Информационные процессы, Том 20, № 3, 2020, стр. 205–214

МАТЕМАТИЧЕСКИЕ МОДЕЛИ, ВЫЧИСЛИТЕЛЬНЫЕ МЕТОДЫ

Об одном новом способе диспетчеризации для ненаблюдаемых систем с параллельным обслуживанием и дисциплиной FIFO в серверах[1]

М.Г. Коновалов, Р.В. Разумчик

Федеральный исследовательский центр "Информатика и управление" Российской академии наук, Москва, Россия

Поступила в редколлегию 21.07.2020

Аннотация—В работе приводятся первые результаты экспериментального исследования новой стратегии диспетчеризации в ненаблюдаемых системах массового обслуживания с параллельным обслуживанием. Главная отличительная особенность рассматриваемой системы — это невозможность наблюдения за ее динамическими характеристиками. Рассмотрен набор численных примеров, охватывающих различные варианты входного потока заданий, различные распределения длины заданий, а также разное количество гетерогенных серверов. Во всех примерах использовалась дисциплина FIFO обслуживания очередей. Новая диспетчеризация сравнивалась с лучшими из известных авторам алгоритмов. В большинстве примеров новый алгоритм обнаруживает сопоставимые (часто лучшие) значения среднего и дисперсии времени пребывания и требует для своей настройки определения существенно меньшего количества параметров.

КЛЮЧЕВЫЕ СЛОВА: системы с параллельным обслуживанием; дисциплина FIFO; диспетчеризация; стратегии размещения заданий; управление при неполном наблюдении.

1. ВВЕДЕНИЕ

Предметом исследования в данной статье являются стратегии диспетчеризации[2] для систем обслуживания при неполном наблюдении. Рассматривается класс почти ненаблюдаемых систем с параллельной обработкой заданий [1,2]. Система этого класса состоит из нескольких параллельно и независимо друг от друга работающих серверов, которые выполняют задания, направляемые на них единственным диспетчером. Обмен заданиями между серверами невозможен, а применяемая дисциплина обслуживания очереди внутри каждого сервера может быть любой.

Диспетчер, осуществляя выбор сервера для очередного поступающего задания, не имеет никакой информации о постоянно меняющемся состоянии системы (например, о числе заданий на серверах, об остаточных временах обслуживания, о размерах заданий и др.). Единственное, что диспетчер может наблюдать (и запоминать) — это свои собственные действия (то есть номера выбранных серверов) и моменты их совершения.

В то же время диспетчеру известна определенная априорная информация. Предполагаются известными: производительности серверов, распределение времени между поступлениями заданий, распределение объема заданий.

[1] Работа выполнена при поддержке Российского фонда фундаментальных исследований (грант 18-07-00692).

[2] Также иногда называемые стратегиями распределения заданий, стратегиями маршрутизации или стратегиями распределения ресурсов.

В указанных условиях возникает проблема построения оптимальной стратегии диспетчеризации. Причем в этой статье оптимальность понимается с точки зрения минимального стационарного среднего времени пребывания задания в системе или, что то же самое, стационарного среднего времени отклика. Отметим два важных обстоятельства. Недоступность для выбора управлений текущих состояний системы очень сужает множество допустимых стратегий. Однако априорная информация оставляет возможность осуществлять имитацию траектории процесса.

Прежде, чем переходить к точной подстановке задачи, скажем несколько слов о возможных приложениях каких-либо теоретических результатов в рамках обозначенной проблемы. По-видимому, основная область приложений сосредоточена в рамках распределенных компьютерных систем передачи, обработки и хранения данных, в которых по тем или иным причинам динамическая информация о системе вообще отсутствует. Наиболее перспективными из них являются системы добровольных вычислений, которые, судя по публикациям в научной периодической печати, являются предметом активных научных исследований [3–5]. За свежим обзором стратегий диспетчеризации в системах добровольных вычислений можно обратиться к [5, раздел 2.3]; в обзоре каждая стратегия относится к одному из двух классов (naive и knowledge-based), в зависимости от того используется ли ею предыстория производительности серверов. Новая стратегия, которая описывается в этой статье, относится к классу naive и, как показано далее, по ряду критериев превосходит наилучшие из известных. Однако она не является оптимальной в строгом математическом смысле.

Перейдем к постановке задачи. Рассмотрим ненаблюдаемую систему с параллельным обслуживанием и предположим, что

- в серверах реализована дисциплина FIFO;
- у диспетчера отсутствует очередь для хранения поступающих заданий, позволяющая осуществлять отложенный выбор;
- узлы абсолютно надежны[3].

Необходимо найти стратегию, которая минимизировала бы предельное среднее время отклика.

Из всех известных из научной периодической печати диспетчеризаций в указанных условиях могут быть реализованы лишь три: случайный выбор[4] (далее — RND), детерминированный выбор[5] (далее — DET) и выбор по предыстории действий. Подробное описание первых двух стратегий дано в [2, раздел 2], а последней — в [1]. В [1] на численных примерах также показано, что в широком диапазоне исходных параметров системы наилучшей является диспетчеризация по предыстории действий, моментам поступления и основанных на них статистиках. Возможность разработать такую эффективную стратегию[6] связана со свойством дисциплины FIFO, позволяющем выписать рекуррентные соотношения для величин остаточной работы в каждом сервере в момент поступления очередного задания и вести по ним (приближенный)

[3] Отметим, что это предположение является существенным при применении полученных результатов в системах добровольных вычислений, для которых свойственна ненадежность вычислительных узлов. Демонстрируемая эффективность предлагаемой стратегии подразумевает, что она применяется к подмножеству узлов, обладающих, по крайней мере, высокой доступностью.

[4] В зарубежной литературе — PAP (Probabilistic Allocation Policy) или RND (Random), или BS (Bernoulli Splitting). Напомним, что стратегия RND описывается набором $(p_1, \dots, p_N)$ из N чисел — вероятностей p_n выбора для очередного задания сервера n.

[5] Это стратегии, параметризуемые бесконечными последовательностями $\{a_1, a_2, \dots, a_{i-1}, a_i, \dots\}$, в которых a_n означает, что n-е задание направляется на сервер с номером a_n. В этой статье для порождения a_n используется алгоритм SG из [6, С. 184].

[6] В [1] также показано, что и для других стационарных характеристик (как, например, среднее время ожидания) можно разработать эффективные стратегии, используя те же общие соображения. Например, на EPS — дисциплину равномерного (справедливого) разделения процессора (см., например, [7, С. 55] и [2]).

расчет. При смене дисциплины обслуживания в серверах на принципиально новую дисциплину, общие соображения, предложенные в [1], редко удается довести до реализуемых вычислительных алгоритмов: либо возникает то, что сейчас принято "называть проклятием размерности", либо необходимые величины вовсе не поддаются ни прямому, ни косвенному расчету (например, остаточные времена пребывания).

В этой статье предлагается новая стратегия диспетчеризации класса naive, которая, с одной стороны, свободна от указанных выше вычислительных недостатков, но, с другой стороны, уступает лучшей стратегии класса (стратегии по предыстории действий) и, пока, во всяком случае, не поддается теоретическому анализу. Но указанные недостатки целиком оправдываются простотой реализации и масштабирования новой стратегии.

Статья организована следующим образом. В следующем разделе приводится более формальное описание рассматриваемой системы. В разделе 3 описывается новая стратегия диспетчеризации и приводится конструктивный алгоритм ее построения. В начале раздела приводятся предпосылки создания новой стратегии, а в конце обсуждаются оптимизационные сложности, возникающие при применении новой стратегии. Для большинства обсуждаемых здесь вопросов пока не известно ни аналитических решений, ни даже подходящих подходов к анализу. В разделе 4 приводятся результаты численных экспериментов, подкрепляющие основной вывод статьи. На численных примерах показано, что в сбалансированных системах[7] новая стратегия обнаруживает сопоставимые с RND и DET (часто лучшие) значения стационарного среднего (и дисперсии) времени отклика. В заключительном разделе резюмируются результаты работы и ставятся задачи дальнейших исследований.

2. ОПИСАНИЕ СИСТЕМЫ

В ненаблюдаемую систему с параллельным обслуживанием, состоящую из N серверов, поступает рекуррентный входной поток однородных заданий. Интервалы между поступлениями заданий образуют последовательность независимых случайных величин с одинаковым распределением F. Задания имеют случайный объем, который определяется одним и тем же распределением G. Каждое поступившее задание должно быть немедленно направлено на один из серверов, причем диспетчеру, осуществляющему этот выбор (в автоматическом или ручном режиме), недоступна информация о текущем (прошлом или будущем) состоянии системы (например, размеры очередей, остаточная работа на каждом сервере и т.п.). В то же время известно распределение F, распределение G, скорости серверов и полная предыстория принятых решений (включая моменты времени, в которые эти решения принимались). Количество заданий, которые могут одновременно выполняться на сервере, не ограничено. Выполнение заданий в каждом сервере происходит в соответствии с дисциплиной FIFO, т.е. первым пришел — первым вышел. В каждом сервере имеется один процессор для обработки заданий. Серверы работают независимо, без обмена заданиями и являются абсолютно надежными.

Производительность (интенсивность обслуживания) сервера n обозначим через $v^{(n)}$. Система функционирует в непрерывном времени $t \geq 0$. Пусть $0 \leq t_1 < t_2 < \ldots$ — последовательность моментов поступления заданий в систем. Решение (действие), принимаемое в момент t_i относительно вновь поступившего задания, обозначим через y_i. Полагаем, что $y_i = n$, если i-е по счету задание направлено на сервер n.

[7] Под этим подразумевается, что скорости серверов не слишком сильно отличаются друг от друга, т.е. среди серверов нет настолько быстрых, что выгодно отправлять все задания именно на них.

3. НОВАЯ СТРАТЕГИЯ ДИСПЕТЧЕРИЗАЦИИ

3.1. Предпосылки

Предположим, что описанная в предыдущем разделе система полностью наблюдаема. Тогда общеупотребительная схема построения правила, по которому выбирается сервер для вновь поступившего задания, заключается в следующем. Задана некоторая функция, которая позволяет количественно оценить состояние каждого сервера на основе текущего размера очереди и остаточного объема работы для всех заданий в очереди. После того как такие оценки выполнены для всех серверов, задание направляется на сервер с минимальной оценкой. В случае, когда несколько серверов имеют одинаковую минимальную оценку, выбирается сервер с наибольшей производительностью. Если по-прежнему имеется неоднозначность, то она разрешается равновероятным выбором.

Рассмотрим следующие варианты выбора оценочной функции:

- текущее количество заданий в очереди (алгоритм JSQ);
- суммарный остаточный объем работы всех заданий в очереди, включая обрабатываемое задание (алгоритм LWL);
- суммарное остаточное время выполнения всех заданий в очереди, включая обрабатываемое задание и вновь поступившее и (условно) присоединенное к очереди (алгоритм Myopic).

В случае алгоритма JSQ оценка состояния сервера строится по “наличному составу” заданий; в двух других случаях учитывается и вновь поступившее задание. Однако отметим, что ни в одной из этих стратегий не учитывается важное обстоятельство: задания, которые поступят позже, могут изменить время выполнения имеющихся заданий.

Теперь вернемся к исходной постановке, т.е. предположим, что система не наблюдаема. Пусть наряду с основным процессом поступления и обслуживания заданий в заданной системе обслуживания, имеется еще $m \geq 1$ вспомогательных процессов, моделирующих точно такую же систему, т.е. с такими же серверами и пр. Функционирование вспомогательных процессов в точности копирует работу основного процесса в части моментов поступления и распределения заданий, но различается длинами заданий. Кроме этого, различие между основным и вспомогательными процессами заключается в характере наблюдений. Если для основного процесса, наблюдения по условию ограничены моментами поступления заданий и совершенными управлениями, то вспомогательные процессы полностью наблюдаемы.

Более подробно и точно алгоритм диспетчеризации при неполном наблюдении можно описать индуктивно. Начальные состояния всех процессов одинаковые (например, соответствуют пустой системе). Пусть все процессы (то есть основной и m вспомогательных) проработали до некоторого момента t_i поступления очередного задания в основном процессе. Этот момент принудительно считается моментом поступления задания и для каждого из вспомогательных процессов. Однако длина нового задания определяется для каждого из вспомогательных процессов индивидуально и независимо с помощью заданной (и известной по предположению) функции распределения длины заданий. Каждый из вспомогательных процессов, исходя из заданного в нем правила диспетчеризации, определяет номер сервера, на который следовало бы отправить его собственное новое задание. Номера серверов, выбранные для различных вспомогательных процессов, вообще говоря, разные, однако среди них есть, по крайней мере, один наиболее часто встречающийся номер. Сервер с этим номером и выбирается для задания, поступившего в момент t_i в основном процессе. Более того, серверы с указанным номером являются фактическими приемниками новых заданий во всех вспомогательных процессах, независимо от того, что было выбрано в результате работы индивидуального правила диспетчеризации.

Далее работа основного и вспомогательных процессов (то есть имитация выполнения заданий) протекает независимо друг от друга вплоть до следующего момента t_{i+1} поступления задания в основном процессе. Подчеркнем, что поступление заданий во вспомогательных процессах происходит только в моменты $t_i, t_{i+1}, \ldots$ поступления заданий в основном процессе. Отметим также, что основной процесс отображает "реальный" процесс (хотя и может быть заменен имитационной моделью), в то время как вспомогательные процессы являются чисто виртуальными компьютерными моделями.

3.2. Алгоритм

Пусть $m = 1$, то есть пусть имеется всего один вспомогательный процесс. Выберем в качестве правила диспетчеризации для вспомогательного процесса правило Myopic. Предположим, что все задания, поступающие на серверы вспомогательного процесса (идентичные с серверами основного процесса), имеют одинаковую длину $c > 0$. Пусть t_i и t_{i+1} — два последовательных момента поступления заданий в основном процессе. Обозначим через $z_{t_i}^{(n)}$ время, необходимое для выполнения всех заданий, имеющихся на сервере n вспомогательного процесса в момент t_i, с учетом задания, распределенного в этот момент каким-то образом на один из серверов того же вспомогательного процесса. Через $\tilde{z}_{t_{i+1}}^{(n)}$ обозначим время, необходимое для выполнения всех заданий, имеющихся на сервере n вспомогательного процесса в момент t_{i+1}, без учета задания, поступившего в этот момент. Очевидно,

$$\tilde{z}_{t_{i+1}}^{(n)} = \max\left(0, z_{t_i}^{(n)} - (t_{i+1} - t_i)\right).$$

Положим

$$y_{i+1} = \operatorname{argmin}_{1 \le n \le N} \left(\tilde{z}_{t_{i+1}}^{(n)} + \frac{c}{v^{(n)}}\right).$$

Неоднозначность при нахождении минимума разрешается равновероятным выбором. Число y_{i+1} служит номером сервера, на который направляется задание, поступившее в момент t_{i+1}. Причем, для основного процесса длина этого задания определяется согласно заданному распределению, а для вспомогательного процесса длина задания равняется c. Таким образом[8],

$$z_{t_{i+1}}^{(n)} = \begin{cases} \tilde{z}_{t_{i+1}}^{(y_{i+1})} + \frac{c}{v^{(y_{i+1})}}, & \text{если } n = y_{i+1}, \\ \tilde{z}_{t_{i+1}}^{(n)}, & \text{иначе.} \end{cases}$$

3.3. Проблема нахождения оптимальных значений параметра

Предложенная стратегия зависит от одного неизвестного параметра — параметра c. Из численных экспериментов получается, что в каждой конкретной задаче существует лишь единственное оптимальное значение c, т.е. такое значение при котором достигается глобальный минимум стационарного среднего времени пребывания задания в системе. Однако формулу для вычисления c получить не удается. Поэтому оптимизационную задачу приходится каждый раз решать, во-первых, приближенно (т.к. множество значений c несчетно), и, во-вторых, с помощью имитационного моделирования, сложность которого зависит от сложности расчета времени пребывания в системе поступающего задания.

[8] Может показаться, что наилучшее c является также и оптимальным для системы из параллельных $M/D/1$ систем (см., например, [8]). Но, как показывают численные эксперименты, это не так. Это также является дополнительным свидетельством того, что значение c во вспомогательном процессе зависит (неизвестным образом) от распределения G.

Когда в серверах используется дисциплина FIFO, ситуация облегчается следующим обстоятельством: время $\theta_i^{(y_i)}$ пребывания в системе i-го задания при любой из трех рассматриваемых стратегий рассчитывается, как известно, по формуле Линдли и равно

$$\theta_i^{(n)} = w_i^{(n)} + \frac{g_i}{v^{(n)}}\delta_{n,y_i}, \quad w_{i+1}^{(n)} = \max\left(0, \theta_i^{(n)} - (t_{i+1} - t_i)\right), \ i \geq 1,$$

где $w_i^{(1)}, \ldots, w_i^{(N)}$ — остаточная работа в серверах в момент t_i, g_i — длина задания, поступившего в момент t_i, δ — символ Кронекера. При этом, конечно, номера серверов $y_1, y_2, \ldots$, на которые направляются задания, выбираются по-разному: для стратегии RND — в соответствии с оптимальным набором $(p_1, \ldots, p_N)$, при стратегии DET — в соответствии с алгоритмом SG ([6, С. 184]), а при новой стратегии — в соответствии с алгоритмом в разделе 3.2. Если условиться обозначать время обслуживания i-го задания при стратегии DET и новой стратегии соответственно $\theta_i^{(y_i)}$ и $\tilde{\theta}_i^{(\tilde{y}_i)}$, то для выбора "направления движения" при поиске оптимального значения c служит знак суммы $n^{-1}\sum_{i=1}^{n}\left(\tilde{\theta}_i^{(\tilde{y}_i)} - \theta_i^{(y_i)}\right)$ при достаточно большом n.

Однако никаких подобных, легко вычислимых выражений нельзя предложить в случае использования в серверах более сложных дисциплин (например, SRPT); в этом случае имитационное моделирование, по-видимому, является единственным способом найти c (см. [9]).

В случае произвольного потока расчет оптимальных значений целевой функции как при рандомизированной стратегии, так и при детерминированной связан с серьезными трудностями. Для RND стратегии задача нахождения оптимального набора $(p_1, \ldots, p_N)$ из N чисел — вероятностей p_n выбора для очередного задания сервера n, в редких случаях может быть решена в явном виде. В общем же случае приходится[9] либо аппроксимировать входящий поток с помощью рекуррентного потока и использовать при решении оптимизационной задачи приближенные формулы (например[10], [10]) для стационарного среднего времени пребывания, либо применять специальные алгоритмы оптимизации на имитируемых траекториях (например, [11]).

В случае детерминированной стратегии вообще не известен метод нахождения оптимальной последовательности действий. Для специальных случаев (так называемых бильярдных последовательностей, см. [6]) есть простые алгоритмы порождения последовательностей, которые, однако, зависят от $N-1$ неизвестных параметров — вероятностей $(q_1, \ldots, q_N)$. Обычным (и достаточно хорошим) образом действия также является замена набора $(q_1, \ldots, q_N)$ оптимальным (для RND) набором $(p_1, \ldots, p_N)$. При этом, конечно, оптимальность детерминированной стратегии не гарантируется.

4. ЧИСЛЕННЫЕ ЭКСПЕРИМЕНТЫ

Проверка большого числа примеров показывает, что в сбалансированных системах новая стратегия всегда лучше любого случайного выбора и, в сравнении с DET — обычно лучшей по своей простоте стратегией для ненаблюдаемых систем, обнаруживает сопоставимые значения стационарного среднего (и дисперсии) времени отклика. Напомним, что под стратегией случайного выбора (в таблицах ниже — RND) понимается алгоритм, в котором серверы выбираются с фиксированными вероятностями оптимальным[11] набором $(p_1, \ldots, p_N)$. Предполагается, что эти вероятности выбираются оптимальным образом. Для порождения детерминированной стратегии используются два алгоритма: алгоритм SG из [10, С. 184], в котором набор неизвестных параметров заменяется набором оптимальных значений $(p_1, \ldots, p_N)$ для стратегии RND

[9] Другое простое решение, не гарантирующее, однако, оптимальность, — это $p_n = v^{(n)}/\sum_{i=1}^{N} v^{(i)}$.
[10] Здесь нужно отметить, что в случае многопроцессорной системы такой подход уже практически не реализуем.
[11] Для немарковских систем — близким к оптимальному.

(в таблицах ниже — DET-RND), и алгоритм SG, в котором набор неизвестных параметров заменяется набором $\left(\frac{v^{(1)}}{\sum_{i=1}^{N} v^{(i)}}, \ldots, \frac{v^{(N)}}{\sum_{i=1}^{N} v^{(i)}}\right)$ (в таблицах ниже — DET-LB). Параметр новой стратегии (в таблицах ниже — AA-FIFO) выбирается в соответствии с правилом, описанным в разделе 3.2.

Отметим, что новая стратегия допускает различные модификации. Например, если рассматривать $z_{t_{i+1}}^{(n)}$ как остаточную работу на сервере n (во вспомогательном процессе) в момент t_{i+1}, и вспомнить, что все задания имеют одинаковую длину c, то ничто не мешает заменить $z_{t_{i+1}}^{(n)}$ на число заданий в сервере n в момент t_{i+1}. Одна из таких модификаций в таблицах ниже обозначается AA-PS.

В таблицах 1 и 3 приводятся значения стационарного среднего (и дисперсии) времени пребывания в системах с различным числом серверов N в зависимости от загрузки системы и стратегии диспетчеризации. В скобках указывается также и дисперсия. Предполагается, что входящий поток F — пуассоновский со средним λ^{-1}, а распределение размера заданий G — экспоненциальное со средним 1.

Таблица 1. Стационарное среднее (и дисперсия) время пребывания задания в системе из двух серверов ($N = 2$) при различных стратегиях диспетчеризации и различной загрузке. Производительности серверов $v^{(1)} = 1/3$, $v^{(2)} = 2/3$. Входящий поток — пуассоновский со средним λ^{-1}. Размер заданий имеет экспоненциальное распределение со средним 1. Загрузка системы $\lambda / \sum_{i=1}^{2} v^{(i)} = \lambda$. Значения параметра стратегии AA-FIFO дано в Таблице 2.

Загрузка	0,1	0,3	0,5	0,7	0,9
RND	1,76 (3,1)	2,58 (6,9)	3,77 (15)	6,39 (43)	19,4 (400)
DET-RND	1,76 (3,1)	2,41 (6,1)	3,22 (11)	5,17 (28)	15,0 (230)
AA-FIFO	1,74 (3,1)	2,31 (5,6)	3,19 (11)	5,18 (28)	15,1 (250)

Таблица 2. Значение параметра c для стратегии AA-FIFO из Таблицы 1.

Загрузка	0,1	0,3	0,5	0,7	0,9
Значение c	1,7	1,75	1,5	1,251	1,1

Таблица 3. Стационарное среднее (и дисперсия) время пребывания задания в системе из 128 серверов ($N = 128$) при различных стратегиях диспетчеризации и различной загрузке. Производительность сервера n равна $v^{(n)} = \frac{n}{64 \times 129}$. Входящий поток — пуассоновский со средним λ^{-1}. Размер заданий имеет экспоненциальное распределение со средним 1. Загрузка системы $\lambda / \sum_{i=1}^{128} v^{(i)} = \lambda$. Значения параметра стратегии AA-FIFO дано в Таблице 4.

Загрузка	0,1	0,3	0,5	0,7	0,9
RND	92,0 (8500)	136,2 (19000)	206 (45000)	362 (150000)	1126 (1500000)
DET-RND	76,5 (6100)	96,2 (10000)	130 (19000)	207 (51000)	592 (430000)
DET-LB	128 (74000)	134 (75000)	162 (110000)	242 (240000)	665 (1700000)
AA-FIFO	73,7 (5500)	95,6 (9600)	133 (20000)	219 (68000)	652 (830000)
AA-PS	73,7 (5500)	94,8 (9400)	130 (19000)	214 (83000)	612 (880000)

Таблица 4. Значение параметра c для стратегии AA-FIFO из Таблицы 3.

Загрузка	0,1	0,3	0,5	0,7	0,9
Значение c	3,4	2,3	1,72	1,38	1,05

Во всем диапазоне нагрузки новые стратегии позволяют заметно улучшить значение целевого показателя по сравнению со стратегиями RND и DET-LB. Сложнее обстоит дело при

сравнении со стратегией DET-RND. Общая картина такова, что наблюдается уменьшение выигрыша с увеличением числа серверов и загрузки. Вместе с тем, новые стратегии обнаруживают сопоставимые (при загрузках ниже средней — лучшие) значения стационарного среднего и дисперсии времени отклика. Это свойство, вкупе с тем обстоятельством, что они требуют для своей настройки определения существенно меньшего количества параметров, позволяет считать новые стратегии равномерно лучшими для ненаблюдаемых систем с параллельным обслуживанием.

Аналогичным образом обстоит дело и с немарковскими системами. В таблицах 5 и 6 приводятся значения стационарного среднего времени пребывания, позволяющие оценить влияние характеристик входящего потока и распределения размера заданий на эффективность новой стратегии. Рассматривается случай двух серверов и загрузок 0,3 и 0,5. В качестве распределения G размера поступающих заданий рассматривались следующие варианты (среднее значение всюду равно 1):

- вырожденное распределение — длина всех заданий равна 1 (коэффициент вариации $CV = 0$);
- бимодальное в точках 0,5 и 5, взятыми с вероятностями $8/9$ и $1/9$ соответственно (коэффициент вариации — $CV = 1.41$).

В качестве распределения F входящего потока рассматривались:

- экспоненциальное распределение с параметром λ (т.е. $F(x) = 1 - e^{-\lambda x}$);
- гиперэкспоненциальное распределение с параметрами $(a_1; a_2; \lambda_1; \lambda_2)$ (т.е. $F(x) = a_1(1 - e^{-\lambda_1 x}) + a_2(1 - e^{-\lambda_2 x})$);
- эрланговское распределение с параметрами $(k; \lambda)$ (т.е. $F(x) = 1 - \sum_{n=0}^{k} \frac{(\lambda x)^n}{n!} e^{-\lambda x}$).

Таблица 5. Стационарное среднее (и дисперсия) время пребывания задания в системе из двух серверов ($N = 2$) при различных предположениях о входящем потоке и распределении размера заданий. Производительности серверов $v^{(1)} = 1/5$, $v^{(2)} = 4/5$. Загрузка системы фиксирована и равна 0,3.

		Распределение размера заданий	
		Вырожденное	Бимодальное
Распределение входящего потока	Экспоненциальное $(\lambda = 0,3)$ $CV = 0$	RND 1,63 (0,45) $p = 0,67$ DET-RND 1,63 (0,45) $q = 0,67$ AA-FIFO 1,62 (0,42) $c = 1,2$	RND 2,38 (8,8) $p = 1$ DET-RND 2,38 (8,8) $q = 1$ AA-FIFO 2,33 (9,1) $c = 1,72$
	Гиперэкспоненциальное $(1/6; 5/6; 0,1; 0,5)$ $CV = 1,61$	RND 1,92 (1) $p = 1$ DET-RND 1,92 (1) $q = 1$ AA-FIFO 1,87 (0,80) $c = 1,24$	RND 3,04 (18) $p = 0,87$ DET-RND 2,90 (15) $q = 0,087$ AA-FIFO 2,83 (14) $c = 1,72$
	Эрланговское $(50; 15)$ $CV = 0,14$	RND 1,25 (≈ 0) $p = 1$ DET-RND 1,25 (≈ 0) $q = 1$ AA-FIFO 1,25 (≈ 0) $c = 1$	RND 1,72 (4,6) $p = 1$ DET-RND 1,72 (4,6) $q = 1$ AA-FIFO 1,72 (4,6) $c = 1$

Для немарковских систем соотношения между рассматриваемыми стратегиями не меняется. При условии, что есть возможность находить близкие к оптимальным наборы $(p_1, \ldots, p_N)$, наилучшей из ранее известных является стратегия DET-RND. Новые стратегии обнаруживают сопоставимые (при загрузках ниже средней — лучшие) значения стационарного среднего и

Таблица 6. Стационарное среднее (и дисперсия) время пребывания задания в системе из двух серверов ($N = 2$) при различных предположениях о входящем потоке и распределении размера заданий. Производительности серверов $v^{(1)} = 1/3$, $v^{(2)} = 2/3$. Загрузка системы фиксирована и равна 0,5.

		Распределение размера заданий	
		Вырожденное	Бимодальное
Распределение входящего потока	Экспоненциальное ($\lambda = 0,5$) $CV = 1$	RND 2,82 (2,4) $p = 0,775$ DET-RND 2,41 (0,89) $q = 0,69$ AA-FIFO 2,28 (0,80) $c = 1,17$	RND 4,72 (34) $p = 0,725$ DET-RND 4,15 (25) $q = 0,725$ AA-FIFO 4,15 (26) $c = 1,56$
	Гиперэкспоненциальное ($0,98; 0,02; 0,98; 0,02$) $CV = 5$	RND 8,18 (48) $p = 0,67$ DET-RND 6,04 (21) $q = 0,67$ AA-FIFO 6,00 (20) $c = 1,67$	RND 13,3 (220) $p = 0,73$ DET-RND 11,9 (170) $q = 0,73$ AA-FIFO 11,9 (180) $c = 1,85$
	Эрланговское ($50; 25$) $CV = 0,14$	RND 1,50 (≈ 0) $p = 1$ DET-RND 1,50 (≈ 0) $q = 1$ AA-FIFO 1,50 (≈ 0) $c = 1$	RND 3,83 (23) $p = 0,75$ DET-RND 3,36 (17) $q = 0,75$ AA-FIFO 3,38 (17) $c = 1,78$

дисперсии времени отклика. Однако также наблюдается уменьшение выигрыша с увеличением дисперсии длины задания.

Наконец отметим, что, как показывают численные эксперименты и аналитический анализ, возможность уменьшить значение целевой функции за счет применения новой стратегии зависит, помимо прочего, от соотношений между скоростями серверов. Так, например, при низкой загрузке в случае двух серверов ($N = 2$) и пуассоновского входящего потока, новая стратегия лучше RND и DET-RND тогда и только тогда, когда скорость самого быстро сервера не превосходит $2(v^{(1)} + v^{(2)})/3$. К сожалению, в общем случае, при произвольном N и произвольных распределениях F и G, не удается выписать соотношения, при которых новая стратегия дает преимущество. В итоге остается возможным сделать лишь довольно расплывчатый и требующий дальнейших уточнений вывод о том, что система должна быть сбалансированной в том смысле, что скорости серверов не должны сильно отличаться друг от друга.

5. ЗАКЛЮЧЕНИЕ

Предложенный эвристический алгоритм диспетчеризации для почти ненаблюдаемых систем с параллельным обслуживанием успешно конкурирует с наилучшими известными стратегиями, а часто и превосходит их по критерию стационарного среднего времени пребывания задания в системе. В последнем случае он обеспечивает также и меньшее нестационарное среднее время. Привлекательными отличительными особенностями новой стратегии являются универсальность (применимость к немарковским системам), а также ее простота. Ожидаемым следствием этих преимуществ является то, что в некоторых частных случаях удается разработать специальные алгоритмы, которые превосходят тот, который предложен в статье (см., например, [1]).

Остается практически непонятной теоретическая основа продуктивности столь простой в реализации, но интуитивно совершенно не очевидной конструкции. Некоторые эффекты удалось обнаружить в процессе моделирования. Например, анализ численных экспериментов показал, что, по-видимому, преимущество новой стратегии получается за счет более активного использования медленных серверов. Так, для случая двух серверов (см. Таблицу 1) и загрузки

0,1 новая стратегия предписывает посылать на медленный сервер почти 10% заданий, тогда как детерминированная стратегия вообще не использует медленный сервер. Однако подобных фактов пока недостаточно, чтобы предложить сколь-нибудь удовлетворительное теоретическое обоснование нового алгоритма. Усилия в направлении решения такой задачи представляются интересным направлением дальнейших исследований.

СПИСОК ЛИТЕРАТУРЫ

1. Konovalov M. Razumchik R. “Improving routing decisions in parallel non-observable queues” // *Computing*, 2018, vol. 100, no. 10, pp. 1–21.
2. Коновалов М.Г., Разумчик Р.В. Минимизация среднего времени пребывания в ненаблюдаемых системах с параллельным обслуживанием и дисциплиной справедливого разделения процессора в серверах. Информационные процессы, 2019, Т. 19, Вып. 3, стр. 327–338.
3. Parkhomenko S. S., Ledeneva T. M. “Scheduling in volunteer computing networks, based on neural network prediction of the job execution time” // *International Journal of Parallel, Emergent and Distributed Systems*, 2019, vol. 34, no. 4, pp. 430–447.
4. Lavoie E., Hendren L. “Personal volunteer computing” // *Proceedings of the 16th ACM International Conference on Computing Frontiers*, 240–246.
5. Mengistu T.M., Che D. “Survey and Taxonomy of Volunteer Computing” // *ACM Comput. Surv.*, 2019, vol. 52, no. 3, Art. ID 59.
6. Hordijk A., van der Laan D.A. “Periodic routing to parallel queues and billiard sequences” // *Mathematical Methods of Operations Research*, 2004, vol. 59, no. 2, pp. 173–192.
7. Яшков С.Ф. Анализ очередей в ЭВМ. М.: Радио и связь, 1989.
8. Hyytiä E. “Optimal routing of fixed size jobs to two parallel servers” // *INFOR: Information Systems and Operational Research*, 2013, vol. 51. pp. 215–224.
9. Konovalov M., Razumchik R. “A Simple Dispatching Policy For Minimizing Mean Response Time In Non-Observable Queues With SRPT Policy Operating In Parallel” // *Communications of the ECMS*, 2020, vol. 34, no. 1, pp. 398–402.
10. Kraemer W., Langenbach-Belz M. “Approximate formulae for the delay in the queueing system $GI/G/1$ // *Proc. 8-th Int. Teletr. Congress*, 1976, pp. 2351-8.
11. Коновалов М.Г. Методы адаптивной обработки информации и их приложения. М.: ИПИ РАН, 2007.

New policy for routing jobs to parallel heterogeneous unobservable servers with FIFO scheduling

Konovalov M.G., Razumchik R.V.

Abstract: We present first results of the numerical analysis of the new policy for routing jobs in unobservable systems with parallel heterogeneous servers, each having an infinite-capacity queue with FIFO scheduling. Jobs, having the same job size distribution, arrive one by one to the dispatcher, which immediately routes it to one of the servers. The peculiar feature of the system is that the dispatcher does not have any online information about it. New single-parameter routing policy is being proposed, which minimizes the job's long-run mean response time. According to the numerical experiments, in all cases it performs (with respect to both the mean and the variance) at least as good as the best dispatching policies (known to the authors) available for such systems.

KEYWORDS: dispatching; server farm; FIFO scheduling; customer assignment; non-observable queues