{ct}(t_{ij}) \geq 0$, while lying contributor will lose the bid, suffering zero payoff.

Hence, resource contributors' payoff under honest behaviors will always be no less than the payoff gained under their lying behaviors, i.e. our algorithm owns incentive-compatibility to resource contributors.

**Corollary 1.** *Under DVA algorithm (i.e., Algorithm 1), the whole SoC system is ex-post efficient w.r.t. each task's execution (a.k.a., ex-post optimality).*

PROOF. For each task's execution, it must be ex-ante efficient according to Theorem 4, which means that each task will be executed with optimal resource allocation **if** the resource states and prices are declared truthfully by the corresponding owners. On the other hand, the resource prices and state availability used in Algorithm 1 are reliable due to two factors: (1) as proved in Theorem 5, we know that each rational resource owner will truthfully reveal their valuations on the prices of their resources, based on the reverse-vickrey auction design; (2) Due to line 12 of Algorithm 1, the resource state availability will be rechecked right before any task schedule, to confirm the selected choice. Consequently, each task schedule and its execution should also be ex-post efficient (i.e., still optimal observed after its practical execution). In other words, the whole SoC system will converge to the ex-post efficiency status, where each task's final execution efficiency can be guaranteed as anticipated.

## 4. Performance Evaluation

### *4.1. Experimental Setting*

For our simulation, we first built an emulated credit-scheduler in accordance with the design of XEN [21]. Then, we carefully constructed the CAN protocol [19] using the Peersim tool [22] and improved it using our designed Proactive-Index-Diffusion (PID) strategy [14]. There are thousands of nodes, each with random settings (Table 1). Each task needs an expected five-dimensional resource vector {computation load, disk-IO load, network load, disk size and memory size} to start and its execution time is only related to the first three dimensions. Each task's workload along some attribute is set as the product of its random capacity value based on Table 1 and a demand ratio (denoted by $\lambda$ ($\leq$1)). Higher demand ratio means higher level of flash-crowd of consumers with similar demand or interest on resources. We simulate the Internet communication by grouping all nodes into different LANs, and two nodes across LANs have to communicate through WAN network bandwidth. By leveraging the event-driven mode under the Peersim tool, each experiment simulates 86400 seconds (one day) using 4320 event cycles and 432000 periodical cycles, and the user tasks will be periodically generated on each node based on Poisson process. The tasks' resource demands ($e(t_{ij})$ will be set according to Table 2. The prices set by nodes and expected by users are randomly generated such that the per-metric-per-time value (i.e., $b_k(p_e) \cdot r_k(t_{ij})$, where $k$=1,2,$\cdot$,$D$) is in [1,10]. $t_{ij}$'s budget is randomly generated in [1$\times\sum_{k=1}^{D} \beta(t_{ij}) \cdot e(t_{ij})$,1.4$\times\sum_{k=1}^{D} \beta(t_{ij}) \cdot e(t_{ij})$].

Table 1: System Setting

| Parameter | Value |
|---|---|
| # of nodes | 1000 $\sim$ 4000 |
| # of processors per node | 1,2,4,8 |
| computation rate per processor | 1,2,2.4,3.2 (GHz) |
| disk-I/O speed per node | 20,40,60,80 Mbps |
| memory size per node | 512, 1024, 2048, 4096 MB |
| disk size per node | 20, 60, 120, 240 GB |
| LAN network bandwidth | 5 $\sim$ 10 Mbps |
| WAN network bandwidth | 0.2 $\sim$ 2 Mbps |

Table 2: Task's Demand Setting

| Parameter | Value | Parameter | Value |
|---|---|---|---|
| demand ratio $\lambda$ | 1, 0.5, 0.25 | cpu rate | $\lambda \sim 25.6\lambda$ |
| I/O speed | $20\lambda \sim 80\lambda$ | memory size | $512\lambda \sim 4096\lambda$ |
| disk size | $20\lambda \sim 240\lambda$ | bandwidth | $0.1\lambda \sim 10\lambda$ |

We mainly focus on seven metrics, scheduled task ratio, throughput ratio, fairness index [23] of task's execution efficiency, execution utility (average $eu(t_{ij})$), task utility (average $tu(t_{ij})$), task's Payment-to-Budget Ratio (abbreviated as *PBR*), and the contributors' Income-to-Expectation Ratio (abbreviated as *IER*). The scheduled task ratio is calculated by the ratio of the number of scheduled tasks that have been migrated/queued onto some resource nodes and the number of total tasks submitted. The throughput ratio (or finished task ratio) is calculated by the ratio of the number of finished tasks and the number of total tasks submitted. Task's execution efficiency (denoted as $e_{ij}$) is defined as the ratio of its execution time to the theoretical value estimated using average computing ability in the system. Hence, the overall fairness index of task's execution efficiency (denoted as $\varphi$) can be calculated by Equation (34), where $n_i$ means the number of tasks submitted to node $p_i$.

$$\varphi = \frac{(\sum_{i=1}^{n} \sum_{j=1}^{n_i} e_{ij})^2}{(\sum_{i=1}^{n} n_i) \cdot (\sum_{i=1}^{n} \sum_{j=1}^{n_i} e_{ij}^2)} \tag{34}$$

The calculation of execution utility and task utility is described in Section 2. In our simulation, if a task fails to find any qualified resources such that its execution cannot be started at all, its execution utility will be set to 0. *PBR* and *IER* are shown in Formula (35) and (36), where $I_{ct}^{real}(p_e)$ and $I_{ct}^{expect}(p_e)$ denote the resource owner's final income earned and its expected income (evaluated using its own assigned prices) respectively.

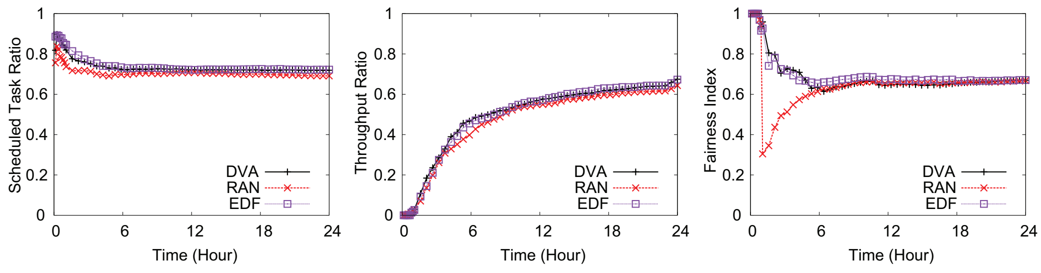
$$PBR(t_{ij}) = \frac{RPU(t_{ij}) \cdot t_e(t_{ij})}{B(t_{ij})} \tag{35}$$

$$IER(p_e) = \frac{I_{ct}^{real}(p_e)}{I_{ct}^{expect}(p_e)} \tag{36}$$

Our experiments are conducted under different competitive situations with various workloads of submitted tasks. As a comparative baseline, we also implement two more node-selection strategy, namely random policy (RAN) and Earliest Deadline First (EDF) strategy. The former is adopted in our previous work [14] and the latter is commonly used in many other task scheduling designs [24, 25]. Under the random strategy, each scheduler node randomly selects one execution node from the qualified candidate resources queried by PID-CAN protocol [14]. This random-selection design delivers satisfactory throughput due to mitigated decision conflict among users. Under EDF, each task will select the nodes from the candidate nodes on which it can be finished earliest as the final execution node. From our experiments, we can observe both of the two solutions will suffer limited contributors' payoffs which may easily mitigate contributors' participating motivations.

### *4.2. Experimental Result*

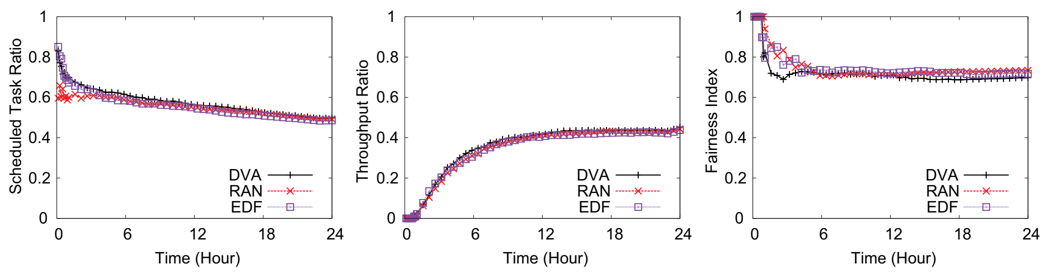
In our previous work [14], the execution nodes are randomly determined from candidate nodes queried by PID-CAN protocol at the task scheduling phase. Such a method has been proved effective to deliver the high system throughput in terms of higher scheduled task ratio and finished task ratio. In comparison, we first compare our new method, i.e. Double-sided Vickrey Auction algorithm (DVA) to the random-selection policy and EDF policy w.r.t. scheduled task ratio, throughput ratio and fairness index, as shown in Figure 4, Figure 5, and Figure 6 respectively. We can observe that the three metrics based on DVA policy is no worse than those of the other two policies. Specifically, the first two metrics imply that the overall task processing efficiency around the whole system will be exactly the same among the three different node-selection policies. The fairness index of task execution efficiency will converge to the same level, implying that all of tasks can be treated in the similarly fair way around the whole system. Accordingly, we conclude that our DVA approach will not degrade the task processing ratios and fairness of treatment at all, in that the main factor impacting the task processing ratios is the resource discovery protocol. In the following text, we will show that our DVA approach will perform satisfactorily on the task utility and significantly outperform the other two solutions especially w.r.t. the level of contributor's income.



(a) scheduled ratio (b) throughput ratio (c) fairness index

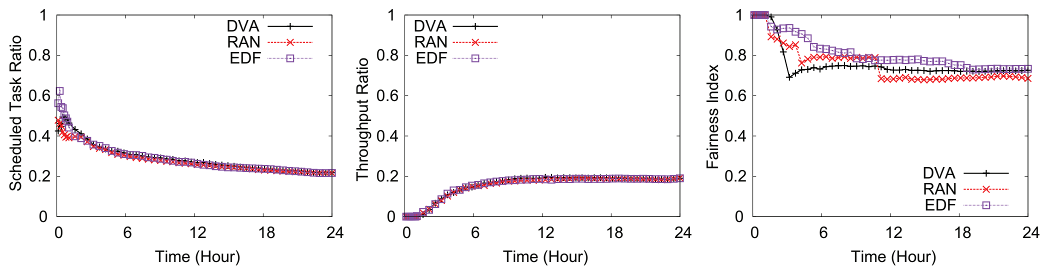
Figure 4: Processing Performance ($\lambda$=0.25)

Through Figure 7, Figure 8, and Figure 9, we show the average execution utility and average task utility, among all the tasks (including finished ones and failed ones). We do not show the scheduling utility of finished tasks because it is observed always equal to 1 for any task scheduling under the three different approaches in our simulation. Through all of the six figures, we can clearly observe that our designed DVA policy (selecting nodes based on reverse vickrey auction) and RAN policy (random selection) will significantly outperform the traditional Earliest Deadline First (EDF) policy. This is mainly because that EDF always select the most powerful node in task scheduling, suffering from the serious decision conflict among the individual schedulers. In the DVA policy, however, we



(a) scheduled ratio (b) throughput ratio (c) fairness index

Figure 5: Processing Performance ($\lambda$=0.5)



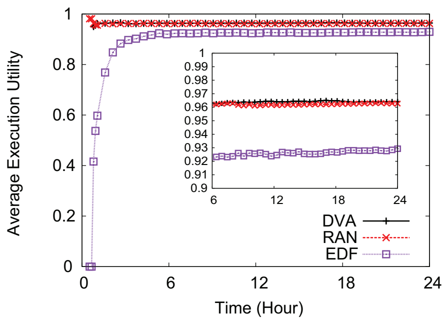
(a) scheduled ratio (b) throughput ratio (c) fairness index

Figure 6: Processing Performance ($\lambda$=0.83)

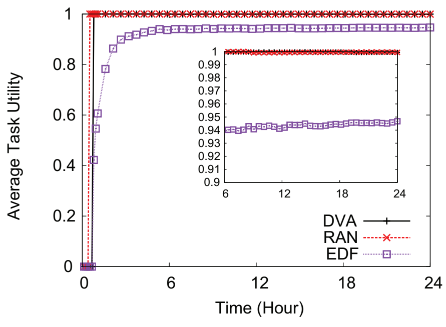
also take into account user's estimated payment (i.e., budget) and the resource prices arbitrarily set by the owners, which can effectively disperse the decision conflict. That is, its effect is a little similar to that of randomly selecting node to a certain extent, so its performance is pretty close to that of random node-selection policy.

The most outstanding feature of the DVA algorithm is realizing the win-win effect, such that any resource consumer and resource contributor will be both satisfied on their payment or incomes. Specifically, the Payment-to-Budget Ratio (Formula (35)) should never be greater than 1, and any contributor's Income-to-Expectation Ratio (Formula (36)) should keep over 1 at any time. We show the results in Figure 10 and Figure 11, where the demand ratio ($\lambda$) is set to 0.5.

Figure 10 (a) presents the average PBR over time. We can clearly observe that the average of PBR under our DVA policy is a little greater than those of RAN and EDF. This is because that the final real payment unit ($RPU$) under the DVA policy is calculated based on Formula (21), whose value must be no lower than the previous auction-based budget ($AB$). In the Figure 10 (b), we can clearly see that

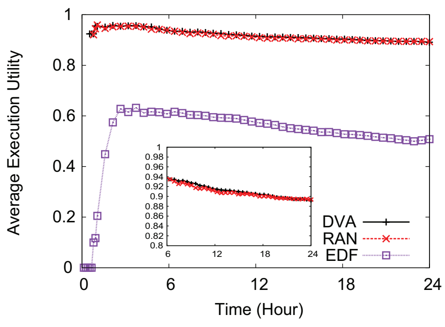


(a) average execution utility

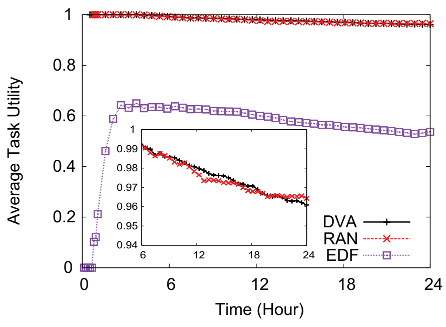


(b) average task utility

Figure 7: Task Utility ($\lambda$=0.25)

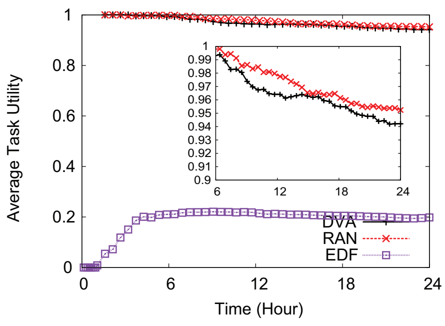


(a) average execution utility

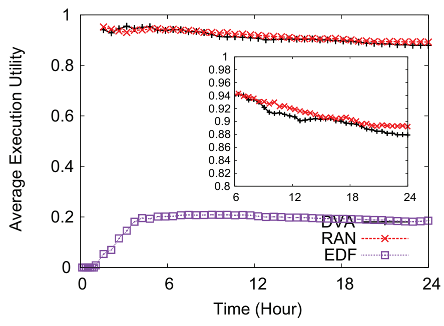


(b) average task utility

Figure 8: Task Utility ($\lambda$=0.5)

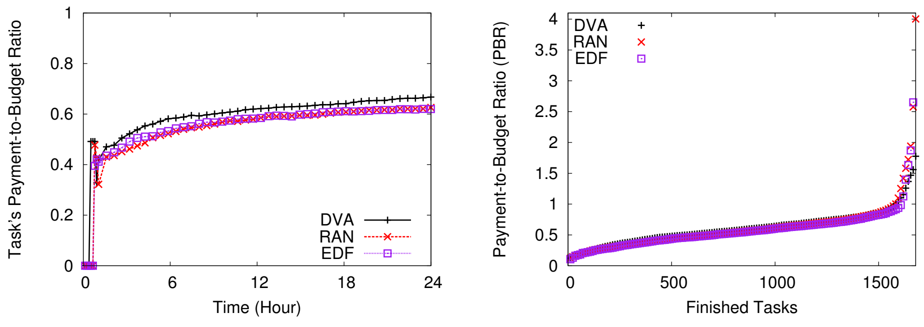


(a) average execution utility



(b) average task utility

Figure 9: Task Utility ($\lambda$=0.83)



(a) average PBR (b) distribution of PBR

Figure 10: Task's Payment-to-Budget Ratio (PBR)

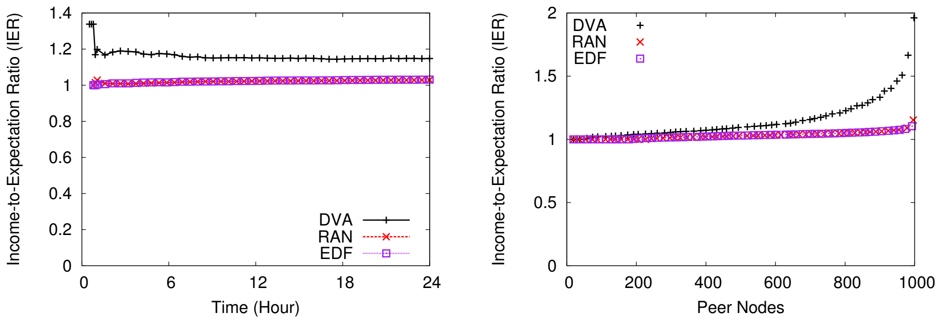
from among all of about 1700 finished tasks during the one-day test, the largest PBR under RAN and EDF scales up to 4 and 2.5 respectively, while that of DVA policy is less than 2, which means that the DVA policy delivers higher stability on the PBR.

In Figure 11 (a), we present the average IER of resource contributors over time. We can observe that our DVA policy will achieve much higher average IER than the other two policies. Moreover, from the Figure 11 (a), we can also conclude that: (1) almost all of the nodes get the prominently greater income than their own expectation based on their own prices, and (2) the largest IER can even get up to about 2. This is because of the reverse Vickrey auction design (i.e., Formula (20) can make sure that any final real payment unit will be greater than the one computed by the current owner's prices). All in all, through these two figures, we can conclude that the resource contributors will get much more satisfied under DVA policy than under the other two policies.

## 5. Related Work

Some socially optimal resource allocations are extensively studied in literatures.

- U. Endriss, et al. [26], for example, propose a negotiation-based allocation for getting socially optimal effect, in the sense that the social welfare is maximized in a distributed utilitarian environment. They also discuss the Pareto and Lorenz optimality [27], as well as envy-freeness [28]. The significant difference between their work and this paper is three-fold: (1) the resources



(a) average IER (b) distribution of IER

Figure 11: Node's Income-to-Expectation Ratio (IER)

in their work are assumed to be indivisible and non-sharable, while the resources in Cloud system are actually allowed to be divided on demand or time-shared; (2) they adopt the negotiation method, which may suffer the serious transmission overhead caused by multiple times of communication, while we adopt the sealed auction that implies the minimized demand on communication between suppliers and consumers; (3) our designed Double-sided Vickery Auction algorithm can bring both resource consumers and contributors to truthfully expose their expectations, but theirs cannot do so.

- I. Menache [29] build a cloud model to analyze the socially optimal pricing mechanism over the divisible resources for cloud computing platform. They could prove that the social optimum among resource consumers can be reached under their cloud model. Our work significantly different from their work in two facets. On one hand, the resources in our model are geographically distributed instead of being centrally managed, which is faced with more challenges in resource discovery and allocation. On the other, our work aims to realize the win-win effect, which also takes into account the resource contributor's payoff (or motivation), while their work only concerns the consumers' benefits.

- Y.M. Teo and M. Mihailescu [30] propose a strategy-proof pricing scheme for multiple resource type allocations. In addition to the objective of reaching "social optimum", their solution can also guarantee three features, individual rationality (rational agents gain higher from actively participation than from avoiding it), incentive compatibility (dominant strategy for each agent is to reveal its true valuation) and budget balance (sum of all agent payments is balanced). However, there are three key problems in their work making it unsuitable for our Self-Organizing Cloud model. Firstly, they adopt the central-controlled market-maker to organize the overall resource allocation, which is not scalable. Secondly, the multiple types of resources consumed by any task are assumed to come from different resource providers, which cannot be realized in the current cloud model. For example, it is costly and inefficient to execute a single application using CPU resource on one node but memory resource on another node, especially for the high communication overhead. Lastly, their work does not consider the difference of priorities of tasks but use a First-Come-First-Serve (FCFS) policy instead. In comparison, our utility function also takes the scheduling priority into account in addition to the execution utility, leading to higher flexibility based on user's demand.

In the domain on economic auction, ex-post efficiency is studied for years. M. Perry et al. [31] first proposed an ex-post efficient auction in 1999. They improved Vickrey auction and proved their solution may get the ex-post efficiency, such that each participant would never regret his/her equilibrium bids, and this remains so even after finding out the winning bids of other bidders. The key feature of our work is designing a practical ex-post efficient method for the Self-organizing Cloud (SoC). Each user task under our model follows an elaborative utility function, that takes into account both scheduling priority and execution priority. The final effect of our solution is rather valuable, because of the guaranteed user payoffs under the demands on specific budgets. P. Dasgupta et al. [32] explored a central-controlled efficient auction based on Vickrey auction, which can maximize surplus conditional on all available information. Differently, our work intensively studied how to improve Vickrey auction into the Cloud computing scenario, where the resources are divisible on demand. Some new researchers (such as [33]) also discussed the ex-post equilibria in double auctions of divisible assets. In comparison, our work is based on a fully distributed architecture, which can query compute resources on the Internet for tasks based on a fully-decentralized P2P protocol, possessing higher flexibility and robustness.

## 6. Conclusion and Future Work

This paper proposes a novel ex-post efficient resource allocation, which can guarantee the ex-post win-win effect such that resource consumers are always satisfied with the task execution and the resource contributors are also content with their payoffs for their resource-provisioning. By extending the traditional second-price bidding policy to a novel double-sided next-price bidding policy, ex-post incentive compatibility can be guaranteed: both rational resource consumers and contributors will truthfully reveal their demands on resources and prices. Finally, we confirm the efficiency of our design via a large-scale event-driven simulation. In the future, we plan to improve the fault-tolerance ability of this win-win resource allocation scheme by combining the replica-task execution strategy and check-pointing technology.

## Acknowledgments

This research is supported by HKU Basic Research grant (Grant No. 10401460), and Hong Kong UGC Special Equipment Grant (SEG HKU09).

## Appendix A. Notation Summarization

Table A.3: Summarization of Notations used in Problem Formulation and Analysis

| **Notation** | **Description** |
|---|---|
| $n$ | number of nodes in the SoC system |
| $p_i$ | a node (acting as both scheduler and supplier), where $i$ = 1,2,$\cdots$,$n$ |
| $\boldsymbol{c}(p_i)$ | capacity vector of node $p_i$ |
| $\boldsymbol{b}(p_i)$ | the price vector of node $p_i$'s resources on multiple dimensions |
| $t_{ij}$ | the $j$th task submitted to $p_i$ |
| $\boldsymbol{l}(t_{ij})$ | the workload vector of task $t_{ij}$ |
| $\boldsymbol{r}(t_{ij})$ | the resource share vector allocated to $t_{ij}$ |
| $\boldsymbol{r}^*(t_{ij})$ | the optimal resource share vector allocated to $t_{ij}$, outputted by *LOVRA* |
| $\beta(t_{ij})$ | the resource price vector expected by task $t_{ij}$'s owner |
| $t_{ij}^{p_e}$ | the task $t_{ij}$ is scheduled to be executed on node $p_e$ |
| $\boldsymbol{e}(t_{ij})$ | expected resource vector of $t_{ij}$ (i.e., least qualified resource requirement) |
| $B(t_{ij})$ | budget of $t_{ij}$'s user (evaluated by per-time-unit) |
| $\boldsymbol{a}(p_i)$ | the availability vector of $p_i$ (=$(a_1(p_d), a_2(p_d), \cdots, a_D(p_d))^T$, where $a_k(p_d)$=$c_k(p_d)-\sum_{\forall i,j} r_k^*(t_{ij}^{p_d})$ |
| $su(t_{ij})$ | scheduling utility of task $t_{ij}$ |
| $eu(t_{ij})$ | execution utility of task $t_{ij}$ |
| $\lambda_{ij}$ | the coefficient customized based on users expectation, to tune weight of $su(t_{ij})$ and $eu(t_{ij})$ |
| $RPU(t_{ij})$ | final real per-time-unit payment by $t_{ij}$'s user on $t_{ij}$'s execution |
| $EPU(t_{ij}^{p_e})$ | expected per-time-unit payment of resource owner |
| $EP(t_{ij}^{p_e})$ | expected payment of resource owner by executing $t_{ij}$ on $p_e$ |
| $DB(t_{ij}^{p_e})$ | demanded per-time-unit budget calculated based on consumer's expected resource vector and price vector |
| $AB(t_{ij})$ | auction-based budget (competed among submitted tasks on the same node) |
| $APU(t_{ij})$ | auction-based payment unit (competed among candidate resource nodes) |
| $TP(t_{ij})$ | total payment of task $t_{ij}$ by its user |
| $P_{cs}^{sch}(t_{ij})$ | consumer's payoff of scheduling task $t_{ij}$ |
| $P_{cs}^{exe}(t_{ij})$ | consumer's payoff of executing task $t_{ij}$ |
| $P_{ct}(p_e)$ | contributor's payoff of supplying node $p_e$'s resources |

## References

[1] L. Cherkasova, D. Gupta, A. Vahdat, Comparison of the three cpu schedulers in xen, SIGMETRICS Performance Evaluation Review 35 (2) (2007) 42–51.

[2] D. Gupta, L. Cherkasova, R. Gardner, A. Vahdat, Enforcing performance isolation across virtual machines in xen, in: Middleware, 2006, pp. 342–362.

[3] Amazon ec2: http://aws.amazon.com/ec2/.

[4] Cloud desktop: http://www.gladinet.com/.

[5] D. P. Anderson, Boinc: a system for public-resource computing and storage, Grid Computing, 2004. Proceedings. Fifth IEEE/ACM International Workshop on (2004) 4–10doi:10.1109/GRID.2004.14.

[6] G. Fedak, C. Germain, V. Neri, F. Cappello, Xtremweb: a generic global computing system, in: Proceedings of 1st IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid'01), 2001, pp. 582–587. doi:10.1109/CCGRID.2001.923246.

[7] W. B. MacLeod, J. Malcomson, Motivation and markets, Boston College Working Papers in Economics 339., Boston College Department of Economics (Jan. 1997).

[8] Cloud@home: http://clouds.gforge.inria.fr/pmwiki.php.

[9] G. Briscoe, A. Marinos, Digital ecosystems in the clouds: Towards community cloud computing, in: Proceedings of the 3rd IEEE International Conference on Digital Ecosystems and Technologies, 2009, pp. 103–108.

[10] Wuala: http://www.wuala.com/.

[11] R. G. Harris, Ex-post efficiency and resource allocation under uncertainty, Review of Economic Studies 45 (3) (1978) 427–36.

[12] R. Harris, N. Olewiler, The welfare economics of ex post optimality, Economica 46 (182) (1979) pp. 137–147.

[13] M. Tennenholtz, Ex-post equilibria in combinatorial auctions, SIGecom Exch. 7 (2007) 43–44.

[14] S. Di, C.-L. Wang, W. Zhang, L. Cheng, Probabilistic best-fit multi-dimensional range query in self-organizing cloud, in: The 40th International Conference on Parallel Processing (ICPP2011), 2011, pp. 763–772.

[15] L. Huang, J. Jia, B. Yu, B.-G. Chun, P. Maniatis, M. Naik, Predicting execution time of computer programs using sparse polynomial regression, in: The 24th Annual Conference on Neural Information Processing Systems, 2010, pp. 1–9.

[16] R. Sarikaya, C. Isci, A. Buyuktosunoglu, Program behavior prediction using a statistical metric model, in: Proceedings of the ACM SIGMETRICS international conference on Measurement and modeling of computer systems (SIGMETRICS'10), ACM, New York, NY, USA, 2010, pp. 371–372.

[17] Y. Song, X. Zhang, Y. Song, An effective and practical performance prediction model for parallel computing on non-dedicated heterogeneous now, Journal of Parallel and Distributed Computing (JPDC) 38 (1996) 63–80.

[18] H. R. Varian, Position auctions, International Journal of Industrial Organization 25 (2006) 1163–1178.

[19] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Shenker, A scalable content-addressable network, in: ACM SIGCOMM Computer Communication Review, New York, NY, USA, 2001, pp. 161–172.

[20] S. Boyd, L. Vandenberghe, Convex Optimization, Cambridge University Press, 2009.

[21] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, A. Warfield, Xen and the art of virtualization, in: Proceedings of the 19th ACM Symposium on Operating Systems Principles, New York, NY, USA, 2003, pp. 164–177.

[22] Peersim simulator: http://peersim.sourceforge.net.

[23] R. K. Jain, The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation and Modelling, John Wiley and Sons, 1991.

[24] C. Mattihalli, Designing and implementing of earliest deadline first scheduling algorithm on standard linux, in: Proceedings of the 2010 IEEE/ACM

Int’l Conference on Green Computing and Communications & Int’l Conference on Cyber, Physical and Social Computing (GREENCOM-CPSCOM’10), IEEE Computer Society, Washington, DC, USA, 2010, pp. 901–906.

[25] J. Singh, An algorithm to reduce the time complexity of earliest deadline first scheduling algorithm in real-time system, CoRR abs/1101.0056.

[26] U. Endriss, N. Maudet, F. Sadri, F. Toni, Negotiating socially optimal allocations of resources, Journal Artificial Intelligence Research (JAIR) 25 (2006) 315–348.

[27] H. Moulin, Axioms of Cooperative Decision Making (Econometric Society Monographs), Cambridge University Press, 1991.

[28] H. Nurmi, Fair division: From cake-cutting to dispute resolution, European Journal of Political Economy 12 (1) (1996) 169–172.

[29] I. Menache, A. Ozdaglar, N. Shimkin, Socially optimal pricing of cloud computing resources, in: in ValueTools’11: 5th International ICST conference on Performance Evaluation Methodologies and Tools, 2011.

[30] Y. M. Teo, M. Mihailescu, A strategy-proof pricing scheme for multiple resource type allocations, in: Proceedings of Intenational Conference on Parallel Processings (ICPP’09), 2009, pp. 172 –179.

[31] M. Perry and P.J. Reny, An Ex-post Efficient Auction, online at http://mailhost.econ.yale.edu/seminars/microt/mt99/reny-991110.pdf, 2000.

[32] P. Dasgupta and E. Maskin, Efficient Auctions, The Quarterly Journal of Economics CXV (2) (2000) 341 – 388.

[33] S. Du and H. Zhu, Ex Post Equilibria in Double Auctions of Divisible Assets, 1 – 33, 2012.