Social-optimized Win-win Resource Allocation for Self-organizing Cloud

Sheng Di [†], Cho-Li Wang [†], Luwei Cheng [†], Ling Chen [‡]

[†] The University of Hong Kong, Pokfulam, Hong Kong
{sdi, clwang, lwcheng}@cs.hku.hk

Abstract—**With the increasing scale of applications and the number of users, we design a Self-organizing Cloud (SoC) which aims to make use of the distributed volunteer computers or dedicated machines to provide powerful computing ability. These resources are provisioned elastically according to user's specific demand, by leveraging virtual machine (VM) resource isolation technology. Based on such a framework, we propose a social-optimized auction-based resource allocation scheme, which mainly tackles two issues: (1) how to make full use of the widely dispersed multi-attribute idle resources to construct a win-win situation, such that each task schedule could let both sides (resource providers and consumers) be satisfied with their final payoffs. (2) The total resource utility welfare should also be optimized to guarantee the overall performance around the global system. The key challenge of getting the win-win effect with social optimization is its provable NP-completeness. Finally, we validate that our approach can effectively improve the resource contributor's payoffs up to about five times as the level without our method via simulation work. Meanwhile, our approach can also keep a high level of the processing rate for the task scheduling.**

I. Introduction

Cloud computing offers scalable and on-demand virtualized resources as a utility service over the Internet with bypassed inter-operability constraints. With VM's resource isolation technology [1], 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. From anywhere on the Internet and at anytime, each user could access a tailor-made on-demand execution environment, where the processing ability comes from other Internet-connected sites over WAN. On the other hand, volunteer computing (or P2P desktop Grid) has also been studied for years especially for its great potential on millions of computers all over the world. Such platforms (e.g. BOINC [2], XtremWeb [3]) have made great contributions to scientific researches since 2000.

Our extended cloud framework (a.k.a. Self-organizing Cloud) combines resource isolation technology in 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 commodity node under cluster) is deployed with an autonomous resource state collector and a virtual machine monitor (VMM), being able to serve as both task scheduler and resource contributor. 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). When the task is submitted to a self-organizing host like desktop PC or dedicated commodity computer, the autonomous scheduler will find a qualified resource node on the network through a distributed range query protocol. Whenever a qualified node is found/determined, the task will be allocated with split resource shares from the execution node, and the resource owner will be awarded by the payment and credit. In such a framework, the participants will have high motivations to contribute their resources, because of two pillars. First, 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 [4]. Second, O. Nov et al. [5] show that many of volunteers will be enthusiastic to provide their idle resources, in order to compete against each other for getting salient credits. For example, there are already more than 280,000 volunteers providing about 5.5 PetaFLOPS every day with more than 650,000 computers in BOINC project [2].

Recently, there already exist many projects being designed based on this framework. A typical example is the on-going project Cloud@Home [6] undertaken by INRIA. Its major aim is to design a predictive model of resource availability for groups of volatile Internet resources and a set of strategies for checkpointing applications using VMs in low-bandwidth and volatile networks. Another example is Community Cloud [7], in which each participating host is also considered the resource supplier and the whole system is organized based on the principle of digital ecosystem. Such a system is claimed to provide openness of the usage (removing the dependence on vendors), avoid the system-wide cascade failures, and quite high environmental sustainability with significantly smaller carbon footprint than traditional server farms. Z. Xu et al. [8] implemented a Self-organizing Cloud prototype and make use of hole-punching technology to support VM live migration on the WAN, which is also transparent to cloud users. Wuala Cloud [9] 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. Its prominent features include guarantee of highly secure online

[‡]Ling Chen is with Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China, EMail: ling.chen@siat.ac.cn

storage using improved encrypting technologies, distributed file synchronization, real-time versioning and backup for the service data and so on.

Based on the above cases, designing the Self-organizing Cloud (SoC) system can definitely 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) it owns high robustness and reliability by minimizing the impact of one single node's failure or malicious user's DDoS attack to the whole system; (4) it suffers much less cost on central management and maintenance than that of traditional vender clouds, by leveraging the autonomous marketized environment with business model between resource consumers and contributors.

In addition to the opportunities mentioned above, there are still many new problems to solve, in order to achieve the user-agreed quality of services (QoS) based on their flexible customization. Since there are no central-controlled servers to coordinate the global resource states and prices, constructing an autonomous win-win situation for all participants by making full use of widely dispersed idle computing resources is a huge challenge. Specifically, there are two desired features: (1) win-win effect: In such a fully-distributed environment, each task-schedule on multi-attribute resources is supposed to let both sides (resource providers and consumers) be satisfied with their individual gains. (2) social optimization: The total resource utility welfare (e.g. the average task-execution turnaround time) should also be optimized to guarantee the performance around the global system. Recently, most existing researches just focus on the social-optimized feature of the resource allocation in their Gird/Cloud platforms for simplicity, but none of them guarantee the autonomous win-win effect to the best of our knowledge. For instance, B. An et al. [10] propose an automated negotiation approach for maximizing the social welfare in cloud's resource allocation, yet such a negotiation mechanism rarely discusses the resource owners' payoffs. Instead, our solution can make sure both sides are satisfied with final gains/payoffs, leading to a harmonious win-win effect with the social-optimized welfare.

Different resource providers may assign different prices (or pricing functions) on their heterogeneous resources, introducing a potentially competitive situation. If there are no robust pricing policies and auction mechanisms to coordinate the interests of resource providers and consumers, some participants may tend to lie on their real demands in order to maximize their selfish gains, weakening other ones' motivations. For example, English auction [11] and Dutch auction [12] are both conducted always open to the competitors, which are time-consuming and may potentially induce the resource price much higher than the real value, sacrificing auctioneers' benefits. In contrast, under the first-price sealed auction, the users/auctioneers cannot know others' bidding prices, so that they may incline to bid the prices that are lower than the true value of the resource they regard, at the cost of resource owner's profit. Hence, one challenging issue is how to guarantee that each participant would like to be honest on their real demands (both resource prices assigned by providers and the resource request proposed by consumers). The second-price based sealed auction [13] is a well-known policy that can make sure any resource users who lie against their true resource demands will get inferior gains. However, such a method can only discipline the resource consumers to give honest bids, while overlooking the resource owners' pricing incentives. In contrast, we propose a novel double-sided auction method that could leverage the second-price policy in order to enforce both resource consumers and providers to reveal their true expectations (both resource demands and price demands).

The rest of the paper is organized as follows. In Section II, we formulate the fully-distributed cloud resource allocation problem. In Section III, we first briefly introduce how nodes discover resources and collects volatile states, and then propose our major approach - namely Dual Vickrey Auction - which takes into account the contributors' incentives on truthfully revealing their expected prices. Section IV shows our simulation result and Section V comprehensively discusses the related works. Finally, we conclude the paper with future work in Section VI.

II. Problem Formulation and Analysis

We present the system overview in Fig. 1, to illustrate the task scheduling procedure. First of all (i.e. Step 1), the scheduler node p_s will find the qualified resources on the network using a fully-distributed federated range query protocol for the submitted task. After that (Step 2 shown in the figure), it will schedule the tasks based on our designed dual vickery auction, in terms of users' different scheduling bids and various prices assigned by resource owners. Finally (in Step 3), the tasks whose demands can match some resource availabilities will be dispatched/migiated onto the target node for its execution. The appropriate resource amounts (e.g. customized CPU speed and I/O speed) will be split from the idle resource of the execution node, by leveraging the VM resource isolation technology.

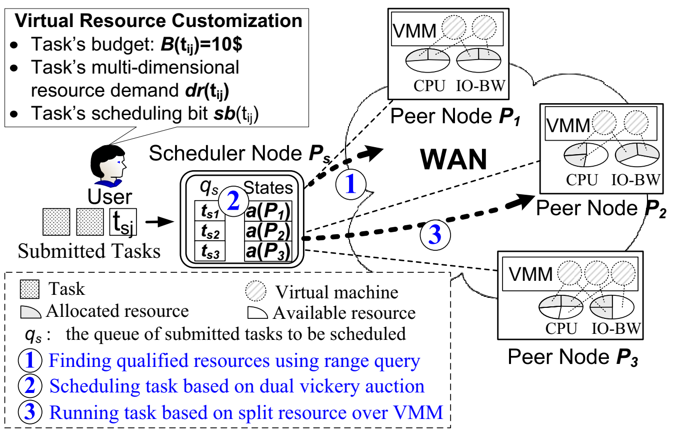
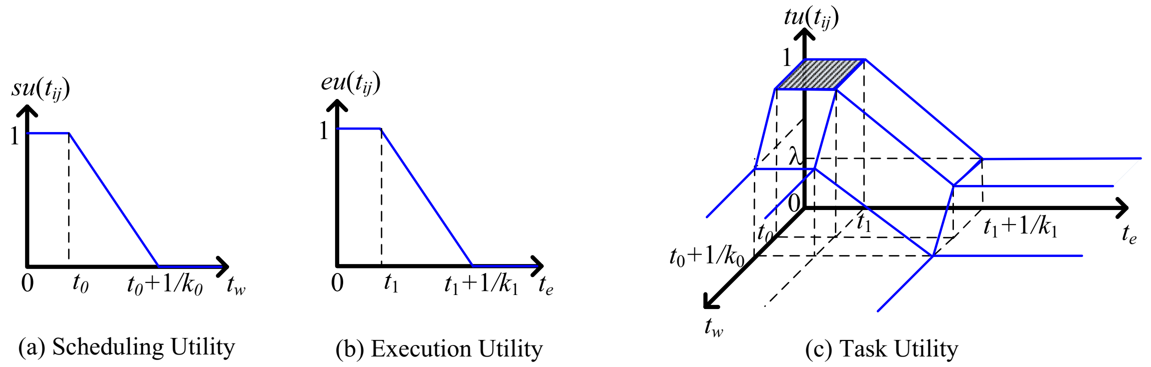

Virtual Resource Customization
• Task's budget: $B(t_{ij})$=10$
• Task's multi-dimensional resource demand $dr(t_{ij})$
• Task's scheduling bit $sb(t_{ij})$
User
Submitted Tasks
t_{sj}
Scheduler Node P_s
q_s
States
t_{s1} $a(P_1)$
t_{s2} $a(P_2)$
t_{s3} $a(P_3)$
1
2
3
WAN
VMM
CPU IO-BW
Peer Node P_1
Peer Node P_2
Peer Node P_3
Task
Virtual machine
Allocated resource
Available resource
q_s : the queue of submitted tasks to be scheduled
1 Finding qualified resources using range query
2 Scheduling task based on dual vickery auction
3 Running task based on split resource over VMM

Fig. 1. Task Scheduling in Self-organizing Cloud

Without loss of generality, we assume that there are n peer



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

Fig. 2. Graphical Illustration of Utility Functions

nodes, denoted by p_i ($1 \le i \le n$). Each node serves as both resource consumer (or user) and resource provider (or contributor). Each node's resource is multi-dimensional along R different attribute types (such as CPU, disk IO, network bandwidth, etc.). All the tasks submitted to p_i are marked as t_{ij} ($1 \le j \le n_i$), where n_i indicates the total number of tasks submitted to p_i. We use $t_{ij}^{p_d}$ to indicate p_d is t_{ij}'s execution node. Let $c_k(p_d)$ ($1 \le d \le n$) denote the capacity of the kth resource attribute on node p_d and $r(t_{ij}) = (r_1(t_{ij}), r_2(t_{ij}), \cdots, r_R(t_{ij}))^T$ denote the actual amounts of resource attributes allocated to t_{ij} when it is executed. Users need to specify a demanding resource vector $dr(t_{ij})$=$(dr_1(t_{ij}), dr_2(t_{ij}), \cdots, dr_R(t_{ij}))^T$, where $dr_k(t_{ij})$ is the estimated amount of resource at kth attribute demanded by task t_{ij}, to complete its execution within an expected execution time t_e. Hence, two necessary conditions are Inequality (1) and Inequality (2).

$$\textstyle\sum_{i,j} dr_k(t_{ij}^{p_d}) \le c_k(p_d), \quad k = 1, 2, \cdots, R \tag{1}$$

$$dr_k(t_{ij}) \le r_k(t_{ij}^{p_d}), \quad k = 1, 2, \cdots, R \tag{2}$$

Based on the above definitions, a node p_d's availability state is denoted as a vector $a(p_d) = (a_1(p_d), a_2(p_d), \cdots, a_R(p_d))^T$, where $a_k(p_d)$=$c_k(p_d)$−$\sum_{\forall i,j} dr_k(t_{ij}^{p_d})$. 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], in order to effectively discover available resources for submitted tasks (i.e. Step 1 shown in Fig. 1).

It is likely that different tasks own various characteristics, e.g. CPU-bound or IO-bound properties. We assume that tasks' characteristic can be predicted based on historical execution records [15] or analysis of their intrinsic programming structures. That is, each task t_{ij} is submitted with a weight vector $w(t_{ij})$=$(w_1(t_{ij})$,$w_2(t_{ij})$,$\cdots$,$w_R(t_{ij}))^T$, implying the weights on R multi-dimensional resource-attributes for its execution.

In this paper, we not only focus on resource consumer's *task utility*, but also concern resource contributor's *payoffs*. 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 (3), where $su(t_{ij})$ and $eu(t_{ij})$ refer to the scheduling utility and execution utility respectively and λ_{ij} is a coefficient customized based on user's expectation.

$$tu(t_{ij}) = \lambda_{ij} \cdot su(t_{ij}) + (1 - \lambda_{ij}) \cdot eu(t_{ij}) \tag{3}$$

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 (4) and Formula (5) respectively, which are also shown in Fig. 2 (a) and Fig. 2 (b) graphically.

$$su(t_{ij}) = \begin{cases} 1 & t_w \le t_0 \\ 1 - k_0(t_w - t_0) & t_0 < t_w \le t_0 + 1/k_0 \\ 0 & t_w > t_0 + 1/k_0 \end{cases} \tag{4}$$

$$eu(t_{ij}) = \begin{cases} 1 & t_e \le t_0 \\ 1 - k_1(t_e - t_1) & t_1 < t_e \le t_1 + 1/k_1 \\ 0 & t_e > t_1 + 1/k_1 \end{cases} \tag{5}$$

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.

In addition, each user should specify a scheduling bid (denoted $sb(t_{ij})$), for his/her task's schedule. Higher scheduling bid implies the cost the user is willing to pay for being scheduled with higher priority, resulting in shorter queuing time. Once the task is scheduled and completed successfully, its user needs to pay the execution node's owner both the payment on scheduling priority and an amount of execution currencies calculated based on the demanded resource amount and per-resource-unit price. We denote by $sp(t_{ij})$ t_{ij}'s final scheduling payment on its scheduling priority. Note that $sb(t_{ij})$ may not be equal to $sp(t_{ij})$ for different auction policies. For example, when t_{ij} wins the bid (i.e. it gives the highest bid among all other competitors), $sp(t_{ij})$=$sb(t_{ij})$ under the first-price sealed-bid policy, while $sp(t_{ij})$=$\max_{\{x,y\} \ne \{i,j\}}(sb(t_{xy}))$ in the second-price sealed-bid policy. For such a definition w.r.t. scheduling bid and execution payment, we could easily distinguish user's quick-scheduling demand (e.g. interactive mode) and slow-scheduling demand (e.g. queuing mode). That is, users could accurately express their expectations in fine granularity, by tuning λ_{ij}, $w(t_{ij})$, $sb(t_{ij})$, and $dr(t_{ij})$.

From resource provider's point of view, each of them tries to maximize its revenue, 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 should be able to assign various prices for their own resources, or via pricing functions of the demanded resource amounts. We use $\beta(p_d)$ to denote the price vector assigned by the owners of the resource node p_d.

The per-time-unit payment (denoted as $C(t_{ij})$) for a task's execution by its user is denoted by Formula (6), where $\beta(t_{ij})$ refers to the resource price adopted by the task t_{ij}. Then, the total payment of this user on this task can be calculated as $C(t_{ij}) \times t_{ij}'s\ execution\ time$.

$$C(t_{ij}) = sp(t_{ij}) + dr(t_{ij})^T \cdot \beta(t_{ij}) \qquad (6)$$

Since either the system or the users themselves cannot exactly predict the execution times for the tasks, it is infeasible for system to accurately predict any task's final total payment before running it. Then, it is non-trivial to directly evaluate the satisfaction level of a user with a preset expectation based on its final real payment. For example, any user may pay more than the original payment amount 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 with their agreement. That is, the user of each task $t_{ij}^{p_d}$ is satisfied if and only if its real payment cost per time unit (i.e. $C(t_{ij}^{p_d})$) is in accordance with Inequality (7), where $B(t_{ij})$ is the user's per-time-unit budget.

$$C(t_{ij}) \leq B(t_{ij}) \qquad (7)$$

The ultimate objective of any participant is to optimize its task utility and profits meanwhile, also leading to a high overall satisfaction level around the whole community. Finally, we also expect such a non-cooperative resource allocation game could maximize the social-welfare on tasks' execution, denoted as aggregated task utility (ATU) (defined in Formula (8)), subject to the constraints (1), (2), (7). It is extremely hard to find the optimal solution with polynomial time complexity to solve this problem, in that we can prove it is NP-complete (please see Appendix A for details). Fig. 2 (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.

$$ATU = \sum_{i=1}^{n} tu(t_{ij}) \qquad (8)$$

III. Social-optimized Win-win Resource Allocation

Our goal is to design a distributed cloud model that can maximize social welfare, i.e. the total sum of task utilities, with win-win effect.

There are two phases for each task scheduling, resource discovery and resource allocation. As for the former, we adopt our designed Proactive Index Diffusion CAN (PID-CAN), which could effectively find available resources for any task around the global systems with mitigated mutual searching contention. As for the latter, we will adopt classical Vickrey auction to realize the resource consumers' truth bidding behaviors, and exploit a reverse Vickrey auction meanwhile to make sure that the resource contributors are also better off truthfully revealing their resources' prices. With respect to the win-win effect, the resource contributors can receive extra amount of payment which is more than their expectations, and the resource consumers' tasks could be finished within their preferred budgets and expected execution performance.

A. Dynamic 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 [16], 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 satisfying the task's demanding vector 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.

B. Dual Vickrey Auction (DVA) Algorithm

The main contribution of this paper is designing a social-optized win-win resource allocation algorithm, namely dual vickrey auction. The pseudo-code is shown in Algorithm 1, where $p_*^{(k)}$ and $\beta_*^{(k)}$ stand for the kth item in $QSET(t_{ij})$ and the corresponding assigned resource price (or pricing function) respectively. $QSET(t_{ij})$ is defined as the set of candidate resources found based on t_{ij}'s demand. This algorithm should run on each individual nodes in the Self-organizing Cloud system. Without loss of generality, suppose it is running on a node p_s as a scheduler. As mentioned previously, the node p_s will receive multiple tasks submitted by users over time, and all of them will be put in the queue q_s, which also contains the old tasks that still have not found matched resources yet. The scheduler of p_s will process q_s's tasks according to the non-increasing order of $sb(t_{ij})$ (i.e. scheduling priority), periodically. From the perspective of the execution node, upon receiving any migrated task, it will split its available resources at multiple dimensions based on the task's demand, update its availability, and run the task over the split resource.

Algorithm 1 DUAL VICKREY AUCTION ALGORITHM

```
1: while (true) do
2:     Sort $q_s$'s tasks in non-increasing order of $sb(t_{ij})$;
3:     for (each task $t_{ij}$ in $q_s$) do
4:         $\max_{sb(t_{xy})\leq sb(t_{ij})}(sb(t_{xy})) \rightarrow sp(t_{ij})$; /*$t_{xy}$ refers to other
           tasks in $q_s$ other than $t_{ij}$*/
5:         Perform PID-CAN to construct $QSET(t_{ij})$ for $t_{ij}$;
6:         Sort all items in $QSET(t_{ij})$ in non-decreasing order of $t_{ij}$'s
           payment based on its resource demand $dr(t_{ij})$;
7:         for (each item $p_*^{(k)}$ in $QSET(t_{ij})$) do
8:             Connect node $p_*^{(k)}$ to confirm its current availability state;
9:             if ($p_*^{(k)}$ is qualified for $t_{ij}$ on Formula (1) and (7)) then
10:                $\beta_*^{(k+1)} \rightarrow \beta(t_{ij})$; /*next lowest price*/
11:                Update $p_*^{(k)}$'s status and execute $t_{ij}$ in a VM on $p_*^{(k)}$;
12:                break;
13:            end if
14:        end for
15:    end for
16:    Sleep a tiny cycle; /*to receive more tasks*/
17: end while
```

For each task (denoted as t_{ij}) selected from q_s, we use PID-CAN protocol to search a set of resource nodes (denoted $QSET(t_{ij})$), whose multi-dimensional resource vectors satisfy the t_{ij}'s expected qualified resource demand $dr(t_{ij})$. Specifically, a query message that contains $dr(t_{ij})$ will be routed on the PID-CAN overlay, until it finds the specific number of qualified resource nodes through the passed duty nodes or the number of hops is greater than a 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). Then, the algorithm will sort the resources in the non-decreasing order of their prices (line 6) and check them them one by one (line 7). That is, the resources with lower prices will have higher priority to be selected (line 8), and the settled payment will be assigned as the next lowest payment from among the set of candidate resources at line 10 (i.e. reverse second-price policy). Such a design is similar to the second-price based pricing policy and serves as the key point of this algorithm, which could make the resource contributors be better off truthfully revealing their true expectations on resource prices. Finally, the task will be migrated and executed on the selected node, with proportional-share resource consumption model [17]. Under such a model, the tasks on the same host will be allocated the multiple resource shares proportional to their demand vector, aiming to make full use of the idle resources.

In addition, such an algorithm could converge to a win-win situation, such that every one satisfies his/her gains. On one hand, from resource consumers' views, their tasks can be executed based on their expected resource demands while their payments are still under their budgets. On the other hand, from resource contributors' point of view, their revenues for contributing their resources to execute other tasks will be more than their original expectations, due to the next-lowest settled payment design (line 10) and the extra amount of allowance (i.e. task's extra payment for paying for contending its scheduling priority). That is, the resource contributors will definitely receive higher payment from the resource contributors based on higher resource prices and additional amount of payment for scheduling priority.

IV. PERFORMANCE EVALUATION

A. Experimental Setting

For our simulation, we first built an emulated credit-scheduler (or proportional-share scheduler) in accordance with the design of XEN [18]. Then, we carefully constructed the CAN protocol [16] using the Peersim tool [19] and improved it using our designed Proactive-Index-Diffusion (PID) strategy [14]. There are thousands of nodes, each with random settings (Table I). 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 elements. Each task's workload along some attribute is set as the product of its random capacity value based on Table I and a demand ratio (denoted by λ ($\leq$1)). Demand ratio is used to set a range within which all tasks' workload are distributed. For example, when λ=0.5, all tasks' workload will be randomly set in the range [0, $0.5 \cdot max\_capacity$] at multiple dimensions. 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 the user tasks will be periodically generated on each node based on Poisson process with 3000 seconds as its mean.

TABLE I
SYSTEM SETTING

Parameter	Value
# of nodes	2000 $\sim$ 12000
# 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 MB/s
memory size per node	512, 1024, 2048, 4096 MB
disk size per node	20, 60, 120, 240 GB
LAN network bandwidth	5 $\sim$ 10 Mibit/s
WAN network bandwidth	0.2 $\sim$ 2 Mibit/s

We mainly focus on five metrics, processed task ratio (both scheduled task ratio and finished task ratio), social welfare (evaluated by average task utility), fairness index [20] of task's execution efficiency, average task utility, and the contributors' average income ratio. 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. Task's execution efficiency (denoted as e_{ij}) is defined as the ratio of its execution time to the

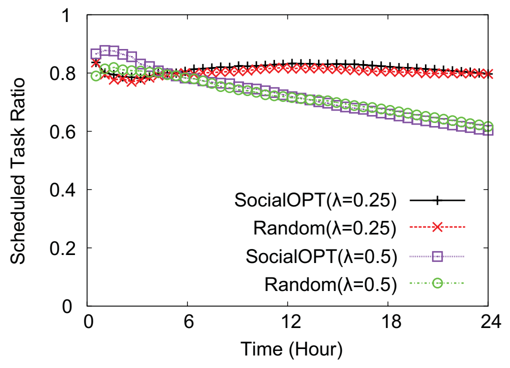


Fig. 3. Scheduled Task Ratio

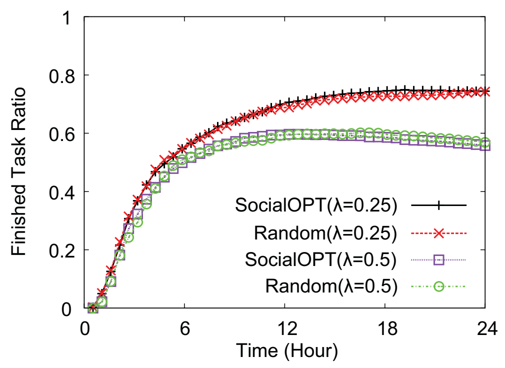


Fig. 4. Finished Task Ratio

theoretical value estimated using average computing ability. Hence, the overall fairness index of task's execution efficiency (denoted as φ) can be calculated by Equation (9), where m_i means the number of tasks submitted to node p_i.

$$\varphi = \frac{(\sum_{i=1}^{n} \sum_{j=1}^{m_i} e_{ij})^2}{(\sum_{i=1}^{n} m_i) \cdot (\sum_{i=1}^{n} \sum_{j=1}^{m_i} e_{ij}^2)} \tag{9}$$

The finished task ratio is equal to the number of finished tasks and that of total tasks submitted. The average task utility indicates the average value of task utility for all finished tasks. The average contribution income ratio is calculated by $\frac{1}{n}\sum_{d=1}^{n}\frac{I_{ct}^{real}(p_d)}{I_{ct}^{expect}(p_d)}$, where $I_{ct}^{real}(p_d)$ and $I_{ct}^{expect}(p_d)$ denote the resource owner's final income and its expected income (evaluated using its own assigned prices) respectively.

Our experiments are conducted under different competitive situations with various workloads of submitted tasks. As a comparative baseline, we also implement another node-selection strategy, namely random-selection strategy, which was adopted in our previous work [14]. Under such a strategy randomly, each scheduler node selects one execution node from the qualified candidate resources queried by PID-CAN protocol [14], in order to alleviate the decision conflict among the resource requesters. This random-selection design also delivers satisfactory throughput based on our previous work [14], but with degraded contributors' payoffs observed in this paper, easily mitigating their participating motivations.

B. 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 compare our new method, i.e. dual vickrey auction algorithm (SocialOPT) to the random-selection method on the scheduled task ratio and finished task ratio in Fig. 3 and Fig. 4, respectively. We could observe that the throughput of SocialOPT is no worse than that of the random-selection design in our previous work [14], no matter for the scheduled tasks or finished tasks. Fig. 5 validates that the fairness indexes of the tasks' execution efficiencies under the two algorithms converge to the same level, which means tasks can be treated fairly around the whole system. Accordingly, our SocialOPT approach will not degrade the task processing ratios and fairness 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 SocialOPT design will significantly outperform the random-selection approach on the social welfare and contributor's average income.

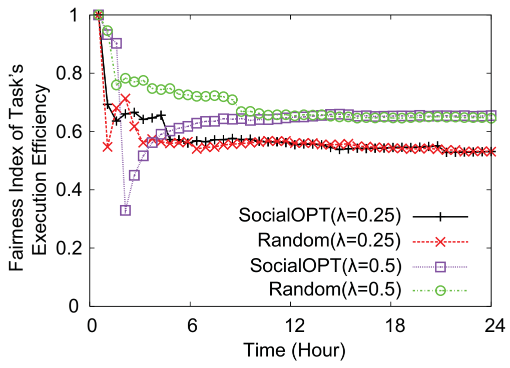


Fig. 5. Fairness Index of Task Execution Efficiency

Fig. 6 presents the average task utility (i.e. social welfare) around the global system. We observe that our SocialOPT solution (i.e. dual vickrey auction) under different always performs (with ATU up to about 0.98 and 0.95) at least no worse than random node-selection strategy (with ATU up to about 0.96 and 0.93), which used to serve as the best solution in our previous work [14]. A minor improvement can be observed in the SocialOPT solution is due to the priority-aware design in scheduling tasks, which can improve task's scheduling utility.

From Fig. 7, we could observe that the contributors' average income could be up to 1.2 times of their expectations under SocialOPT approach, and converges to 1 over one-day execution. Such a situation that contributors' incomes are greater than their expectations is due to the extra allowance (i.e. scheduling payment provided by users) in our reverse-vickrey auction design (i.e. line 10 in Algorithm 1). In contrast, under the random-node-selection approach, contributors are probably disappointed for the much lower incomes gained: the average contribution income ratio will converge to only 0.2 and 0.1 when demand ratio is set to 0.25 and 0.5 respectively.

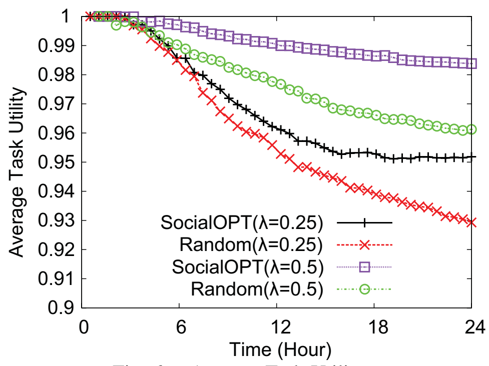


Fig. 6. Average Task Utility

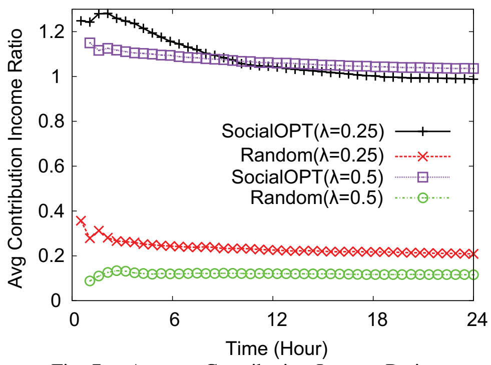


Fig. 7. Average Contribution Income Ratio

V. Related Work

Compared to the traditional task resource allocation problem, we emphasize economic theories that are essential for improving participants' motivations, in that contributors with little reward may probably flee, causing collapse to the whole system. Existing economy-based resource allocation mechanisms could be classified into three categories:

Reciprocation-Based Economy (RBE): Reciprocation-based incentive mechanisms, a.k.a. Network of Favors (NoF), have been proposed to deal with the free-riding problem in a P2P grid environment [21]. In RBE model, each node always donates its service to others based solely on the record of their past bilateral inverted service actions. So, the nodes which contribute more will get more in return when they make requests. Such a model is not flexible since there are no currencies to coordinate interests between two sides. The examples are OurGrid [22] and SHARP [23].

Nash Bargaining Solution (NBS) [24]: NBS is different from RBE in that any requester must make an agreement on a price by negotiating with another supplier before the consumption. Although NBS may ensure demand and supply match reciprocally, the matching procedure is relatively low-efficient because the tasks cannot be started/executed without a couple of bargaining rounds. The example projects include Nimrod-G [25] and Mobile Grid [26].

Auction-Based Solution (ABS): In ABS, each consumer has to submit bids for the needed resources to one or more auctioneer nodes. ABS can be further classified into four categories, English auction [11], Dutch auction [12], first-price and second-price sealed auction, and double auction [27]. Since the first two auctions are carried out completely open to the competitors at any time, they are time-consuming and potentially lift up the bid price the winner has to pay due to the explicit competition. The sealed auction differs in that the bids are always not open. In such an auction, each supplier receives a number of sealed bids from resource requesters and the bidder offering the highest price is the winner. Mirage [28] adopts the first-price policy (i.e. the payment of bidder is the highest price) but it is argued that the requesters may have the incentives of bidding the prices smaller than the real value they regard. Bellagio [29] thereby switched to the second-price policy (a.k.a. Vickrey auction [13]) in which the winner will pay the second highest bidding price, in order to ensure the strategy-proof property which may implicitly urge users to reveal their real demands. All the above auctions belong to single-sided auction. In comparison, double-sided auction (a.k.a. double auction) also allows resource providers to specify expected prices for their resources, leading to an agreement from both sides. The example projects are SCDA [30] and Coordinated P2P Grid [31] (ticket match strategy). Compared to ABS, our solution also considers the win-win effect such that each participant is satisfied with its payoff and social-optimum with maximized total welfare into one algorithm, as well as the incentives of truthful behaviors on demand revealing from both sides (consumers and contributors), which is the first attempt to our knowledge.

VI. Conclusion and Future Work

This paper proposes a novel win-win resource allocation, which could guarantee that resource consumers are satisfied with their tasks' execution and the resource contributors are also content with their payoffs for their resource-provisioning. Moreover, by extending the traditional second-price bidding policy to a novel double-sided next-price bidding policy, our algorithm can induce both strategic resource consumers and contributors to truthfully reveal their demands (either on resource capacity or resource prices). Finally, we confirm the efficiency of our design via event-driven simulation. In the future, we may 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 a HKU Basic Research grant (Grant No. 10401460), and a Hong Kong UGC Special Equipment Grant (SEG HKU09).

Appendix A

Theorem 1: With global complete information, maximizing ATU s.t. constraints (1), (2), (7) is an NP-complete problem.

Proof: First of all, this problem belongs to NP set because any one resource allocation could be verified in polynomial time. We will prove it is NP-complete problem.

With Equation (3) and (8), the objective can be converted to "maximizing $\sum_{i=1}^{n}(\lambda_{ij}\cdot su(t_{ij})+(1-\lambda_{ij})\cdot eu(t_{ij}))$". Consider a special case: $\forall i,j$, let λ_{ij}=1, then the objective becomes "maximizing $\sum_{i=1}^{n} su(t_{ij})$". In addition, suppose the execution time of every task is sufficiently long and k_0 is extremely large, then the problem can be simplified to "maximizing the number of tasks whose minimum demands can be met (i.e. minimizing the number of waiting tasks), given a set of divisible resources", and we call it *min-number* problem in following text. Note that the resource here is divisible, thus any task can multiplex nodes with other tasks as long as its received share satisfies its minimum demand. As follows, we will prove the min-number problem is reducible from another NP-complete problem, bin-packing.

The bin-packing problem is generally stated as: given a set of items of different sizes and a set of bins of the same capacity, try to put all items into as few bins as possible. As follows, we will show a polynomial algorithm to map the bin-packing problem to the min-number problem, thereby, the min-number problem is also NP-complete..

Without loss of generality, denote by *TSet*($task_1$, $task_2$, $\cdots$, $task_m$) the given set of m tasks each with a different minimum resource demand sorted in non-increasing order (i.e. higher demand comes first). Denote by *PSet*(p_1, p_2, $\cdots$, p_n) the set of n nodes. Suppose there exists a solution (TM or non-TM, denoted by *BSOL*(*TSet*)) for the bin-packing problem, with the minimum number of bins as a return value. Then, we can solve the min-number problem using a few steps of *BSOL*(*TSet*), as shown in Algorithm 2.

Algorithm 2 Polynomial Reduction from Bin-packing solution

```
1: $n' = BSOL(TSet(task_1, task_2, \cdots, task_m))$;
2: if ($n' \le n$) then
3:     Output $TSet(task_1, task_2, \cdots, task_m)$;
4: else
5:     for ($i$=1$\rightarrow m$) do
6:         $n' = BSOL(TSet(task_{i+1}, \cdots, task_m))$;
7:         if ($n' \le n$) then
8:             Output $TSet(task_{i+1}, \cdots, task_m)$;
9:             break;
10:        end if
11:    end for
12: end if
```

■

References

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

[2] D. P. Anderson, "Boinc: a system for public-resource computing and storage," in *Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*, 2004, pp. 4–10.

[3] G. Fedak, C. Germain, V. Neri, and 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.

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

[5] O. Nov, D. Anderson, and O. Arazy, "Volunteer computing: a model of the factors determining contribution to community-based scientific research," in *WWW*, 2010, pp. 741–750.

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

[7] G. Briscoe and 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.

[8] Z. Xu, S. Di, L. Cheng, and C.-L. Wang, "Wavnet: Wide-area network virtualization for elastic cloud computing," in *40th International Conference on Parallel Processing (IEEE ICPP2011)*, 2011, pp. 285–294.

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

[10] B. An, V. Lesser, D. Irwin, and M. Zink, "Automated negotiation with decommitment for dynamic resource allocation in cloud computing," in *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, Toronto, 2010, pp. 981–988.

[11] R. P. McAfee and J. McMillan, "Auctions and bidding," *Journal of Economic Literature*, vol. 25, no. 2, pp. 699–738, 1987.

[12] L. S. Bagwell, "Dutch auction repurchases: An analysis of shareholder heterogeneity," *The Journal of Finance*, vol. 47, no. 1, pp. 71–105, 1992.

[13] H. R. Varian, "Position auctions," *International Journal of Industrial Organization*, vol. 25, pp. 1163–1178, 2006.

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

[15] L. Huang, J. Jia, B. Yu, B.-G. Chun, P. Maniatis, and 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] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network," in *ACM SIGCOMM Computer Communication Review*, New York, NY, USA, 2001, pp. 161–172.

[17] M. Feldman, K. Lai, and L. Zhang, "The proportional-share allocation market for computational resources," *IEEE Transactions on Parallel and Distributed Systems*, vol. 20, pp. 1075–1088, 2009.

[18] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and 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.

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

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

[21] N. Andrade, F. Brasileiro, W. Cirne, and M. Mowbray, "Discouraging free riding in a peer-to-peer CPU-sharing grid," in *Proceedings of 13th IEEE International Symposium on High Performance Distributed Computing*, 2004, pp. 129–137.

[22] N. Andrade, W. Cirne, F. Brasileiro, and P. Roisenberg, "Ourgrid: An approach to easily assemble grids with equitable resource sharing," in *Job Scheduling Strategies for Parallel Processing*, 2003, pp. 61–86.

[23] Y. Fu, J. Chase, B. Chun, S. Schwab, and A. Vahdat, "SHARP: an architecture for secure resource peering," in *Proceedings of the 19th ACM Symposium on Operating Systems Principles*, New York, NY, USA, 2003, pp. 133–148.

[24] K. Binmore, A. Rubinstein, and A. Wolinsky, "The nash bargaining solution in economic modelling," *RAND Journal of Economics*, vol. 17, pp. 176–188, 1986.

[25] D. Abramson, R. Buyya, and J. Giddy, "A computational economy for grid computing and its implementation in the nimrod-g resource broker," *Journal of Future Generation Computer Systems*, vol. 18, no. 8, pp. 1061–1074, 2002.

[26] P. Ghosh, N. Roy, S. K. Das, and K. Basu, "A game theory based pricing strategy for job allocation in mobile grids," *International Parallel and Distributed Processing Symposium*, vol. 1, p. 82a, 2004.

[27] R. B. Myerson and M. A. Satterthwaite, "Efficient mechanisms for bilateral trading," *Journal of Economic Theory*, vol. 29, no. 2, pp. 265–281, 1983.

[28] B. Chun, P. Buonadonna, A. AuYoung, C. Ng, D. Parkes, J. Shneidman, A. Snoeren, and A. Vahdat, "Mirage: A microeconomic resource allocation system for sensornet testbeds," in *The 2nd IEEE Workshop on Embedded Networked Sensors*, 2005, pp. 19–28.

[29] A. Auyoung, B. Chun, A. Snoeren, and A. Vahdat, "Resource allocation in federated distributed computing infrastructures," in *Proceedings of 1st Workshop on Operating System and Architectural Support for On-demand IT InfraStructure*, 2004.

[30] Z. Tan and J. Gurd, "Market-based grid resource allocation using a stable continuous double auction," in *The 8th IEEE/ACM International Conference on Grid Computing*, 2007, pp. 283–290.

[31] R. Ranjan, A. Harwood, and R. Buyya, "Coordinated load management in peer-to-peer coupled federated grid systems," Grid Computing & Distributed Systems Laboratory, University of Melbourne, Tech. Rep., 2008.