Неманипулируемый для пользователей механизм распределения процессорного времени между неделимыми заданиями

А.А. Замятин

Новосибирский государственный университет

Задачу распределения ресурсов между пользователями в многопроцессорной системе можно решать с помощью экономических механизмов. J. Stößer, D. Neumann и C. Weinhardt в [1] предложили механизм, который является слабо бюджетно-сбалансированным, индивидуально рациональным, обеспечивает неманипулируемость для пользователей, но не является эффективным. Цель работы: модифицировать правило ценообразования так, чтобы новый механизм при той же оценке трудоемкости был в любой исходной ситуации не менее эффективен, а в некоторых случаях — более эффективен, чем исходный механизм.

1. Введение

ИТ-отделам крупных компаний требуется поддерживать работоспособность трудоемких приложений, которые становятся все более требовательными к вычислительным ресурсам. Чтобы система выдерживала пиковые нагрузки, приходится содержать значительное количество вычислительных ресурсов, которые большую часть времени бездействуют**.** Результаты исследований показали, что корпоративные центры обработки данных используют от 10% до 35% вычислительной мощности [2].

Эффективный путь снижения затрат на вычислительные ресурсы — использование распределенных вычислений, позволяющее организациям своевременно реагировать на пиковые нагрузки и арендовать требующиеся ресурсы. Рынок вычислительных услуг позволяет предприятию содержать оборудование, рассчитанное на базовую нагрузку, а пиковые нагрузки обслуживать с помощью внешних сетевых ресурсов.

Для вычислительной системы, оказывающей платные услуги, главной проблемой является распределение ресурсов и назначение цен, то есть определение того, какой ресурс в какое время и по какой цене предоставляется каждому пользователю.

Решать эту проблему можно с помощью экономического механизма распределения ресурсов. Дизайн экономических механизмов – это конструктивный подход, позволяющий создать такой механизм взаимодействия, при котором эгоистические действия каждого из агентов в сумме приведут к решению, оптимальному с точки зрения общей целевой функции [3]. Экономический механизм распределения вычислительных ресурсов есть набор правил взаимодействия между пользователями, арендующими вычислительные ресурсы, и провайдерами — арендодателями.

Для успешного применения в сетях желательно, чтобы экономический механизм обладал рядом свойств. В [1] описаны следующие свойства.

Двусторонний рынок. Двусторонние рынки (двусторонние сети) — сетевые рынки, которые имеют две группы участников (в нашем случае это пользователи и провайдеры) с возникновением сетевых эффектов между ними. Механизм должен обеспечивать заключение сделок между множеством пользователей и множеством провайдеров. Это свойство является обязательным для сетей, которые по определению состоят из ресурсов, не подчиненных централизованному контролю.

Вычислительная приемлемость. Механизм должен распределять задания по вычислительным узлам и устанавливать цены за полиномиальное время относительно размера входных данных, т.е. суммарного количества заявок спроса и предложения.

Комплектация. Пользователи должны иметь возможность описывать зависимость между несколькими заданиями и вычислительными узлами и указывать технические ограничения сво-

их заданий, что сужает множество вычислительных узлов, на которые эти задания могут быть распределены [4].

Временные ограничения. Предполагается, что пользователи знают желательные сроки выполнения своих заданий, а провайдеры время доступности своих вычислительных узлов. Механизм должен учитывать сформулированные участниками временные ограничения при распределении ресурсов.

Эффективность. Механизм является эффективным, если при любых входных данных он всегда генерирует распределение, максимизирующее суммарный излишек (общее благосостояние) участников взаимодействия.

Бюджетная сбалансированность. Механизм называется слабо бюджетно сбалансированным, если его платежная схема не нуждается в субсидировании извне. Платежи пользователей полностью покрывают платежи, которые получают провайдеры.

Индивидуальная рациональность (добровольность). Механизм обладает свойством индивидуальной рациональности, если участники не могут понести убытки от участия в механизме. В частности, пользователи не должны вносить плату за участие, не получая при этом ресурсов.

Неманипулируемость. Механизм называется неманипулируемым, если слабо доминирующей стратегией каждого участника является сообщение истинных параметров своего задания или вычислительного узла при произвольных параметрах других участников. Манипулируемый механизм не может гарантировать эффективность, так как распределение, эффективное при значениях параметров, сообщенных участниками, скорее всего не будет эффективным для истинных данных. Поэтому неманипулируемость является желательным свойством.

Экономический механизм состоит из трех основных компонентов [5]: правил описания заявок, алгоритма распределения ресурсов и схемы платежей. Правила описания заявок задают множества стратегий участников. Алгоритм распределения определяет, какое задание, на каком узле и в какое время выполняться, а схема платежей по результатам распределения назначает платежи пользователей провайдерам.

Существуют теоретические результаты, показывающие недостижимость некоторых комбинаций свойств экономического механизма [6]. В частности, из теоремы Майерсона-Сэттерсвэйта следует, что при естественных предположениях неманипулируемый механизм не может быть одновременно эффективным, бюджетно-сбалансированным и индивидуально рациональным. Так как последние два свойства необходимы для устойчивости механизма, жертвовать приходится или неманипулируемостью, или эффективностью. При большом количестве входных данных конфликтуют вычислительная приемлемость и эффективность, так как для вычисления оптимального распределения требуется много времени.

J. Stößer, D. Neumann, C. Weinhardt в [1] предложили механизм (назовем его SNW-механизм), который распределяет ресурсы между заданиями (этап 1) и определяет платежи пользователей (этап 2). Этот механизм является слабо бюджетно-сбалансированным, индивидуально рациональным и обеспечивает неманипулируемость для пользователей. Однако SNW-механизм не является эффективным, то есть не максимизирует суммарную полезность, полученную всеми участниками взаимодействия (так называемый, суммарный излишек).

Цель работы: в SNW-механизме модифицировать правило ценообразования так, чтобы новый механизм при той же оценке трудоемкости в любой исходной ситуации обеспечивал суммарный излишек не меньший, а в некоторых случаях больший, чем SNW-механизм.

2. SNW-механизм

2.1 Описание модели

Задан горизонт планирования, промежуток времени T_0, на котором решается задача распределения ресурсов. Он разделен на промежутки равной длины (назовем их единичными промежутками, е.п.) с номерами $1,\ldots,T$.

N — множество номеров заявок от вычислительных узлов с предложениями ресурсов на период $[0,T_0]$. Чтобы сделать предложение провайдер узла $n \in N$ должен отправить системе заявку вида $\eta_n = (V_n, C_n, M_n, R_n, E_n)$. Здесь: $V_n \in \mathrm{R}_+$ — себестоимость использования единицы компьютерной мощности в течение одного е.п. (единицы мощности-времени в терминологии авто-

ров) в денежном эквиваленте; $C_n \in \mathrm{N}$ и $M_n \in \mathrm{N}$ — максимально доступные мощность и память вычислительного узла соответственно; R_n и E_n, $1 \le R_n \le E_n \le T$, — такие е.п., что узел n доступен в единичные промежутки с номерами $R_n,\ldots, E_n$; V_n — минимальная цена, по которой провайдер согласится сдать в аренду узел (отправная цена). Вычислительная мощность C_n может измеряться, например, числом циклов ЦПУ (тактов) за е.п, а память M_n — в мегабайтах, гигабайтах и т.п.

J — множество номеров заявок на выполнение заданий. Пользователь, желающий выполнить задание $j \in J$, отправляет заявку вида $\theta_n = (v_n, c_n, m_n, r_n, e_n)$, где: $v_j \in \mathrm{R}_+$ — максимальная цена, которую пользователь готов заплатить за единицу мощности-времени; $c_j \in \mathrm{N}$ и $m_j \in \mathrm{N}$ — минимально требующиеся заданию за один е.п. количества мощности и памяти соответственно; r_j и e_j, $1 \le r_j \le e_j \le T$, — такие номера е.п, что задание j должно выполняться в единичные периоды $r_j,\ldots, e_j$. Время выполнения задания j, выраженное числом единичных промежутков, обозначим $d_j = e_j - r_j + 1$. Тогда денежную оценку полезности (бюджет) задания можно посчитать по формуле $b_j = v_j c_j d_j$.

Множества заявок провайдеров и потребителей обозначим соответственно $\eta = \{\eta_n \mid n \in N\}$ и $\theta = \{\theta_j \mid j \in J\}$.

Предполагается, что вычислительный узел может выполнять несколько заданий одновременно, а задание не может выполняться одновременно на разных вычислительных узлах, но может мигрировать между ними без необходимости перезапуска, то есть в различные промежутки времени может выполняться на различных узлах.

Требуется распределить задания по вычислительным узлам и определить платежи пользователей провайдерам. Решение задачи включает два этапа: распределительный и оценочный.

Положим $X = (x_{jnt} \mid j \in J, n \in N, t \in [r_j, e_j] \cap [R_j, E_j])$, где

$$x_{jnt} = \begin{cases} 1, \text{ если задача } j \text{ выполняется на узле } i \text{ в е.п. } t, \\ 0 \text{ иначе.} \end{cases}$$

Пусть W — общее благосостояние, которое определяется как разность суммы денежных оценок полезности, полученной пользователями, и суммарных затрат провайдеров. W есть суммарный излишек участников механизма:

$$W = \sum_j c_j \sum_n \sum_t x_{jnt}(v_j - V_n). \qquad (*)$$

Проблему распределения заданий по узлам можно представить в виде задачи целочисленного линейного программирования:

$$W_{\max} = \max_X \sum_j c_j \sum_n \sum_t x_{jnt}(v_j - V_n), \qquad (1)$$

$$\sum_n x_{jnt} \le 1 \text{ для всех } j \text{ и } t \in [r_j, e_j], \qquad (2)$$

$$\sum_j x_{jnt} c_j \le C_n \text{ для всех } n \text{ и } t \in [R_n, E_n], \qquad (3)$$

$$\sum_j x_{jnt} m_j \le M_n \text{ для всех } n \text{ и } t \in [R_n, E_n], \qquad (4)$$

$$\sum_{u=r_j}^{e_j} \sum_n x_{jnu} = d_j \sum_n x_{jnt} \text{ для всех } j \text{ и } t \in [r_j, e_j], \qquad (5)$$

$$x_{jnt} \in \{0,1\} \text{ для всех } j, n, t \in [r_j, e_j] \cap [R_n, E_n],\ V_n \le v_j. \qquad (6)$$

Ограничения (2) - (6) имеют следующую интерпретацию. Неравенство (2) гарантирует, что задание не будет запланировано на разных узлах одновременно. Неравенства (3) и (4) требуют, чтобы распределенные на вычислительный узел задания не запрашивали больше ресурсов, чем этот узел может предоставить. Равенство (5) обеспечивает выполнение условия неделимости заданий: задание либо выполнится полностью, либо не будет выполняться вообще. Наконец, ограничение (6) сокращает количество возможных распределений в соответствии с указанными участниками временными интервалами, а также обеспечивает, что задание не будет размещено на узле, отправная цена которого больше, чем пользователь готов заплатить.

Описанная задача является гибридом задачи об упаковке в контейнеры и обобщенной задачи о назначениях [7]. Нетрудно заметить, что задача (1) – (6) не менее трудоемка, чем задача об упаковке в контейнеры, так как функция благосостояния зависит от отправных цен вычислительных узлов, а в задаче об упаковке в контейнеры стоимость использования всех контейнеров равна нулю. Кроме того, постановка задачи осложнена наличием двух различных ресурсов, а главное, ограничение (5) связывает между собой распределительные задачи всех временных промежутков, каждая из которых по отдельности сложнее задачи об упаковке в контейнеры. Так как задача об упаковке в контейнеры является NP-полной [8], задача (1) – (6) также NP-полна. Следовательно, вычисление оптимального распределения может оказаться очень трудоемким. Но для нормального функционирования системы распределение должно происходить как можно быстрее.

2.2 Алгоритм распределения ресурсов

Чтобы обойти вычислительную сложность, в [1] предложено на этапе 1 работы SNW-механизма использовать жадный эвристический алгоритм. Идея алгоритма состоит в том, чтобы на каждом шаге максимизировать разность $x_{jnt}(v_j - V_n)$, содержащуюся в функции общего благосостояния. Приведем этот алгоритм.

Алгоритм 1. Жадный эвристический алгоритм распределения ресурсов.

Шаг 0. Сортируем задания $j \in J$ в порядке невозрастания v_j. Сортируем вычислительные узлы $n \in N$ в порядке неубывания V_n. Упорядоченные множества J и N будем в дальнейшем называть очередями. Пусть $j(k)$ — номер задания, стоящего на месте k в очереди J.

Шаг k. Последовательно пытаемся разместить задание $j(k)$ на вычислительных узлах очереди N, начиная с первого. То есть, для задания $j(k)$ на каждом временном промежутке $t \in \{r_{j(k)},\ldots, e_{j(k)}\}$ подбираем первый в очереди N вычислительный узел, удовлетворяющий техническим характеристикам задания. Если хотя бы на одном промежутке не удалось распределить задание, то, в соответствии с ограничением (5), задание не попадет в распределение и не будет выполнено. Если же для каждого временного промежутка найдется подходящий вычислительный узел, то задание будет распределено. Тогда на соответствующих узлах резервируются указанные в заявке задания количества ресурсов.

Конец алгоритма.

Заметим, что в худшем случае общее благосостояние, полученное данным алгоритмом, может быть в сколько угодно раз меньше благосостояния при оптимальном распределении [1]. И все же в среднем жадный эвристический алгоритм обеспечивает близкие к оптимальным распределения. На смоделированных случайных начальных данных алгоритм показал эффективность около 95%, то есть $W / W_{opt} \approx 0{,}95$ [1]. При этом алгоритм имеет полиномиальную трудоемкость, делая механизм вычислительно приемлемым.

Кроме того, в [1] утверждается, что при некоторых технических правилах функционирования рынка жадный эвристический алгоритм распределения ресурсов существенно ограничивает стратегическое поведение как пользователей, так и провайдеров. А именно, для участников рынка доминирующей стратегией является сообщение истинных технических характеристик задания (c_j, m_j для пользователя или C_n, M_n для провайдера). Пользователю невыгодно занижать технические характеристики задания, так как в сетевых системах обычной политикой является прерывание приложений, запрашивающих больше ресурсов, чем заявлено. Очевидно, и завышать характеристики не имеет смысла: в этом случае пользователь не только заплатит больше за выполнение задания, но и уменьшит число узлов, подходящих заданию по техническим характеристикам. Для провайдера завышение характеристик вычислительного узла может привести к тому, что назначенным на этот узел заданиям не хватит ресурсов для корректной работы (что грозит штрафом провайдеру), а занижение характеристик приведет к тому, что незаявленные мощности вычислительного узла будут простаивать.

2.3 Платежная схема SNW-механизма

Рассмотрим этап 2 работы SNW-механизма: назначение платежей. Предложенная авторами платежная схема — оценивание по критическому значению. Идея алгоритма состоит в том, что

каждое задание должно заплатить за единицу мощности-времени минимальную цену, при которой механизм включил бы его в распределение. Обозначим эту цену $\varphi_j(\eta, \theta)$. Тогда задание j должно заплатить $p_j = \varphi_j c_j d_j$, если будет выполнено, и $p_j = 0$ иначе. Главное преимущество оценивания по критическому значению состоит в том, что оно, как доказано в [1], обеспечивает неманипулируемость по параметрам v_j.

Для $j \in J$ положим $\theta_{-j} = \theta \setminus \{j\}$. Неманипулируемость механизма по v_j означает, что

$$u_j(\theta_j, \tilde{\theta}_{-j}, \tilde{\eta} \mid \theta_j) \geq u_j(\tilde{\theta}_j, \tilde{\theta}_{-j}, \tilde{\eta} \mid \theta_j) \text{ для всех } j \in J .$$

Приведем алгоритм вычисления критических значений для заданий.

Алгоритм 2. Вычисление критических значений.

Шаг $k \geq 1$. Если задание k не было распределено, переходим к шагу $k + 1$, так как для нераспределенного задания искать критическое значение не нужно, его платеж равен нулю. Иначе начинаем распределять задания по жадному эвристическому алгоритму, исключив из очереди задание k. После распределения каждого задания проверяем, осталось ли достаточно ресурсов для распределения задания k. Возможны два исхода.

Если осталось достаточно ресурсов для распределения задания k, переходим к следующему заданию в очереди и продолжаем распределение. Если очередь закончилась, задание k должно платить по отправным ценам вычислительных узлов, на которые оно назначено,

$$p_k = c_k \sum_n \sum_{t=r_k}^{e_k} x_{kmt} V_n .$$

Если же для задания k недостаточно ресурсов, то $\varphi_k = v_j$, где j – последнее распределенное задание из очереди, и $p_k = v_j c_k d_k$.

Конец алгоритма.

Фактически, чтобы определить критическое значение задания j, требуется частично построить распределение заданий без j. Нахождение критических значений является наиболее трудоемкой процедурой при реализации механизма, и все же выполняется за полиномиальное время благодаря эвристическому алгоритму распределения. Распределение выполняется за полиномиальное время относительно размера входных данных: сортировочный этап выполняется за время $O(|J|\ln|J|)$ и $O(|N|\ln|N|)$, а непосредственно распределение — за $O(|J||N|)$. Процесс нахождения критических значений всех заданий имеет трудоемкость $O(|J|^2|N|)$.

В [1] доказано, что при использовании жадного эвристического алгоритма распределения не существует такой схемы платежей, которая обеспечивала бы неманипулируемость в отношении отправных цен вычислительных узлов. Предложено распределять платежи между провайдерами пропорционально количеству предоставленной вычислительной мощности (основной ресурс). Авторы считают, что этот способ ограничивает стратегическое поведение провайдеров. Общая прибыль провайдеров равна

$$S(\theta, \eta) = \sum_j p_j(\theta, \eta) - \sum_j \sum_n \sum_t x_{jmt} c_j V_n .$$

Если распределять прибыль пропорционально количеству арендованной пользователями вычислительной мощности, то платежи провайдерам будут равны

$$P_n = \sum_j \sum_t x_{jmt} c_j V_n + S(\theta) \frac{\sum_j \sum_t x_{jmt} c_j}{\sum_j \sum_{m \in N} \sum_t x_{jmt} c_j} .$$

Подведем итог. SNW-механизм обладает свойствами двойного рынка, вычислительной приемлемости, комплектации, временных ограничений, бюджетной сбалансированности, индивидуальной рациональности, является частично неманипулируемым (для пользователей), но не является эффективным. Поиск эффективного распределения является NP-полной задачей, а так как для использования механизма на практике вычислительная приемлемость обязательна, приходится довольствоваться эвристическим приближением к эффективному распределению.

3. Ценообразование в соответствии с обобщенным механизмом Викри

Как отмечено выше, для нахождения критического значения задания требуется частично построить распределение без этого задания. При этом может оказаться, что $W_{-j} > W$, где W_{-j} — общее благосостояние при распределении без задания j. То есть, на оценочном этапе может быть обнаружено лучшее распределение, чем найденное на распределительном этапе.

Последовательное выполнение распределительного и оценочного этапов в описанном выше механизме не позволяет корректировать уже полученное распределение. Поэтому мы предлагаем динамически корректировать распределение, используя цены, порождаемые обобщенным механизмом Викри [9]. Для распределения будем использовать изложенный выше жадный алгоритм из [1]. Принцип ценообразования состоит в том, что платеж назначенного задания j равен денежной оценке полезности, которую другие задания «теряют» из-за присутствия задания j. Другими словами, платеж равен разности суммарного излишка при распределении без задания j и суммарного излишка при распределении с заданием j за вычетом полезности, полученной заданием j, то есть $p_j = W_{-j} - (W - v_j c_j d_j)$. При этом задание j, для которого $W_{-j} > W$, окажется неплатежеспособным, будет превышен его бюджет. В таком случае механизм корректирует распределение.

Приведем алгоритм назначения цен с динамическим изменением распределения.

Алгоритм 3. Ценообразование по Викри с корректировкой распределения.

Шаг 0. Имеется X_0 — начальное распределение, полученное жадным эвристическим алгоритмом, а также отсортированные списки заданий J (в порядке невозрастания v_j) и вычислительных узлов N (в порядке неубывания V_n). Пусть W_0 — суммарный излишек при распределении X_0. Положим $X := X_0$, $W := W_0$. Будем говорить, что X — текущее распределение. Введем пустую очередь F, включающую задания, платежи которых уже назначены, а также пустую очередь D, в которую будем помещать «вытесненные» задания.

Шаг $m \geq 1$. Если очередь J пуста, то присваиваем $J := D$, $D := \{\varnothing\}$. Выбираем первое в очереди J задание (пусть его номер равен k). Если это задание не входит в текущее распределение, перемещаем его в конец очереди F, полагаем $p_k := 0$ и переходим к следующему шагу. Иначе производим распределение по жадному эвристическому алгоритму для очереди $F+J+D$ (пропуская этап сортировки, и распределяя сначала задания из F, затем из J, и затем из D) без задания k и находим общее благосостояние W_{-k} при таком распределении. Возможны два исхода.

Если $W_{-k} \leq W$, то назначаем платеж $p_k = W_{-k} - W + v_k c_k d_k$ и перемещаем задание j в конец очереди F.

Если $W_{-k} > W$, то задание k неплатежеспособно, т.к. $p_k > b_k = v_k c_k d_k$, платеж по обобщенному механизму Викри превышает бюджет задания. Назначаем построенное распределение X_{-k} (без задания k) текущим, $X := X_{-k}$, $W := W_{-k}$. Задание k «вытесняется», перемещаем его в конец очереди D.

Алгоритм завершается, когда все задания перейдут в очередь F (при этом очереди J и D будут пусты).

Конец алгоритма.

Результатом выполнения алгоритма будет распределение X, которое полностью задается очередью F (если произвести распределение по жадному эвристическому алгоритму без шага сортировки для очереди F, получим распределение X), и вектор платежей $(p_j \mid j \in J)$. Заметим, что после того, как заданию назначен платеж и оно перешло в очередь F, его позиция в распределении не изменяется. То есть на каждом шаге имеется частичное распределение (заданий из очереди F). Поэтому нет необходимости на каждом шаге заново вычислять распределение, достаточно хранить в памяти частичное распределение заданий из F.

Описанный механизм распределения ресурсов и назначения платежей обозначим GA+GVA (Greedy Algorithm + Generalized Vickrey Auction).

Теорема 1. GA+GVA выполняется за конечное число шагов при любых начальных данных.

Доказательство. Проведем доказательство от противного. Предположим, что при некоторых данных алгоритм не выполняется за конечное количество шагов (зацикливается). Условие завершения алгоритма — переход всех заданий в очередь F. Заметим, что задание, попав в очередь F, остается там и не переходит в другие очереди в процессе выполнения алгоритма. По-

этому бесконечная работа алгоритма означает, что после какого-то шага задания перестанут попадать в очередь F.

Следовательно, все задания очереди J с некоторого момента переходят только в очередь D. Когда кончается очередь J, выполняются присвоения $J := D$, $D := \{\varnothing\}$ (что не меняет общую очередь $F+J+D$), и далее задания снова переходят в очередь D. Задание k переходит из J в D при условии, что $W_{-k} > W$. То есть на каждом следующем шаге общее благосостояние возрастает. А так как количество распределений, достижимых жадным эвристическим алгоритмом при произвольно заданной очереди заданий, конечно (и оценивается числом $J!$), то конечно и множество возможных значений функции общего благосостояния. Следовательно, переход заданий в очередь D может произойти только конечное число раз. Противоречие. Теорема доказана.

Теорема 2. Распределение заданий, порожденное механизмом GA+GVA, обеспечивает суммарный излишек, не меньший, а в некоторых случаях — больший, чем распределение, построенное SNW-механизмом.

Доказательство. Механизм GA+VGA начинает работу с распределения X_0, построенного SNW-механизмом и дающего общее благосостояние W_0. По теореме 1 GA+GVA за конечное число шагов построит финальное распределение X с общим благосостоянием W. В этом алгоритме переход от одного распределения к другому осуществляется, только если общее благосостояние в новом распределении больше общего благосостояния в текущем распределении. Поэтому либо $X = X_0$ (и тогда $W = W_0$), либо $W > W_0$.

Теорема 3. GA+GVA имеет трудоемкость $O(|J|^2|N|)$.

Доказательство. Для времени выполнения одного шага алгоритма справедлива оценка $O(|J||N|)$., это трудоемкость жадного эвристического алгоритма без этапа сортировки. Построим верхнюю оценку числа шагов, нужных для того, чтобы по крайней мере одно задание перешло в очередь F.

Как в доказательстве теоремы 1, предположим, что с некоторого шага m задания не попадают в F. Когда все задания перейдут из J в D и будут выполнены присвоения $J := D$, $D := \{\varnothing\}$, новая очередь J станет такой же, какой была изначально (на шаге m). Следовательно, если очередь J будет очищена без перехода заданий в F, то получим ту же общую очередь ($F+J+D$), и распределение X, порожденное этой очередью, будет таким же. Но тогда $W(X) > W(X)$, так как задание переходит в D только при условии $W_{-k} > W$. Противоречие.

Отсюда следует, что не более чем за $|J|$ шагов хотя бы одно задание перейдет в очередь F. То есть GA+GVA имеет трудоемкость $O(|J|^2|N|)$. Теорема доказана.

Очевидно, механизм GA+GVA является индивидуально рациональным, а благодаря использованию GVA для назначения платежей — неманипулируемым для пользователей. Покажем, что GA+GVA будет слабо бюджетно-сбалансированным.

Теорема 4. Механизм GA+GVA является слабо бюджетно-сбалансированным.

Доказательство. Достаточно показать, что платеж каждого задания покрывает отправную цену занятых этим заданием вычислительных ресурсов.

Платеж задания k, попавшего в распределение, равен $p_k = W_{-k} - W + v_k c_k d_k$. Если при распределении без задания k ресурсы, которые использовало k, остаются незадействованными (то есть у задания k нет конкурентов), то

$$W_{-k} = W - v_k c_k d_k + \sum_{t=r_k}^{e_k} \sum_{n \in N} V_n C_n x_{knt} .$$

Подставляя в выражение для платежа задания, получаем

$$p_k = \sum_{t=r_k}^{e_k} \sum_{n \in N} V_n C_n x_{knt} ,$$

то есть платеж задания k в точности равен сумме отправных цен арендованных им вычислительных ресурсов.

Если у задания k есть конкуренты, то его платеж будет не меньше полученного выше [6]. Таким образом, платежи пользователей всегда будут покрывать затраты провайдеров, механизм GA+GVA является слабо бюджетно-сбалансированным. Теорема доказана.

Можем заключить, что GA+GVA обладает всеми полезными свойствами SNW-механизма, работает с такой же скоростью, как и SNW-механизм, но в некоторых случаях обеспечивает более эффективные распределения.

Приведем пример, который демонстрирует преимущество механизма GA+VGA.

Пусть η_1=(1, 10, 2, 1, 1), η_2=(2, 6, 1, 1, 1), θ_1=(5, 6, 1, 1, 1), $\theta_2 = \theta_3 = \theta_4 = (4, 5, 1, 1, 1)$.

В систему поступают заявки двух вычислительных узлов, первый производит 10 единиц мощности, имеет 2 единицы памяти и отправную цену 1; второй производит 6 единиц мощности, имеет 1 единицу памяти и отправную цену 2. Также поступают заявки от четырех заданий, первое запрашивает 6 единиц мощности и 1 единицу памяти за е.п. и оценивает полезность единицы мощности-времени 5. Остальные три задания имеют идентичные характеристики: запрашивают 5 единиц мощности и 1 единицу памяти за е.п. и оценивают полезность единицы мощности-времени 4. Для простоты рассматривается ситуация на одном промежутке.

В начале очереди заданий имеют вид $F = \{\varnothing\}$, $J = \{1, 2, 3, 4\}$, $D = \{\varnothing\}$. При распределении по жадному эвристическому алгоритму задание 1 выполняется на узле 1, задание 2 выполняется на узле 2, задания 3 и 4 в распределение не попадают. $W_0 = (5 - 1) \cdot 6 + (4 - 2) \cdot 5 = 34$.

При ценообразовании в соответствии с GVA на первом шаге алгоритма GA+GVA первое в очереди задание 1 оказывается неплатежеспособным и перемещается в D:

$$W_0 < W_{-1} = (4 - 1) \cdot 5 + (4 - 1) \cdot 5 + (4 - 2) \cdot 5 = 40,\ F = \{\varnothing\},\ J = \{2, 3, 4\},\ D = \{1\}.$$

Согласно алгоритму, распределение изменяется на $X := X_{-1}$, $W := W_{-1}$. Задания 2 и 3 выполняются на узле 1, задание 4 — на узле 2. Задание 1 не попадает в распределение. Переходим ко второму шагу алгоритма. Первое в очереди J задание 2 оказывается неплатежеспособным и перемещается в D:

$$W < W_{-2} = (4 - 1) \cdot 5 + (4 - 1) \cdot 5 + (5 - 2) \cdot 6 = 48,\ F = \{\varnothing\},\ J = \{3, 4\},\ D = \{2, 1\}.$$

Полагаем $X := X_{-2}$, $W := W_{-2}$. В новом распределении задания 3 и 4 выполняются на узле 1, задание 1 выполняется на узле 2, а задание 2 не попадает в распределение.

На следующих двух шагах GA+GVA рассматриваются задания 3 и 4. Они платежеспособны и платят суммы, равные их бюджетам:

$$p_3 = p_4 = 48 - 48 + 20 = 20,\ F = \{3, 4\},\ J = \{\varnothing\},\ D = \{1, 2\}.$$

Вполне логично, что заданиям 3 и 4 приходится платить по максимуму, ведь отсутствие одного из них не уменьшает суммарный излишек, так как есть задание 2 с такими же характеристиками, не попавшее в распределение.

Очередь J закончилась, присваиваем $J := \{1, 2\}$, $D := \{\varnothing\}$.

Следующее в очереди задание 1 платежеспособно, его платеж равен: $p_1 = 40 - 48 + 30 = 22$. После этого шага $F = \{3, 4, 1\}$, $J = \{2\}$, $D = \{\varnothing\}$.

И последнее в очереди задание 2 не входит в текущее распределение, его платеж $p_2 = 0$. После этого шага $F = \{3, 4, 1, 2\}$, $J = \{\varnothing\}$, $D = \{\varnothing\}$.

Все задания перешли в очередь F, работа алгоритма завершена.

В результате получили новое распределение, в котором задания 3 и 4 выполняются на узле 1, а задание 1 — на узле 2. При этом

$$W = 48,\ p_1 = 22,\ p_2 = 0,\ p_3 = p_4 = 20.$$

При использовании SNW-механизма получается следующий результат:

$$W = 34,\ p_1 = 24,\ p_2 = 20,\ p_3 = p_4 = 0.$$

4. Заключение

В работе рассматривается проблема построения экономического механизма для распределения вычислительных ресурсов в многопроцессорной системе. За основу взят SNW-механизм из [1], в котором вычислительная эффективность достигается в ущерб экономической эффективности. Цель работы — модифицировать SNW-механизм так, чтобы модифицированный механизм в любой исходной ситуации обеспечивал суммарный излишек не меньший, а в некоторых ситуациях — больший, чем при SNW-механизме.

Построен механизм распределения, сочетающий жадный эвристический алгоритм из [1] с ценообразованием в соответствии с обобщенным механизмом Викри и динамически изменяющий распределение при выявлении неплатежеспособного задания. Доказана конечность алго-

ритма. Показано, что этот алгоритм обеспечивает распределение с не меньшим, а в некоторых случаях — большим суммарным излишком, чем при SNW-механизме. Построена оценка трудоемкости алгоритма, которая совпадает с оценкой трудоемкости алгоритма поиска критического значения [1] и равна $O(|J|^2|N|)$.

Литература

1. Stößer J., Neumann D., Weinhardt C., et al. Market-based pricing in grids: On strategic manipulation and computational cost. European Journal of Operational Research (EJOR) 203(2). 464–475. 2009.
2. Carr N. The end of corporate computing. MIT Sloan Management Review 46 (3), 67. 2005.
3. Николенко С.И., Теория экономических механизмов. 2009.
4. Nisan, N., Roughgarden, T., Tardos, E., Vazirani, V.V. Algorithmic Game Theory. Cambridge Univercity Press, New York, NY, USA. 2007.
5. de Vries, S., Vohra, R., 2003. Combinatorial auctions: A survey. INFORMS Journal on Computing 15 (3), 284.
6. Parkes, D. Iterative combinatorial auctions: Achieving budget-balance with Vickrey-based payment schemes in combinatorial exchanges. 2001.
7. Hans Kellerer, Ulrich Pferschy, David Pisinger. Knapsack Problems. Springer Verlag. 2005.
8. Chekuri, C., Khanna, S. A PTAS for the multiple knapsack problem. SIAM Journal on Computing 35 (3), 713-728. 2006.
9. Varian, R. Economic Mechanism Design for Computerized Agents. 1995.