2.3. ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ И ТЕЛЕКОММУНИКАЦИИ

Вычислительные системы и их элементы (технические науки) (2.3.2)

Информатика и информационные процессы (технические науки) (2.3.8)

УДК 004.75
DOI: 10.24160/1993-6982-2023-5-156-168

Предоставление ресурсов облачных платформ для выполнения композитных приложений в парадигме «поток работ как сервис»

В.В. Топорков, Д.М. Емельянов, А.Н. Булхак

Статья посвящена анализу существующих моделей, методов и алгоритмического обеспечения для выполнения наукоемких приложений на облачных платформах в рамках новой концепции «поток работ как сервис» (Workflow as a Service (WaaS)). Платформы WaaS, представляющие собой так называемые мультиарендные среды, позволяют реализовать эффективные механизмы для управления непрерывными потоками разнотипных работ в облачных вычислениях.
Рассмотрен ряд важнейших аспектов: наличие различных провайдеров IaaS, предоставляющих разнотипные ресурсы; территориальная распределенность центров обработки данных; разнородность потоков работ, поступающих на платформу WaaS; необходимость реализации принципа «платить за использование» для конкретного пользователя; решение проблемы размещения на физических серверах виртуальных машин и создания множества контейнеров в них, каждый из которых может быть использован задачами разных потоков. Выделены перспективные направления в развитии методов предоставления ресурсов облачных платформ на основе технологий разворачивания виртуальных машин и контейнеров для выполнения композитных приложений в рамках активно развивающейся парадигмы WaaS.

Ключевые слова: облачные вычисления, система управления потоками работ, планирование, предоставление ресурсов, виртуальная машина, контейнер.

Для цитирования: Топорков В.В., Емельянов Д.М., Булхак А.Н. Предоставление ресурсов облачных платформ для выполнения композитных приложений в парадигме «поток работ как сервис» // Вестник МЭИ. 2023. № 5. С. 156—168. DOI: 10.24160/1993-6982-2023-5-156-168.

Providing Cloud Resources for Running Composite Applications in the Workflow-as-a-Service Paradigm

V.V. Toporkov, D.M. Yemelyanov, A.N. Bulkhak

The article analyzes the existing models, methods, and algorithmic support for performing science-intensive applications on cloud platforms within the framework of the new concept called Workflow-as-a-Service (WaaS). Being so-called multitenant environments, the WaaS platforms offer the possibility to implement efficient mechanisms for managing continuous and heterogeneous workflows in cloud computing. A number of most important aspects are considered: the availability of various IaaS providers, which offer different types of resources; geographic distribution of data processing centers; heterogeneity of workflows; the need to implement the "pay-per-use" model for a specific user; and solution of the problem of deploying virtual machines on physical servers and setting up a multitude of containers in these machines, with each container ready for being used by tasks from different flows. The article also points out promising lines in the development of methods for providing the cloud platform resources based on technologies for deploying virtual machines and containers to run composite applications within the framework of the actively developing WaaS paradigm.

Key words: cloud computing, workflow management system, scheduling, provision of resources, virtual machine, container.

For citation: Toporkov V.V., Yemelyanov D.M., Bulkhak A.N. Providing Cloud Resources for Running Composite Applications in the Workflow-as-a-Service Paradigm. Bulletin of MPEI. 2023;5:156—168. (in Russian). DOI: 10.24160/1993-6982-2023-5-156-168.

Введение

В настоящее время поток работ — наиболее распространенная модель выполнения композитных (составных) приложений в различных областях знаний. В данной модели научными сообществами реализуются такие известные проекты, как Montage (астрономия), CyberShake (сейсмология), Epigenomics, Sipht (биоинформатика), LIGO (физика гравитационных волн). Поток работ, как правило, — совокупность взаимосвязанных задач, формализуется в виде ориентированного бесконтурного графа (Directed Acyclic Graph — DAG), вершины которого представляют задачи, а дуги — информационные и логические связи. Проблема планирования в рамках этой модели является *NP*-полной.

Облачные технологии активно используются для выполнения научных потоков работ, составляющих важнейший класс композитных приложений. В частности, «инфраструктура как сервис» (Infrastructure as a Service — IaaS) позволяет системе управления потоками работ (Workflow Management System — WMS) получить доступ к практически неограниченному пулу виртуализированных ресурсов по принципу «платить за использование» (Pay-per-use). Однако, возникает ряд фундаментальных проблем, связанных с планированием разнородных работ, от решения которых критично зависит эффективность использования ресурсов в облачных вычислениях.

К числу таких проблем относят разработку адекватных моделей, методов и соответствующего инструментария предоставления ресурсов облачных платформ на основе технологий разворачивания виртуальных машин (ВМ) и контейнеров для выполнения композитных приложений в рамках активно развивающейся парадигмы «поток работ как сервис» (Workflow as a Service — WaaS). Научная значимость исследований в этой области распределенных вычислений заключается в комплексном решении проблемы планирования и управления разнородными потоками работ на платформах WaaS.

В рамках моделей WaaS следует учитывать ряд важнейших аспектов. Во-первых, наличие различных провайдеров IaaS, предоставляющих разнотипные ресурсы. Во-вторых, территориальную распределенность центров обработки данных. В-третьих, разнородность потоков работ, поступающих на платформу WaaS. В-четвертых, необходимость реализации принципа «платить за использование» для конкретного пользователя. Наконец, своего решения требует задача размещения на физических серверах ВМ и создания множества контейнеров в них, каждый из которых может быть использован задачами разных потоков.

Известные к настоящему времени работы в области планирования потоков работ на облачных платформах не охватывают всей совокупности указанных аспектов, учитывая лишь отдельные из них. В частности, они не предполагают отбор провайдеров IaaS. Как правило, оптимизируется выполнение лишь одного потока работ. Известные модели планирования нескольких потоков зачастую являются весьма упрощенными, например, предполагают создание одного контейнера в каждой из ВМ, выполнение потоков работ в одном центре обработки данных, в качестве критерия оптимальности их планирования используется общая стоимость использования виртуальных машин соответствующего типа, что нарушает принцип «платить за использование».

Проведен анализ современного состояния методов и технологий выполнения наукоемких композитных приложений в рамках парадигмы WaaS. Обозначены перспективы развития моделей, методов и средств организации облачных вычислений на базе комплексного сочетания алгоритмов приоритетного планирования как отдельных задач в потоках работ, так и независимых и разнородных потоков композитных приложений.

Исходные положения в моделях планирования потоков работ на платформе WaaS

Под ресурсами понимается множество ВМ соответствующих типов и созданных в них контейнеров.

Поток работ — совокупность взаимосвязанных задач, формализуемая в виде DAG, вершины которого представляют задачи, а дуги — информационные и логические связи.

Отношение частичного порядка на совокупности $T = P \cup D$ задач задается с помощью DAG, подмножество P вершин которого соответствует задачам обработки и доступа к памяти, а подмножество D — процедурам обмена данными между задачами. На рисунке 1 приведен пример информационного графа: вершины p_1 ,..., p_6 представляют задачи обработки, а d_1, ..., d_8 — передачу данных. Граф параметризуется априорными оценками длительности t_{ij}^0 выполнения задачи $T_i \in T$, $i = 1, ..., n$ на ресурсе типа j относительных объемов v_{ij} вычислений на ресурсе j-го типа или передаваемых данных в коммуникационной подсистеме типа j и т. п.

Задание состоит из задач, которые могут быть запущены параллельно, если завершено выполнение их предшественников.

Так, задача p_1 является предшественником задач p_2 и p_3, задачи p_2 и p_3 — предшественники задач p_4 и p_5, задачи p_4 и p_5 — предшественники задачи p_6.

В результате планирования определяется время t_{ij}, отведенное для выполнения каждой из задач потока, которое не меньше априорной оценки длительности t_{ij}^0 выполнения задачи $T_i \in T$, $i = 1, ..., n$ на ресурсе типа j. Выбор ресурса корректируется в ходе выполнения пакета заданий с учетом динамики загрузки ресурсов.

В качестве примера критерия оптимальности плана используем функцию стоимости завершения обработки следующего вида:

$$CF = \sum_{i=1}^{n} \left\lceil v_{ij} / t_{ij} \right\rceil, \; t_{ij} \geq t_{ij}^0,$$

где $[v_{ij}/t_{ij}]$ — частная функция стоимости выполнения задачи; $\lceil \cdot \rceil$ — ближайшее не меньшее целое число.

На практике графовые структуры композитных приложений, представляемые DAG, весьма сложны. На рисунке 2 изображены результаты визуализации ряда средствами библиотеки Pyvis на языке Python. Примеры построены авторами статьи на синтетических наборах файлов DAX (Directed Acyclic Graph in XML Format), сформированных генератором потоков работ «Pegasus Workflow Generator» [1]. Pegasus — система управления рабочими процессами с открытым исходным кодом.

На платформу WaaS в любой момент времени поступает множество потоков работ. При этом каждая из задач потока может выполняться на некотором подмножестве типов ВМ, предоставляемых провайдером IaaS.

Планирование и управление потоками работ на облачных платформах в рамках концепции WaaS

Насчитывается огромное число систем управления потоками работ [1, 2]. Это — ASKALON, Galaxy, HyperFlow, Kepler, Pegasus, Taverna, CloudBus и ряд других.

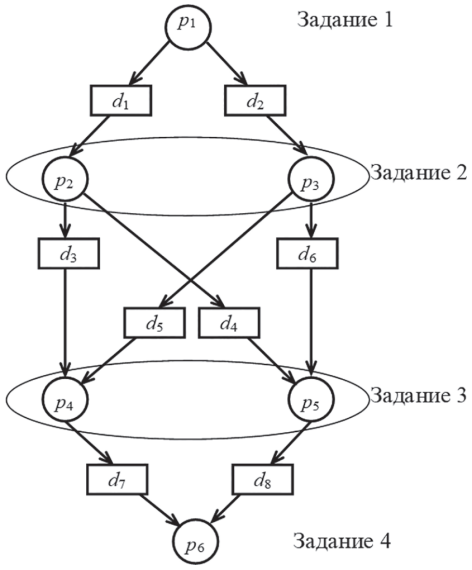
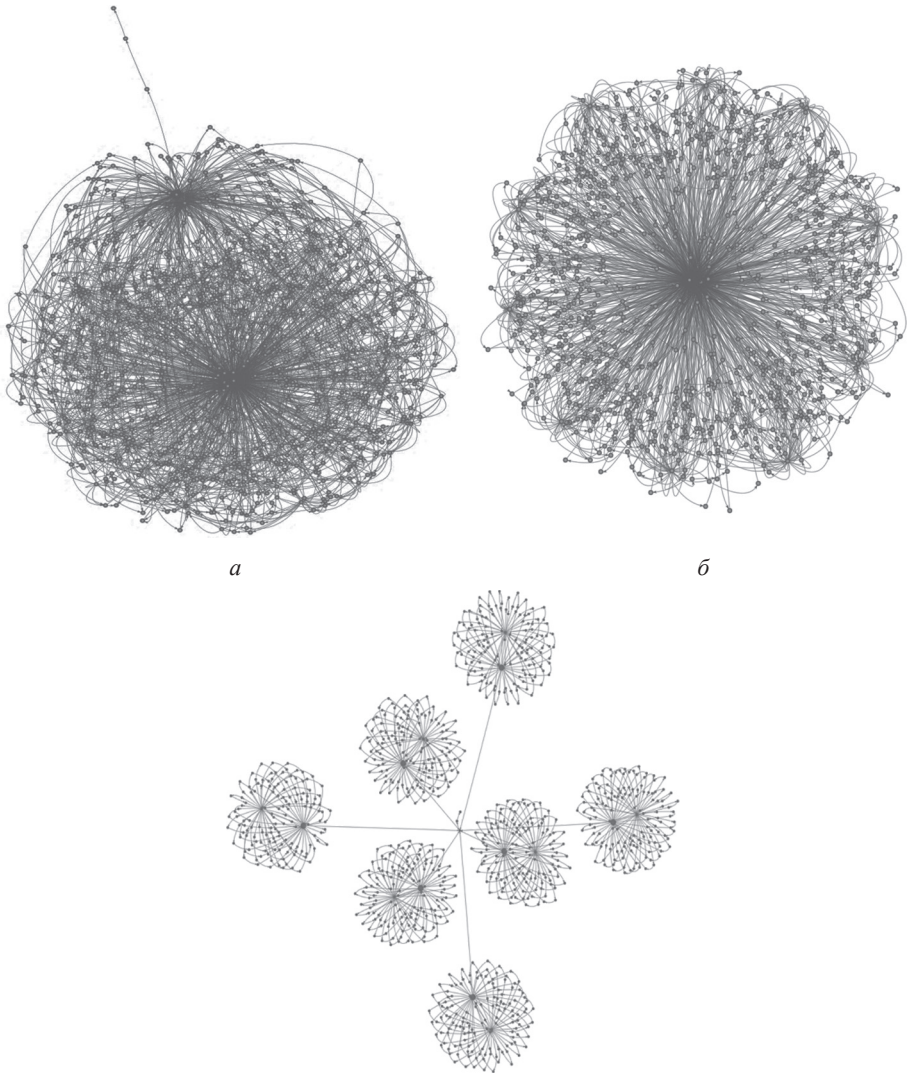


Рис. 1. Структура потока работ

Новая парадигма WaaS позволяет реализовать эффективные механизмы для управления непрерывными потоками разнотипных работ в облачных вычислениях [3, 4]. Платформы WaaS представляют собой мультиарендные среды, интегрирующие вычислительные и сетевые ресурсы, а также хранилища данных, предоставляемые провайдерами инфраструктуры как сервис. Провайдеры IaaS предлагают упомянутые ресурсы как виртуальные машины, реализуя соответствующую ценовую политику. Важнейший аспект, от которого критично зависит эффективность использования ресурсов в облачных вычислениях, — планирование разнородных работ. Большинство существующих подходов, как правило, реализует планирование в пределах одного потока работ с соблюдением соответствующих требований к качеству обслуживания (Quality of Service — QoS). Парадигма WaaS позволяет решить проблему планирования для совокупности независимых работ.

Бóльшая часть известных алгоритмов планирования в качестве критерия оптимизации берет общую стоимость при заданном ограничении на время выполнения потока работ. Здесь можно упомянуть такие алгоритмы, как IC-PCP, IC-PCPD2 [5], EIPR [6], TB [7] и CCA [8]. Они позволяют занимать свободные временны́е слоты предоставленных ВМ, чтобы максимизировать загрузку ресурсов и минимизировать стоимость их использования. Однако наличие ограничений на время выполнения задач и зависимости между задачами потока приводят к тому, что нельзя полностью исключить незанятые слоты.

Некоторые композитные научные приложения состоят из взаимосвязанных работ — так называемых ансамблей [9 — 11]. Известно несколько алгоритмов планирования приложений такого типа [12 — 15]. Эти



а *б* *в*

Рис. 2. Графы потоков работ с 1000-ю вершинами в проектах Montage (*а*), CyberShake (*б*), Epigenomics (*в*)

алгоритмы учитывают требования QoS не для каждого потока, а для ансамбля в целом. Число потоков в ансамбле известно заранее. Наконец, потоки работ в ансамбле имеют один и тот же тип, то есть обладают одинаковой структурой и различаются только по объемам вычислений и входным данным.

В [16] предложен алгоритм планирования для множества потоков работ, поступающих от разных пользователей в произвольные моменты времени. Однако он предназначен для кластерных приложений и, следовательно, не учитывает стоимость использования инфраструктуры, а уровень доступных ресурсов полагается фиксированным. В нем минимизируется время выполнения каждого из потоков работ и не принимаются во внимание временны́е ограничения для решения отдельных задач. Тем не менее, важные достоинства предложенного подхода — динамичное планирование потоков и возможность контролировать уровень использования ресурсов.

Ряд работ посвящен планированию множественных потоков работ в облачных вычислениях. Здесь можно упомянуть [17], однако в указанной работе полагается, что число и типы потоков работ известны заранее, причем все потоки поступают одновременно. Кроме этого, не принимаются во внимание стоимость выполнения и временные ограничения. Основной критерий — уровень использования ресурсов.

Алгоритм, описанный в [18], основан на эвристике списочного планирования. Качество данных, обрабатываемых каждым потоком, рассматривается как часть требований QoS.

Подход, анализируемый в [19], базируется на стратегиях планирования с различными критериями при наличии ограничения на бюджет выполнения потока работ как фактора QoS.

В [20] дан алгоритм планирования для конечного множества разнородных ВМ, число которых остается неизменным в течение всего жизненного цикла системы. Следовательно, не затрагивается проблема гибкого предоставления пула ресурсов. Впрочем, это же относится и к алгоритмам из [17 — 19].

Алгоритм планирования Dyna [21] разработан для выбора провайдеров облачных сервисов, допускает масштабирование и динамичное назначение ВМ в зависимости от текущего состояния выполнения задач. Он позволяет отобрать типы ВМ для каждой задачи потока с целью минимизации стоимости выполнения. При этом вводится вероятностная оценка времени завершения, а стоимость использования ВМ рассматривается для двух моделей — статической и динамической, подобно тому, как это поддерживается в сервисах Amazon. При этом полагается, что ВМ разделяются потоками работ одного и того же типа, но с различным числом задач.

Алгоритмы SCS [22] и WPPDS [23] предназначены для планирования потоков работ в облачных вычислениях и имеют механизмы автоматического масштабирования. SCS реализует начальный план предоставления ресурсов на основе эвристической процедуры глобальной оптимизации, а затем уточняет его в ходе выполнения потока, чтобы учесть задержки предоставления ресурсов. Уточнение плана осуществляется алгоритмом глобальной оптимизации для оставшихся задач потока, причем эта оптимизация применяется для каждой из задач. Это обусловливает высокую вычислительную сложность алгоритма и ограничивает его масштабируемость в зависимости от числа задач в потоке. WPPDS учитывает бюджет на выполнение всей совокупности потоков и временные ограничения для каждого из потоков. Критерий планирования — завершить как можно большее число высокоприоритетных потоков при заданном бюджете.

Алгоритмы планирования EPSM [3] и EBPSM [4] поддерживают множественные потоки работ. EPSM минимизирует общую стоимость использования ресурсов при заданных ограничениях для выполнения каждого из потоков работ. При этом создается лишь один контейнер в каждой из виртуальных машин, потоки работ выполняются лишь в одном центре обработки данных, а в качестве критерия оптимальности их планирования выступает общая стоимость использования виртуальных машин соответствующего типа, что нарушает принцип «платить за использование». EBPSM минимизирует общее время выполнения потоков при заданном бюджете. Оба алгоритма не предполагают отбор провайдеров IaaS.

В [24] предложены архитектура платформы WaaS и четыре эвристических алгоритма планирования: статический, динамический, адаптивный и жадный. Алгоритмы поддерживают разделение пула ВМ задачами одного и того же потока, а не задачами различных потоков работ. Они предназначены для минимизации времени и стоимости выполнения работ. При этом не учитываются задержки предоставления ресурсов и затраты времени на передачу данных. Достоинство алгоритмов заключается в том, что явно поддерживается политика разделения пула ВМ. Все представленные алгоритмы предполагают, что любая задача может быть назначена на любую из доступных ВМ, причем модель приложения явно не определяется.

Skyport [25] и Asterism DIaaS [26] представляют фреймворки WaaS, основанные на использовании технологии контейнеризации. Решаются проблемы упаковки задач потока работ и удобства разворачивания контейнеров на уже предоставленных ВМ так, чтобы выполнить любые задачи из любого потока. Вопросы предоставления ресурсов и планирования в вышеупомянутых работах не анализируются.

В [27] даны прототипы промежуточных фреймворков WaaS, учитывающие особенности структуры потоков работ, поддерживающие интеграцию WMS, стоимостную модель и назначение на ресурсы. Основное внимание уделено вопросам выполнения потоков с непрерывной и наращиваемой обработкой данных. Иными словами, завершение какой-либо задачи-предшественника не означает немедленного запуска задач-последователей. Эти задачи стартуют лишь тогда, когда задача-предшественник генерирует выходные данные, сильно влияющие на выполнение завершающей задачи потока. Предложенный алгоритм позволяет планировать лишь один поток работ.

Разворачивание виртуальных машин на физических серверах

Следующее направление исследований — решение проблемы разворачивания ВМ. Сильная сторона облачных вычислений — их масштабируемость [28 — 30]. В облачных средах ВМ создаются и отображаются на хосты (физические серверы) по мере формирования запросов на ресурсы. Проблема поиска подходящего хоста известна как задача размещения ВМ [31, 32]. Для ее решения предложен ряд методов [33]. В [34] проблема размещения ВМ решена методами эволюционных вычислений для минимизации энергопотребления хост-узлами (серверами). Авторы применяют алгоритм муравьиных колоний (Ant Colony System — ACS) в сочетании с изменением порядка миграции. Численные результаты демонстрируют преимущество предложенного подхода по сравнению с известными стратегиями, позволяют повысить эффективность использования ресурсов и экономить энергопотребление. В [35] проблема оптимизации размещения ВМ решается с учетом пропускной способности коммуникаций и формулируется как задача упаковки в контейнеры. Используется метаэвристический алгоритм оптимизации китов (Whale Optimization Algorithm, или WOA). В качестве инструментария применена среда Cloudsim. Аналогичное исследование проведено в [36]. Проблемы размещения ВМ и данных в этой работе рассматриваются совместно. Цель — снизить интенсивность сетевого трафика и повысить эффективность использования коммуникаций. Оптимизация выполняется на основе ACS. Отбор физических серверов происходит

по признаку их территориальной близости. Проблеме размещения ВМ посвящены также работы [37 — 39].

Впервые постановка задачи размещения ВМ в форме многокритериальной оптимизации с ограничениями и двумя целевыми функциями приведена в [40]. Основная идея — поиск оптимального размещения ВМ по физическим машинам (серверам) (множества Парето-оптимальных недоминируемых решений — фронта Парето). Идет поиск решения, минимизирующего простой ресурсов и энергопотребления. Авторами [40] использован метод оптимизации на основе ACS. В [41] подход, предложенный в [40], развивается с целью построения более реалистичной модели. Задача многокритериальной оптимизации сводится к двухуровневой оптимизационной задаче (Bilevel Optimization Problem). Различие между использованием памяти и процессоров каждой ВМ представляется отдельными функциями, входящими в критерий простоя ресурсов.

Открытые вопросы

Проблемы управления и планирования потоков работ продолжают привлекать к себе внимание научного сообщества. Здесь остается много открытых вопросов [42]. Это и обусловливает огромное разнообразие систем управления потоками работ [2]. Один из таких вызовов — собственно модель потока работ, формализуемая, как правило, в виде ориентированного бесконтурного графа (DAG). В ряде приложений циклы естественным образом присутствуют в потоках работ, поэтому в некоторых системах управления потоками работ прибегают к специальным (паллиативным) приемам, чтобы эту проблему обойти.

Первая такая система — Pegasus [43]. Чтобы устранить циклы, в Pegasus необходимо генерировать множество экземпляров подпотоков. Подобное решение принято и в ряде других проектов: Apache Airflow [44], Taverna [45], Kepler [46]. Весьма перспективным представляется подход, описанный в [47], допускающий циклы в графах, формализующих представление потоков работ в ряде научных приложений. Графы динамично трансформируются определенным образом уже в процессе выполнения приложения, когда порождаются линейные последовательности задач — так называемые процéссные цепочки. При этом затраты на трансформации значительно ниже, чем в известных решениях [43 — 46].

Таким образом, анализ современного состояния исследований в области планирования потоков работ на облачных платформах показывает, что известные к настоящему времени результаты не охватывают ряда важных аспектов:

- наличия различных провайдеров IaaS, предоставляющих разнотипные ресурсы;
- территориальную распределенность центров обработки данных;
- разнородность потоков работ, поступающих на платформу WaaS;
- необходимость реализации принципа «платить за использование» для конкретного пользователя;
- решения проблем размещения на физических серверах виртуальных машин и создания множества контейнеров в них, каждый из которых может быть использован задачами разных потоков.

Планирование потоков работ на основе циклической схемы

Одна из возможных и перспективных стратегий планирования заданий в потоках работ может быть реализована по циклической схеме [48 — 50] на альтернативных наборах предварительно отобранных слотов — временны́х отрезков доступности соответствующих ресурсов (рис. 3).

Описание слота, помимо технических характеристик соответствующих ВМ и контейнеров, дополняется таким атрибутом, как цена ресурса. Затем используются понятия набора — последовательности слотов, число которых не меньше числа слотов, необходимых заданию, и комбинации — последовательности наборов слотов для заданий пакета в текущем цикле планирования.

Таким образом, доступные к началу очередного цикла планирования ресурсы (контейнеры ВМ) формально представляются неупорядоченными слотами с соответствующими атрибутами: техническими характеристиками ВМ (тип процессора, пропускная способность сети), длительностью, стоимостью использования. Слоты имеют произвольные и несовпадающие моменты времени начала и окончания (рис. 4).

В каждом цикле выполняется планирование пакетов, готовых к выполнению заданий. Выполнение параллельного задания требует такого выделения определенного числа слотов, чтобы составные части задания (задачи) могли стартовать одновременно. План выполнения задания представляет собой набор отобранных слотов, а план выполнения пакета — комбинацию слотов.

Множество доступных слотов известно к началу каждого цикла планирования на основе прогноза занятости и освобождения подходящих для заданий контейнеров ВМ.

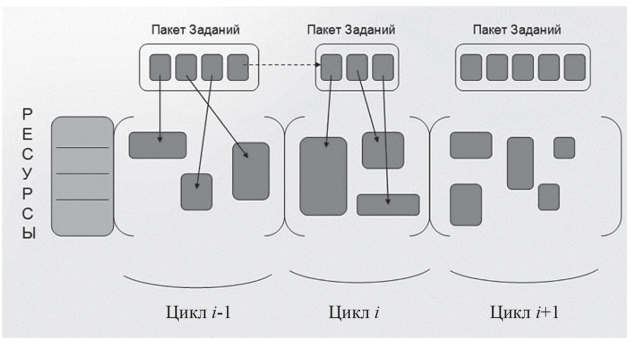


Рис. 3. Циклическая схема планирования пакетов заданий

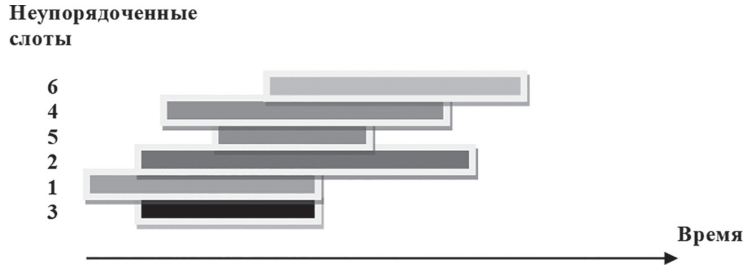


Рис. 4. Неупорядоченные по времени доступные слоты

Сначала делается попытка найти комбинацию подходящих слотов для всех заданий пакета. Если для какого-либо из заданий такого набора не существует, его планирование переносится в следующий цикл, и задание занимает место в очереди (пакете) в соответствии с заданным приоритетом, например, в начале очереди (см. рис. 3). При успешном отборе слотов для k-го задания список просматриваемых слотов для $(k + 1)$-го задания модифицируется. Из соответствующих временных отрезков исключаются периоды, занятые выполнением k-го задания. Отбор слотов для $(k+1)$-го задания осуществляется на модифицированном списке по рассмотренной ранее схеме. После просмотра слотов для всех заданий пакета отыскиваются альтернативные наборы, поскольку при отборе слотов для каких-либо заданий не весь исходный список может быть просмотрен. Эта процедура итеративно повторяется для всех заданий, планирование которых не перенесено в следующий цикл. В очередном цикле совокупность доступных слотов (см. рис. 3) обновляется в соответствии с динамикой загрузки ресурсов.

В каждом цикле выполняется назначение готовых к выполнению задач. При этом делается попытка использовать уже развернутые ВМ и контейнеры соответствующего типа. Если это удается, то из времени выполнения вычитается время на развертывание ВМ или контейнера в ВМ.

Время выполнения складывается из следующих составляющих:

- времени собственно обработки — как отношения объема вычислений (в миллионах инструкций) к производительности процессора (миллион инструкций в секунду) на процессоре ВМ соответствующего типа;
- времени на обмен данными между задачами потока (чтение и запись в глобальное хранилище, например, Amazon S3);
- времени на развертывание ВМ и контейнера в ВМ соответствующего типа.

Стоимость выполнения задачи потока на ВМ определенного типа определяется как отношение времени выполнения к периоду биллинга развернутых ВМ, умноженное на стоимость одного периода биллинга.

Стоимость выполнения потока работ — сумма стоимостей выполнения каждой из задач.

Если выполнение задачи может быть закончено до наступления очередного периода биллинга, то стоимость ее выполнения на ранее развернутых ВМ и созданных контейнерах полагается равной нулю.

Если выполнение задачи не заканчивается в текущем периоде биллинга, то оплачивается время, требуемое для ее завершения в последующих периодах.

Если в текущем цикле планирования какие-либо ВМ не востребованы, то они сворачиваются.

В каждом цикле планирования на основе локальных расписаний для каждого задания пакета должно быть отобрано требуемое число l подходящих по ресурсу, стоимости и времени слотов на соответствующих ВМ и контейнерах. Множество доступных слотов известно к началу каждого цикла планирования. Оно представляется неупорядоченными по времени доступными слотами (см. рис. 4). Выполнение параллельного задания может начаться при согласованном выделении l слотов таким образом, чтобы параллельные процессы, соответствующие задачам задания, стартовали одновременно. Для однородных ресурсов с процессорами одинаковой производительности и задачами примерно одинаковой сложности из совокупности доступных слотов требуется выделить прямоугольное «окно» $l \times t$, где t — время использования слотов (рис. 5, *а*). В случае неоднородных узлов и существенно различных по сложности задач это непрямоугольное «окно» с неровным правым краем (рис. 5, *б*).

Рассмотрим общую схему алгоритма отбора подходящих слотов с ранним временем старта «окна». Обозначим через t_s^b, t_s^e время начала и окончания слота s, используемого задачей задания в течение времени tp_s, где $p_s \le 1$.

1. Доступные слоты (см. рис. 4) упорядочиваются по неубыванию времени старта t_s^b (см. рис. 5).

2. Из полученного на шаге 1 списка выбирается очередной подходящий по ресурсу, цене и длительности слот s.

3. Из списка ранее просмотренных подходящих слотов от 1 до $s - 1$ удаляются слоты с временем окончания $t_{s'}^e < t_s^b + tp_s$, где $1 \le s' \le s^{-1}$.

4. Шаги 2, 3 повторяются до тех пор, пока не наберется заданное число l слотов.

Заметим, что в этой схеме на шаге 2 фигурируют не только время, но и требования к ресурсу и цене.

В примерах, иллюстрируемых рис. 5, $l = 3$. Время запуска задания определяется временем начала послед-

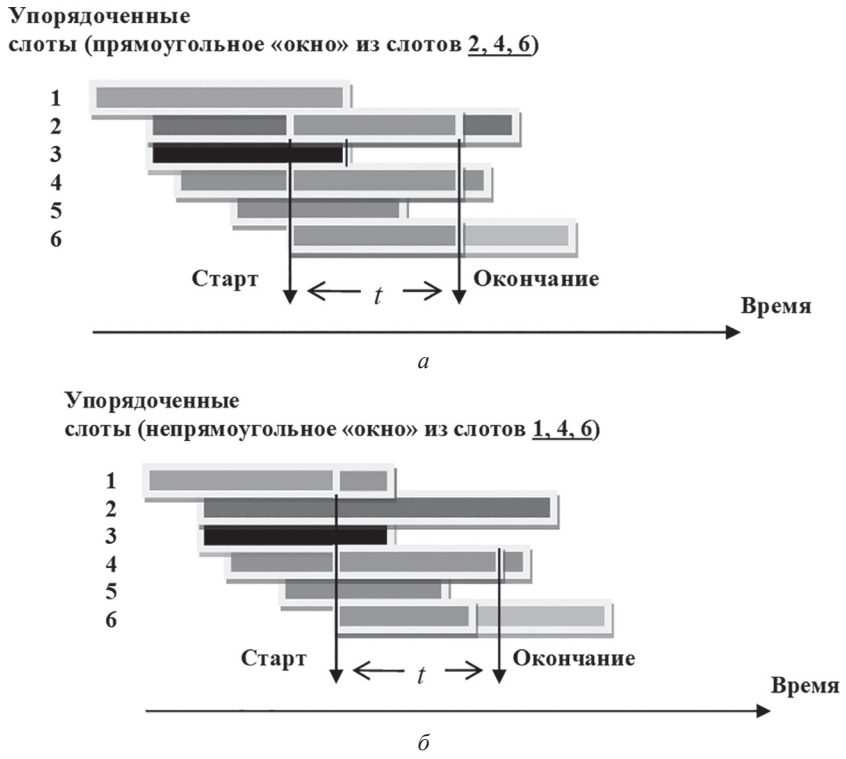


Рис. 5. Отбор слотов для выполнения задания

него из отобранных подходящих слотов (на рис. 5 — это слот 6). Время окончания выполнения задания зависит от *t*, — это максимум длительности выполнения задачи соответствующего задания по всем отобранным слотам (на рис. 5, *б* ему соответствует слот 4).

Данная схема применяется для каждого из заданий пакета. После просмотра слотов для всех *n* заданий пакета ищутся альтернативные наборы, поскольку при отборе слотов для каких-либо заданий не весь исходный список может быть просмотрен. Указанная процедура итеративно повторяется для всех заданий, планирование которых не перенесено в следующий цикл. В очередном цикле совокупность доступных слотов обновляется в соответствии с динамикой загрузки ВМ и контейнеров.

Стратегии назначения виртуальных ресурсов при выполнении потоков работ

Одна из важнейших проблем планирования и выполнения множества вычислительных задач потоков работ — эффективная аллокация и управление ресурсами (виртуальными машинами). Под управлением ресурсами понимается определение моментов старта (создания) и завершения (остановки, высвобождения) отдельных виртуальных машин и контейнеров, а также назначение и очередность выполнения готовых для запуска вычислительных задач. В [51] предложен ряд подходов к формированию пула виртуальных машин, предоставляемых провайдерами IaaS, для онлайн-планирования потока работ на основе методов машинного обучения.

Выделим несколько стратегий управления ВМ для выполнения потоков работ.

Во-первых, можно создавать новую специализированную ВМ для выполнения каждой отдельной задачи и останавливать ее по завершению выполнения. Данная стратегия позволяет экономно и гибко использовать вычислительное время ВМ, особенно в условиях, когда время старта и остановки намного меньше времени выполнения отдельных задач. При этом, время старта, как правило, зависит от сложности настройки необходимого программного окружения и времени на копирование и доставку входных данных. Время остановки ВМ может увеличиваться из-за необходимости сохранения и копирования результатов вычислений в глобальное хранилище данных.

Во-вторых, возможно поддерживать некоторый динамически изменяющийся пул постоянно активных виртуальных машин и распределять готовые к запуску задачи между ними (стратегия контроля). Динамичность пула подразумевает уменьшение и увеличение количества активных машин в зависимости от вычислительных потребностей в рассматриваемый момент времени. В данном подходе имеется возможность более эффективного планирования выполнения задач таким образом, чтобы минимизировать время старта и остановки виртуальных машин, загрузки и сохранения данных. Например, если выполнять две последовательные (зависящие по данным) задачи одного потока работ на одной виртуальной машине, то операция копирования и передачи данных не требуется.

С другой стороны, из-за особенностей и разнообразия структур потоков работ, а также их переменного количества не всегда можно обеспечить полную и постоянную загрузку всего пула активных ВМ. Таким образом, часть виртуальных машин будет время от времени простаивать, тем самым понижая полезность и экономическую эффективность данного подхода. Кроме того, определенную сложность составляет разработка алгоритма для эффективного назначения задач на доступные виртуальные машины.

В качестве третьего подхода рассмотрим класс смешанных стратегий, когда при выполнении потоков работ поддерживается некоторый базовый минимальный пул активных виртуальных машин, но дополнительные виртуальные машины создаются для выполнения отдельных заданий, например, не требовательных ко времени загрузки и сохранения данных.

Если реализация первого подхода выглядит тривиальной, то строгая и смешанная стратегии требуют наличия алгоритма для управления пулом активных виртуальных машин и назначения на них поступающих задач.

Заключение

Проведенный анализ показывает, что известные к настоящему времени решения в области планирования композитных приложений на облачных платформах не охватывают целого ряда аспектов, связанных с планированием разнотипных и непрерывных потоков работ.

Предстоящие исследования будут направлены на создание комплекса моделей, методов и инструментальных средств планирования непрерывных и разнородных потоков работ в композитных приложениях на основе особенностей их структуры, ресурсных потребностей, прогноза состояния облачной среды на платформе WaaS.

Значимость ожидаемых результатов состоит в том, что они позволят реализовать эффективное выполнение наукоемких междисциплинарных приложений на облачных платформах в рамках новой парадигмы «поток работ как сервис».

Литература

1. **Workflows** Community Summit [Электрон. ресурс] https://workflowsri.org/summits/community (дата обращения 24.02.2023).

2. **Existing** Workflow Systems [Электрон. ресурс] https://s.apache.org/existing-workflow-systems (дата обращения 24.02.2023).

3. **Rodriguez M.A., Buyya R.** Scheduling Dynamic Workloads in Multi-tenant Scientific Workflow as a Service Platforms // Future Generation Computer Systems. 2018. V. 79. Pp. 739—750.

4. **Hilman M.H., Rodriguez M.A., Buyya R.** Workflow-as-a-Service Cloud Platform and Deployment of Bioinformatics Workflow Applications [Электрон. ресурс] https://www.researchgate.net/scientific-contributions/Maria-A-Rodriguez-2114894132 (дата обращения 24.02.2023).

5. **Abrishami S., Naghibzadeh M., Epema D.H.** Deadline-constrained Workflow Scheduling Algorithms for Infrastructure as a Service Clouds // Future Generation Computer Systems. 2013. V. 29(1). Pp. 158—169.

6. **Calheiros R.N., Buyya R.** Meeting Deadlines of Scientific Workflows in Public Clouds with Tasks Replication // IEEE Trans. Parallel Distrib. Syst. 2014. V. 25(7). Pp. 1787—1796.

7. **Liu S., Ren K., Deng K., Song J.** A Task Backfill-based Scientific Workflow Scheduling Strategy on Cloud Platform // Proc. Intern. Conf. Information Sci. and Technol. 2016. Pp. 105—110.

8. **Deldari A., Naghibzadeh M., Abrishami S.** CCA: a Deadline-constrained Workflow Scheduling Algorithm for Multicore Resources on the Cloud // J. Supercomput. 2017. V. 73(2). Pp. 756—781.

9. **Maechling P. e. a.** SCEC CyberShake Workflows — Automating Probabilistic Seismic Hazard Ana-

References

1. **Workflows** Community Summit [Electron. Resurs] https://workflowsri.org/summits/community (Data Obrashcheniya 24.02.2023).

2. **Existing** Workflow Systems [Electron. Resurs] https://s.apache.org/existing-workflow-systems (Data Obrashcheniya 24.02.2023).

3. **Rodriguez M.A., Buyya R.** Scheduling Dynamic Workloads in Multi-tenant Scientific Workflow as a Service Platforms. Future Generation Computer Systems. 2018;79:739—750.

4. **Hilman M.H., Rodriguez M.A., Buyya R.** Workflow-as-a-Service Cloud Platform and Deployment of Bioinformatics Workflow Applications [Electron. Resurs] https://www.researchgate.net/scientific-contributions/Maria-A-Rodriguez-2114894132 (Data Obrashcheniya 24.02.2023).

5. **Abrishami S., Naghibzadeh M., Epema D.H.** Deadline-constrained Workflow Scheduling Algorithms for Infrastructure as a Service Clouds. Future Generation Computer Systems. 2013;29(1):158—169.

6. **Calheiros R.N., Buyya R.** Meeting Deadlines of Scientific Workflows in Public Clouds with Tasks Replication. IEEE Trans. Parallel Distrib. Syst. 2014;25(7):1787—1796.

7. **Liu S., Ren K., Deng K., Song J.** A Task Backfill-based Scientific Workflow Scheduling Strategy on Cloud Platform. Proc. Intern. Conf. Information Sci. and Technol. 2016:105—110.

8. **Deldari A., Naghibzadeh M., Abrishami S.** CCA: a Deadline-constrained Workflow Scheduling Algorithm for Multicore Resources on the Cloud. J. Supercomput. 2017;73(2):756—781.

9. **Maechling P. e. a.** SCEC CyberShake Workflows — Automating Probabilistic Seismic Hazard Ana-

lysis Calculations // Workflows for E-Science. N.-Y.: Springer, 2007. Pp. 143—163.

10. **Deelman E. e. a.** The Cost of Doing Science on the Cloud: the Montage Example // Proc. ACM/IEEE Conf. Supercomputing. 2008. P. 50.

11. **Vockler J.-S. e. a.** Experiences Using Cloud Computing for a Scientific Workflow Application // Proc. Intern. Workshop Sci. Cloud Computing. 2011. Pp. 15—24.

12. **Malawski M., Juve G., Deelman E., Nabrzyski J.** Algorithms for Cost- and Deadline-constrained Provisioning for Scientific Workflow Ensembles in IaaS Clouds // Future Generation Computer Syst. 2015. V. 48. Pp. 1—18.

13. **Pietri I. e. a.** Energy-constrained Provisioning for Scientific Workflow Ensembles // Proc. International Conf. Cloud and Green Computing. 2013. Pp. 34—41.

14. **Jiang Q., Lee Y.C., Zomaya A.Y.** Executing Large Scale Scientific Workflow Ensembles in Public Clouds // Proc. Intern. Conf. Parallel Proc. 2015. Pp. 520—529.

15. **Bryk P., Malawski M., Juve G., Deelman E.** Storage-aware Algorithms for Scheduling of Workflow Ensembles in Clouds // J. Grid Comput. 2016. V. 14(2). Pp. 359—378.

16. **Yu Z., Shi W.** A Planner-guided Scheduling Strategy for Multiple Workflow Applications // Proc. Intern. Conf. Parallel Processing. 2008. Pp. 1—8.

17. **Jiang H.-J. e. a.** Scheduling Concurrent Workflows in HPC Cloud through Exploiting Schedule Gaps // Proc. Intern. Conf. Algorithms and Architectures for Parallel Processing. 2011. Pp. 282—293.

18. **Stavrinides G.L., Karatza H.D.** A Cost-effective and QoS-aware Approach to Scheduling Real-time Workflow Applications in PaaS and SaaS Clouds // Proc. Intern. Conf. Future Internet of Things and Cloud. 2015. Pp. 231—239.

19. **Xu M., Cui L., Wang H., Bi Y.** A Multiple QoS-constrained Scheduling Strategy of Multiple Workflows for Cloud Computing // Proc. IEEE Intern. Symp. Parallel and Distributed Processing with Appl. 2009. Pp. 629—634.

20. **Chen W., Lee Y.C., Fekete A., Zomaya A.Y.** Adaptive Multiple-workflow Scheduling with Task Rearrangement // J. Supercomput. 2015. V. 71(4). Pp. 1297—1317.

21. **Zhou A.C., He B., Liu C.** Monetary Cost Optimizations for Hosting Workflow-as-a-Service in IaaS Clouds // IEEE Trans. Cloud Comput. 2016. V. 4(1). Pp. 34—48.

22. **Mao M., Humphrey M.** Auto-scaling to Minimize Cost and Meet Application Deadlines in Cloud Workflows // Proc. Intern. Conf. High Performance Computing, Networking, Storage and Analysis. 2011. P. 49.

23. **Shi J., Luo J., Dong F., Zhang J.** A Budget and Deadline-aware Scientific Workflow Resource Provisioning and Scheduling Mechanism for Cloud // Proc. IEEE Intern. Conf. Computer Supported Cooperative Work in Design. 2014. Pp. 672—677.

24. **Wang J. e. a.** Workflow as a Service in the Cloud: Architecture and Scheduling Algorithms // Proc. Comput. Sci. 2014. V. 29. Pp. 546—556.

lysis Calculations. Workflows for E-Science. N.-Y.: Springer, 2007:143—163.

10. **Deelman E. e. a.** The Cost of Doing Science on the Cloud: the Montage Example. Proc. ACM/IEEE Conf. Supercomputing. 2008:50.

11. **Vockler J.-S. e. a.** Experiences Using Cloud Computing for a Scientific Workflow Application. Proc. Intern. Workshop Sci. Cloud Computing. 2011:15—24.

12. **Malawski M., Juve G., Deelman E., Nabrzyski J.** Algorithms for Cost- and Deadline-constrained Provisioning for Scientific Workflow Ensembles in IaaS Clouds. Future Generation Computer Syst. 2015;48:1—18.

13. **Pietri I. e. a.** Energy-constrained Provisioning for Scientific Workflow Ensembles. Proc. International Conf. Cloud and Green Computing. 2013:34—41.

14. **Jiang Q., Lee Y.C., Zomaya A.Y.** Executing Large Scale Scientific Workflow Ensembles in Public Clouds. Proc. Intern. Conf. Parallel Proc. 2015:520—529.

15. **Bryk P., Malawski M., Juve G., Deelman E.** Storage-aware Algorithms for Scheduling of Workflow Ensembles in Clouds. J. Grid Comput. 2016;14(2): 359—378.

16. **Yu Z., Shi W.** A Planner-guided Scheduling Strategy for Multiple Workflow Applications. Proc. Intern. Conf. Parallel Processing. 2008:1—8.

17. **Jiang H.-J. e. a.** Scheduling Concurrent Workflows in HPC Cloud through Exploiting Schedule Gaps. Proc. Intern. Conf. Algorithms and Architectures for Parallel Processing. 2011:282—293.

18. **Stavrinides G.L., Karatza H.D.** A Cost-effective and QoS-aware Approach to Scheduling Real-time Workflow Applications in PaaS and SaaS Clouds. Proc. Intern. Conf. Future Internet of Things and Cloud. 2015:231—239.

19. **Xu M., Cui L., Wang H., Bi Y.** A Multiple QoS-constrained Scheduling Strategy of Multiple Workflows for Cloud Computing. Proc. IEEE Intern. Symp. Parallel and Distributed Processing with Appl. 2009:629—634.

20. **Chen W., Lee Y.C., Fekete A., Zomaya A.Y.** Adaptive Multiple-workflow Scheduling with Task Rearrangement. J. Supercomput. 2015;71(4):1297—1317.

21. **Zhou A.C., He B., Liu C.** Monetary Cost Optimizations for Hosting Workflow-as-a-Service in IaaS Clouds. IEEE Trans. Cloud Comput. 2016;4(1):34—48.

22. **Mao M., Humphrey M.** Auto-scaling to Minimize Cost and Meet Application Deadlines in Cloud Workflows. Proc. Intern. Conf. High Performance Computing, Networking, Storage and Analysis. 2011:49.

23. **Shi J., Luo J., Dong F., Zhang J.** A Budget and Deadline-aware Scientific Workflow Resource Provisioning and Scheduling Mechanism for Cloud. Proc. IEEE Intern. Conf. Computer Supported Cooperative Work in Design. 2014:672—677.

24. **Wang J. e. a.** Workflow as a Service in the Cloud: Architecture and Scheduling Algorithms. Proc. Comput. Sci. 2014;29:546—556.

25. **Gerlach W. e. a.** Skyport: Container-based Execution Environment Management for Multi-cloud Scientific Workflows // Proc. Intern. Workshop on Data-Intensive Computing in the Clouds. 2014. Pp. 25—32.

26. **Filgueira R. e. a.** Asterism: Pegasus and Dispel4py Hybrid Workflows for Data-intensive Science // Proc. Intern. Workshop Data-Intensive Computing in the Cloud. 2016. Pp. 1—8.

27. **Esteves S., Veiga L.** Waas: Workflow-as-a-service for the Cloud with Scheduling of Continuous and Data-intensive Workflows // Comput. J. 2016. V. 59(3). Pp. 371—383.

28. **Hashem I.A.T. e. a.** The Rise of "Big Data" on Cloud Computing: Review and Open Research Issues // Inf. Syst. 2015. V. 47. Pp. 98—115.

29. **Senyo P.K., Addae E., Boateng R.** Cloud Computing Research: a Review of Research Themes, Frameworks, Methods and Future Research Directions // Int. J. Inf. Manag. 2018. V. 38(1). Pp. 128—139.

30. **Stergiou C., Psannis K.E., Kim B.G., Gupta B.** Secure Integration of IoT and Cloud Computing // Future Gener. Comput. Syst. 2018. V. 78. Pp. 964—975.

31. **Kumar M.R.V., Raghunathan S.** Heterogeneity and Thermal-aware Adaptive Heuristics for Energy Efficient Consolidation of Virtual Infrastructure Clouds // J. Comput. Syst. Sci. 2016. V. 82(2). Pp. 191—212.

32. **Lopez-Pires F., Baran B.** Virtual Machine Placement Literature Review [Электрон. ресурс] https://arxiv.org/abs/1506.01509 (дата обращения 24.02.2023).

33. **Usmani Z., Singh S.** A Survey of Virtual Machine Placement Techniques in a Cloud Data Center // Proc. Comput. Sci. 2016. V. 78. Pp. 491—498.

34. **Liu X.F. e. a.** An Energy Efficient ant Colony System for Virtual Machine Placement in Cloud Computing // IEEE Trans. Evol. Comput. 2016. V. 22(1). Pp. 113—128.

35. **Abdel-Basset M., Abdle-Fatah L., Sangaiah A.K.** An Improved Lévy Based Whale Optimization Algorithm for Bandwidth-efficient Virtual Machine Placement in Cloud Computing Environment // Cluster Computing. 2018. V. 22(1). Pp. 1—16.

36. **Shabeera T.P., Kumar S.M., Salam S.M., Krishnan K.M.** Optimizing VM Allocation and Data Placement for Data-intensive Applications in Cloud using ACO Metaheuristic Algorithm // Eng. Sci. Technol. Int. J. 2017. V. 20(2). Pp. 616—628.

37. **Abdelaziz A. e. a.** Intelligent Algorithms for Optimal Selection of Virtual Machine in Cloud Environment, Towards Enhance Healthcare Services // Proc. Intern. Conf. Advanced Intelligent Systems and Informatics. 2017. Pp. 289—298.

38. **Ghobaei-Arani M., Shamsi M., Rahmanian A.A.** An Efficient Approach for Improving Virtual Machine Placement in Cloud Computing Environment // J. Exp. Theor. Artif. Intell. 2017. V. 29(6). Pp. 1149—171.

39. **Saber T., Thorburn J., Murphy L., Ventresque A.** VM Reassignment in Hybrid Clouds for Large

25. **Gerlach W. e. a.** Skyport: Container-based Execution Environment Management for Multi-cloud Scientific Workflows. Proc. Intern. Workshop on Data-Intensive Computing in the Clouds. 2014:25—32.

26. **Filgueira R. e. a.** Asterism: Pegasus and Dispel4py Hybrid Workflows for Data-intensive Science. Proc. Intern. Workshop Data-Intensive Computing in the Cloud. 2016:1—8.

27. **Esteves S., Veiga L.** Waas: Workflow-as-a-service for the Cloud with Scheduling of Continuous and Data-intensive Workflows. Comput. J. 2016;59(3):371—383.

28. **Hashem I.A.T. e. a.** The Rise of "Big Data" on Cloud Computing: Review and Open Research Issues. Inf. Syst. 2015;47:98—115.

29. **Senyo P.K., Addae E., Boateng R.** Cloud Computing Research: a Review of Research Themes, Frameworks, Methods and Future Research Directions. Int. J. Inf. Manag. 2018;38(1):128—139.

30. **Stergiou C., Psannis K.E., Kim B.G., Gupta B.** Secure Integration of IoT and Cloud Computing. Future Gener. Comput. Syst. 2018;78:964—975.

31. **Kumar M.R.V., Raghunathan S.** Heterogeneity and Thermal-aware Adaptive Heuristics for Energy Efficient Consolidation of Virtual Infrastructure Clouds. J. Comput. Syst. Sci. 2016;82(2):191—212.

32. **Lopez-Pires F., Baran B.** Virtual Machine Placement Literature Review [Electron. Resurs] https://arxiv.org/abs/1506.01509 (Data Obrashcheniya 24.02.2023).

33. **Usmani Z., Singh S.** A Survey of Virtual Machine Placement Techniques in a Cloud Data Center. Proc. Comput. Sci. 2016;78:491—498.

34. **Liu X.F. e. a.** An Energy Efficient ant Colony System for Virtual Machine Placement in Cloud Computing. IEEE Trans. Evol. Comput. 2016;22(1):113—128.

35. **Abdel-Basset M., Abdle-Fatah L., Sangaiah A.K.** An Improved Lévy Based Whale Optimization Algorithm for Bandwidth-efficient Virtual Machine Placement in Cloud Computing Environment. Cluster Computing. 2018;22(1):1—16.

36. **Shabeera T.P., Kumar S.M., Salam S.M., Krishnan K.M.** Optimizing VM Allocation and Data Placement for Data-intensive Applications in Cloud using ACO Metaheuristic Algorithm. Eng. Sci. Technol. Int. J. 2017;20(2):616—628.

37. **Abdelaziz A. e. a.** Intelligent Algorithms for Optimal Selection of Virtual Machine in Cloud Environment, Towards Enhance Healthcare Services. Proc. Intern. Conf. Advanced Intelligent Systems and Informatics. 2017:289—298.

38. **Ghobaei-Arani M., Shamsi M., Rahmanian A.A.** An Efficient Approach for Improving Virtual Machine Placement in Cloud Computing Environment. J. Exp. Theor. Artif. Intell. 2017;29(6):1149—171.

39. **Saber T., Thorburn J., Murphy L., Ventresque A.** VM Reassignment in Hybrid Clouds for Large

Decentralized Companies: a Multi-objective Challenge // Future Gener. Comput. Syst. 2018. V. 79. Pp. 751—764.

40. **Gao Y. e. a.** A Multi-objective ant Colony System Algorithm for Virtual Machine Placement in Cloud Computing // J. Comput. Syst. Sci. 2013. V. 79(8). Pp. 1230—1242.

41. **Ganesan T., Vasant P., Litvinchev I.** Chaotic Simulator for Bilevel Optimization of Virtual Machine Placements in Cloud Computing // J. Operations Research Soc. China. 2022. V. 10(4). Pp. 703—723.

42. **Deelman E. e. a.** The Future of Scientific Workflows // Int. J. High Perform. Comput. Appl. 2018. V. 32(1). Pp. 159—175.

43. **Chen W., Ferreira da Silva R., Livny M., Wenger K.** Pegasus: a Workflow Management System for Science Automation // Futur. Gener. Comput. Syst. 2015. V. 46. Pp. 17—35.

44. **Apache** Airflow Documentation [Электрон. ресурс] https://airflow.apache.org/ (дата обращения 24.02.2023).

45. **Hull D. e. a.** Taverna: a Tool for Building and Running Workflows of Services // Nucleic Acids Research. 2006. V. 34. Pp. 729—732.

46. **Altintas I. e. a.** Kepler: an Extensible System for Design and Execution of Scientific Workflows // Proc. XVI Intern. Conf. Scientific and Statistical Database Management. 2004. Pp. 423—424.

47. **Kramer M., Wurz H.M., Altenhofen C.** Executing Cyclic Scientific Workflows in the Cloud // J. Cloud Computing: Advances, Systems and Appl. 2021. V. 10(25). Pp. 1—26.

48. **Toporkov V., Yemelyanov D., Toporkova A.** Coordinated Global and Private Job-flow Scheduling in Grid Virtual Organizations // Simulation Modelling Practice and Theory. 2021. V. 107(14). P. 102228.

49. **Toporkov V., Yemelyanov D.** Micro-scheduling for Dependable Resources Allocation // Performance Evaluation Models for Distributed Service Networks. Studies in Systems, Decision and Control. 2021. V. 343. Pp. 81—105.

50. **Toporkov V., Yemelyanov D., Grigorenko M.** Optimization of Resources Allocation in High Performance Computing under Utilization Uncertainty // Proc. Intern. Conf. Computational Sci. 2021. V. 14747. Pp. 540—553.

51. **Toporkov V., Yemelyanov D., Bulkhak A.** Machine Learning-Based Scheduling and Resources Allocation in Distributed Computing // Proc. Internat. Conf. Computational Sci. 2022. V. 13353. Pp. 3—16.

Decentralized Companies: a Multi-objective Challenge. Future Gener. Comput. Syst. 2018;79:751—764.

40. **Gao Y. e. a.** A Multi-objective ant Colony System Algorithm for Virtual Machine Placement in Cloud Computing. J. Comput. Syst. Sci. 2013;79(8):1230—1242.

41. **Ganesan T., Vasant P., Litvinchev I.** Chaotic Simulator for Bilevel Optimization of Virtual Machine Placements in Cloud Computing. J. Operations Research Soc. China. 2022;10(4):703—723.

42. **Deelman E. e. a.** The Future of Scientific Workflows. Int. J. High Perform. Comput. Appl. 2018;32(1): 159—175.

43. **Chen W., Ferreira da Silva R., Livny M., Wenger K.** Pegasus: a Workflow Management System for Science Automation. Futur. Gener. Comput. Syst. 2015;46: 17—35.

44. **Apache** Airflow Documentation [Electron. Resurs] https://airflow.apache.org/ (Data Obrashcheniya 24.02.2023).

45. **Hull D. e. a.** Taverna: a Tool for Building and Running Workflows of Services. Nucleic Acids Research. 2006;34:729—732.

46. **Altintas I. e. a.** Kepler: an Extensible System for Design and Execution of Scientific Workflows. Proc. XVI Intern. Conf. Scientific and Statistical Database Management. 2004:423—424.

47. **Kramer M., Wurz H.M., Altenhofen C.** Executing Cyclic Scientific Workflows in the Cloud. J. Cloud Computing: Advances, Systems and Appl. 2021;10(25): 1—26.

48. **Toporkov V., Yemelyanov D., Toporkova A.** Coordinated Global and Private Job-flow Scheduling in Grid Virtual Organizations. Simulation Modelling Practice and Theory. 2021;107(14):102228.

49. **Toporkov V., Yemelyanov D.** Micro-scheduling for Dependable Resources Allocation. Performance Evaluation Models for Distributed Service Networks. Studies in Systems, Decision and Control. 2021;343: 81—105.

50. **Toporkov V., Yemelyanov D., Grigorenko M.** Optimization of Resources Allocation in High Performance Computing under Utilization Uncertainty. Proc. Intern. Conf. Computational Sci. 2021;14747:540—553.

51. **Toporkov V., Yemelyanov D., Bulkhak A.** Machine Learning-Based Scheduling and Resources Allocation in Distributed Computing. Proc. Internat. Conf. Computational Sci. 2022;13353:3—16.

Сведения об авторах:

Топорков Виктор Васильевич — доктор технических наук, заведующий кафедрой вычислительных технологий НИУ «МЭИ», e-mail: ToporkovVV@mpei.ru

Емельянов Дмитрий Михайлович — кандидат технических наук, доцент кафедры вычислительных технологий НИУ «МЭИ», e-mail: YemelyanovDM@mpei.ru

Булхак Артем Николаевич — аспирант кафедры вычислительных технологий НИУ «МЭИ», e-mail: BulkhakAN@mpei.ru

Information about authors:

Toporkov Viktor V. — Dr.Sci. (Techn.), Head of Computational Technologies Dept., NRU MPEI, e-mail: ToporkovVV@mpei.ru
Yemelyanov Dmitriy M. — Ph.D. (Techn.), Assistant Professor of Computational Technologies Dept., NRU MPEI, e-mail: YemelyanovDM@mpei.ru
Bulkhak Artem N. — Ph.D.-student of Computational Technologies Dept., NRU MPEI, e-mail: BulkhakAN@mpei.ru

Работа выполнена при поддержке: Российского научного фонда (грант № 22-21-00372), https://rscf.ru/project/22-21-00372/
The work is executed at support: Russian Science Foundation (Grant No. 22-21-00372), https://rscf.ru/project/22-21-00372/

Конфликт интересов: авторы заявляют об отсутствии конфликта интересов
Conflict of interests: the authors declare no conflict of interest

Статья поступила в редакцию: *01.03.2023*
The article received to the editor: *01.03.2023*

Статья принята к публикации: *06.06.2023*
The article has been accepted for publication: *06.06.2023*