УДК 004.7

АЛГОРИТМ НАСТРОЙКИ ПАРАМЕТРОВ РАСПИСАНИЯ ОБЛАЧНЫХ ВЫЧИСЛЕНИЙ НА ОСНОВЕ ОПТИМИЗАЦИИ РОЯ ЧАСТИЦ

А. А. Спицын, адъюнкт, Военный учебно-научный центр Военно-воздушных сил «Военно-воздушная академия имени профессора Н. Е. Жуковского и Ю. А. Гагарина», Воронеж, Россия; orcid.org/0000-0002-8650-6736, e-mail: csit@bk.ru

Облака типа «Infrastructure as a service» (IaaS) предлагают огромные возможности для решения масштабных задач. Выполнение рабочих процессов в таких средах может быть дорогостоящим по времени, если они не запланированы должным образом. Хотя планирование рабочих процессов в облаке широко изучено, большинство подходов фокусируется на двух требованиях пользователя к качеству обслуживания, а именно на времени выполнения и затратах. Необходимо учитывать и другие важные особенности облачных вычислений, гетерогенные облачные центры обработки данных. ***Цель работы*** *– создание алгоритма настройки параметров расписания облачных вычислений на основе оптимизации роя частиц. Результаты моделирования показывают значительное улучшение с точки зрения времени выполнения передаваемых данных и надежности по сравнению с проверенным методом HEFT (гетерогенное самое раннее время выполнения) для эталонных рабочих процессов.*

Ключевые слова: *облако IaaS, рабочий процесс, планирование, качество обслуживания, время передачи данных, надежность, гетерогенные центры, время выполнения.*

DOI: 10.21667/1995-4565-2021-75-44-52

Введение

Требования к рабочим процессам приложений в области данных и вычислений предполагают более высокую производительность вычислительной среды для того, чтобы выполнять их в разумные сроки и без сбоев. Проблема планирования рабочего процесса является NP-полной [1] и на протяжении многих лет широко изучалась в контексте распределенных систем, главным образом в грид-средах. Однако с появлением инфраструктуры облачных вычислений планирование научных рабочих процессов по-прежнему остается фундаментальной проблемой. Облачные вычисления – это модель обеспечения доступа к динамической сети по требованию к пулу общих ресурсов, который может быть быстро обеспечен [2] и который затрудняет удовлетворение некоторых ограничений качества обслуживания (QoS). Ограничения качества обслуживания используются в задачах планирования рабочих процессов в рамках сети [3] или облачной системы [4]. Среди всех этих ограничений – время выполнения и стоимость – это два основных ограничения, которые широко изучаются в [5].

Однако другие ограничения QoS, такие как надежность, могут быть полезны из-за их влияния на проблемы планирования рабочих процессов для облачных систем для достижения приемлемой производительности.

Известно, что такое планирование – это NP-полная задача [6]. Именно по этой причине были предложены для решения этой проблемы многие эвристики [7] и мета-эвристики, такие как генетический алгоритм (GA) [8], оптимизация роя частиц (PSO) [4] и оптимизация муравьиным алгоритмом (ACO) [2].

Фактически облачные сервисы, которые полагаются на виртуальные машины (VM), расположены в нескольких распределенных центрах обработки данных, что требует учета времени передачи данных и надежности сети.

В работе предлагается стратегия, ориентированная на надежность и основанная на дискретном PSO (DPSO) для планирования рабочих процессов в нескольких облачных центрах

обработки данных, где надежность определяется совместно вычислительными ресурсами и сетевыми связями. Подход уделяет внимание минимизации времени выполнения рабочего процесса, включая как время выполнения задачи, так и время передачи данных, одновременно повышая общую надежность приложений.

Существующие исследования

Рассмотрим существующие исследования, связанные с оптимизацией передачи данных и повышением ограничений надежности в распределенных вычислительных системах.

В [7] рассматривается проблема оптимизации надежности с учетом времени передачи данных. Представлены две двухцелевые эвристики планирования для минимизации времени и повышения надежности приложений в гетерогенных вычислительных системах. Первой из них было усовершенствование традиционного алгоритма динамического планирования уровня (DLS) [9], называемого двухобъектным динамическим планированием уровня (BDLS). Вторая была основана на генетических алгоритмах и называется двухобъектным генетическим алгоритмом (BGA), оптимизирующим те же критерии, что и первая.

Для независимых задач был представлен двухкритериальный алгоритм планирования [10], оптимизирующий время выполнения и надежность в гетерогенных системах. Авторы показали, что задачи должны быть сопоставлены с ресурсом таким образом, чтобы произведение времени выполнения задачи и частоты отказов ресурса можно было свести к минимуму. Затем эта идея была использована для улучшения алгоритма HEFT [11], чтобы рассмотреть надежность, дающую эвристику reliable-HEFT.

В [12] предложен алгоритм планирования рабочего процесса на основе ограничений для максимизации надежности. Созданы три новые эвристики планирования рабочего процесса, основанные на муравьином алгоритме. Совокупность этих эвристик сводила к минимуму нарушения ограничений по времени, стоимости и надежности. Для простоты предполагалось, что сеть надежна, но это нереально в среде нескольких облачных центров обработки данных.

Формализация задачи планирования

Расписание определяется с точки зрения отображения приложений, инфраструктуры ресурсов и критериев производительности. В данной работе критерием являются время выполнения и надежность.

Контекст

Подход предлагается в контексте облака IaaS с несколькими центрами обработки данных, где приложения рабочего процесса должны быть запланированы для выполнения в разных центрах обработки данных. Чтобы достичь минимизации времени выполнения приложения при одновременном повышении надежности, моделируется время передачи данных при вычислении времени выполнения и учитывается надежность связи (надежность канала) для выражения надежности ресурсов.

Модель приложения

Приложение рабочего процесса (w) моделируется как ориентированный ациклический граф (DAG):

$$G = (N; E),$$

где N – множество n узлов, обозначающих задачи $t_i (1 \le i \le n)$, а E – набор направленных ребер. Ребро $e(i, j) \in E$ соответствует зависимости между задачей t_i и задачей t_j, в которой t_i является непосредственной родительской задачей t_j, а t_j – непосредственной дочерней задачей t_i. Дочерняя задача не может быть выполнена до тех пор, пока не будут выполнены все ее родительские задачи. Матрица данных размерности $n \times n$ представляет объем данных, которыми обмениваются задачи.

Модель облачных ресурсов

Представим ресурсную модель, которая учитывает время передачи данных между несколькими центрами обработки данных, чтобы компьютерные ресурсы могли быть развернуты на разных региональных центрах обработки данных. Кроме того, рассматривается надежность как компьютерных ресурсов, так и сетевых связей.

Модель времени

В облаке IaaS компьютерные ресурсы создаются в виде виртуальных машин (VM). Как правило, облачные провайдеры предлагают несколько типов виртуальных машин. Каждый тип vm_i определяется с точки зрения его вычислительной мощности CC_{vm_p}. Предполагаем, что для каждого типа виртуальной машины можно оценить ее вычислительную мощность в операциях с плавающей запятой в секунду. Таким образом, можно оценить время выполнения задачи на данном типе виртуальной машины. Кроме того, определим время вычислений $CT(t_i, vm_p)$ как расчетное время выполнения задачи t_i в виртуальной машине типа vm_p в соответствии с ее мощностью. I_{t_i} аналогично определяется в операциях с плавающей запятой и вычисляется из уравнения (1):

$$CT(t_i, vm_p) = \frac{I_{t_i}}{CC_{vm_p}}. \quad (1)$$

В нескольких облачных центрах обработки данных виртуальные машины развертываются в территориально разных регионах. Поэтому время передачи данных становится значительным и напрямую влияет на время отклика приложений рабочего процесса. Следовательно, необходимо учитывать эти времена при расчете времени выполнения рабочего процесса. Рассмотрим U центров обработки данных в облачной инфраструктуре.

Определим время передачи $TT_{(i,j)}$ между двумя задачами, соответствующее весу ребра $(i, j) \in E$ в графе приложения DAG, которое соответствует времени, затраченному на передачу данных из задачи t_i (выполняемой на vm_p, размещенной в центре обработки данных U_a) в задачу t_j (выполняется на vm_k, поданном в дата-центр U_b), как в следующем уравнении:

$$TT_{(i,j)} = \frac{data_{ij}}{Transfer_{rate_{(p,k)}}}, \quad (2)$$

где $data_{ij}$ – размер выходных данных, полученных задачей t_i и переданных в t_j.

Поскольку виртуальные машины расположены в разных центрах обработки данных, скорости передачи $Transfer_{rate_{(p,k)}}$ между ними неоднородны. При выполнении двух взаимодействующих задач на одной виртуальной машине время передачи равно нулю, и при запуске на разных виртуальных машинах, размещенных в одном центре обработки данных, время передачи равно нулю. Таким образом, предлагается новая модель, описывающая несколько платформ облачных центров обработки данных. В модели общее время вычислений $TCT(t_i, vm_p)$ задачи в виртуальной машине вычисляется так, как показано в (3):

$$TCT(t_i, vm_p) = CT(t_i, vm_p) + \sum_{i=1}^{m} s_m r_m TT_{(i,j)}, \quad (3)$$

где m – число ребер, в которых t_i является родительской задачей. Логическое значение s_m равно 0 всякий раз, когда t_i и t_j работают на одной и той же виртуальной машине ($p = k$), или 1 в противном случае. Кроме того, r_m равно 0 всякий раз, когда vm_p и vm_k размещаются в одном и том же центре обработки данных ($U_a = U_b$), или 1 в противном случае.

Общее время вычислений $TCT(w)$ всех задач в рабочем процессе определяется из (4):

$$TCT(w) = \max\{TCT(t_i;\ vm_p)\}. \quad (4)$$

Модель надежности

Под надежностью будем понимать частоту отказов. Предположим, что отказы виртуальных машин и каналов связи в системе имеют пуассоновское распределение [5]. Свяжем частоту отказов для каждой виртуальной машины и канала связи. Пусть λ_p обозначает интенсивность отказов vm_p в единицу времени, а $\lambda_{L_{(k;p)}}$ – интенсивность отказов линии связи $L_{(k;p)}$ между vm_k и vm_p.

В модели надежность задачи t_i равна вероятности ее успешного выполнения на виртуальной машине, которой она назначена, и вероятности успешной передачи данных этой задаче от ее непосредственных предшественников. Используем аппроксимацию надежности сети для передачи данных, предложенную в [13]. Надежность $R(t_i, vm_p)$ t_i, выполненного или отображенного на vm_p, вычисляется следующим образом:

$$R(t_i, vm_p) = e^{-\lambda_p CT(t_i, vm_p)}. \quad (5)$$

Поскольку время связи между задачами, назначенными на одной и той же виртуальной машине, равно нулю, то передача данных считается безотказной. Таким образом, элемент $R_{(j;i)}$, соответствующий надежности передачи между t_j, выполняемой на vm_k, и t_i, выполняемой на vm_p, может быть вычислен следующим образом:

$$R_{(i,j)} = e^{-\lambda_{L_{(k,p)}} TT_{(j,i)}}. \quad (6)$$

С учетом (5) и (6) надежность $R(t_i)$ задачи t_i может быть определена из (7), где $pred_{t_i}$ – множество задач-предшественников задачи t_i:

$$R(t_i) = \prod_{t_j \in pred_{t_i}} R(t_i, vm_p) R_{(i,j)}. \quad (7)$$

Надежность рабочего процесса *w* равна надежности всех его задач и может быть выражена следующим образом:

$$R(\omega) = \prod_{t_i \in N} R(t_i). \quad (8)$$

Критерий модели

Цель подхода к планированию состоит в том, чтобы повысить надежность и минимизировать время выполнения и передачи данных приложения рабочего процесса. Чтобы максимизировать качество планирования рабочего процесса, необходимо минимизировать индекс:

$$R_{index}(\omega) = \sum_{vm_p \in VM} \sum_{t_i \in N} (\lambda_p CT(t_i, vm_p) + \lambda_{L_{(k,p)}} TT_{(j,i)}). \quad (9)$$

Задача может быть формально определена следующим образом: найти расписание *S* для рабочего процесса *w* с минимальным $TCT(w)$ и максимальным $R(\omega)$ [таким образом, с минимальным $R_{index}(\omega)$].

Чтобы решить эту проблему, предлагается стратегия, основаннfz на популяционной метаэвристике PSO. PSO имеет меньше параметров, чем другие метаэвристики, такие как GA или ACO, и быстро сходится.

Планирование с учетом надежности на основе оптимизации роя частиц

Дадим краткое описание эвристики PSO. Затем выделим дискретный PSO, ориентированный на надежность, RDPSO.

PSO – популяционная метаэвристика, вдохновленная стадными моделями социального поведения [14]. Алгоритм стохастической оптимизации основан на концепции, называемой частицей. Частица соответствует особи, которая обладает способностью перемещаться или

летать в заданном проблемном пространстве. Каждая частица характеризуется положением *X*, скоростью *V* и значением пригодности.

Положения частиц рассматриваются как потенциальные решения задачи и оцениваются с помощью фитнес-функции. Поставлена задача оптимизировать фитнес-функцию и дать ей возможность отслеживать лучшую позицию *pbest* и глобально лучшую позицию *gbest*. Скорость частицы представляет собой направление и величину следующего движения. Она вычисляется с учетом фактической скорости, *pbest* и *gbest* на каждом шаге алгоритма. Движение частиц к этим значениям обусловлено их фактической скоростью (взвешенной коэффициентом инерции), а также промежутками между их положением и значением *pbest* и *gbest* (взвешенными случайными числами для когнитивного и социального коэффициентов соответственно).

На каждой итерации алгоритма PSO генерируется значение целевой функции. Затем обновляются *pbest* и *gbest*. После этого положение и скорость частицы обновляются на основе уравнений (10) и (11) соответственно.

$$v_i^{k+1} = \omega v_i^k + c_1 r_1 [pbest_i^k - x_i^k] + c_2 r_2 [qbest_i^k - x_i^k], \quad (10)$$

$$x_i^{k+1} = x_i^k + v_i^{k+1}, \quad (11)$$

где v_i^k – скорость частицы i на итерации k; v_i^{k+1} – скорость частицы i на итерации $k+1$; ω – инерционный вес; x_i^k – текущее положение частицы i на итерации k; x_i^{k+1} – положение частицы i на итерации k+1; $pbest_i$ – лучшее положение частицы i; $gbest$ – положение лучшей частицы в популяции; c_1, c_2 – коэффициенты ускорения; r_1, r_2 – случайное число между 0 и 1.

Алгоритм PSO повторяется до тех пор, пока не будет удовлетворен критерий остановки, который обычно представляет собой заранее определенное значение целевой функции, оцениваемое как приемлемое, или заданное максимальное число итераций.

В работе используется дискретный вариант PSO под названием discrete particle swarm Optimization (DPSO), который обрабатывает зависимости задач [15].

Каждое положение частицы – это возможное решение. В нашей модели частица имеет n измерений, соответствующих *n* задачам рабочего процесса. *N* задач должны быть назначены m виртуальным машинам, где $m < n$. Схема расположения частиц, показанная на рисунке 1, представляет собой возможное графическое решение.

t_1	t_2	t_3	…	t_n
vm_2	vm_1	vm_m	…	vm_1

Рисунок 1 – Сопоставление частиц
Figure 1 – Comparison of particle

Подход, основанный на DPSO, стремится оптимизировать время выполнения и надежность одновременно. Подход назван RDPSO для обеспечения надежности DPSO. Для оценки решения с RDPSO фитнес-функция выражается через обе метрики – время выполнения и надежность.

Использовали весовые коэффициенты взвешенной суммы для включения времени выполнения и индекса надежности в единую целевую функцию, как указано в (12). Время выполнения и коэффициент надежности нормализуются в этом уравнении. Нормализация позволяет выровнять шкалу сравниваемых величин. На самом деле нормализация необходима для того, чтобы функция пригодности не была искажена одним из показателей, имеющих высокое значение.

В алгоритме значения $T_{\max}$ и $R_{index_{\max}}$ представляют собой максимальное значение времени выполнения и надежности соответственно, найденное на предыдущей итерации.

Параметры α_1, α_2 – это относительные веса, присвоенные времени выполнения и индексу надежности в соответствии с требованиями пользователя к компромиссу, например $\alpha_1 + \alpha_2 = 1$.

$$Fitness = \alpha_1 \frac{TCT}{T_{\max}} + \alpha_2 \frac{R_{index}}{R_{index_{\max}}}. \quad (12)$$

Структура алгоритма RDPSO представлена на рисунке 2 с учетом особенностей фитнес-функции.

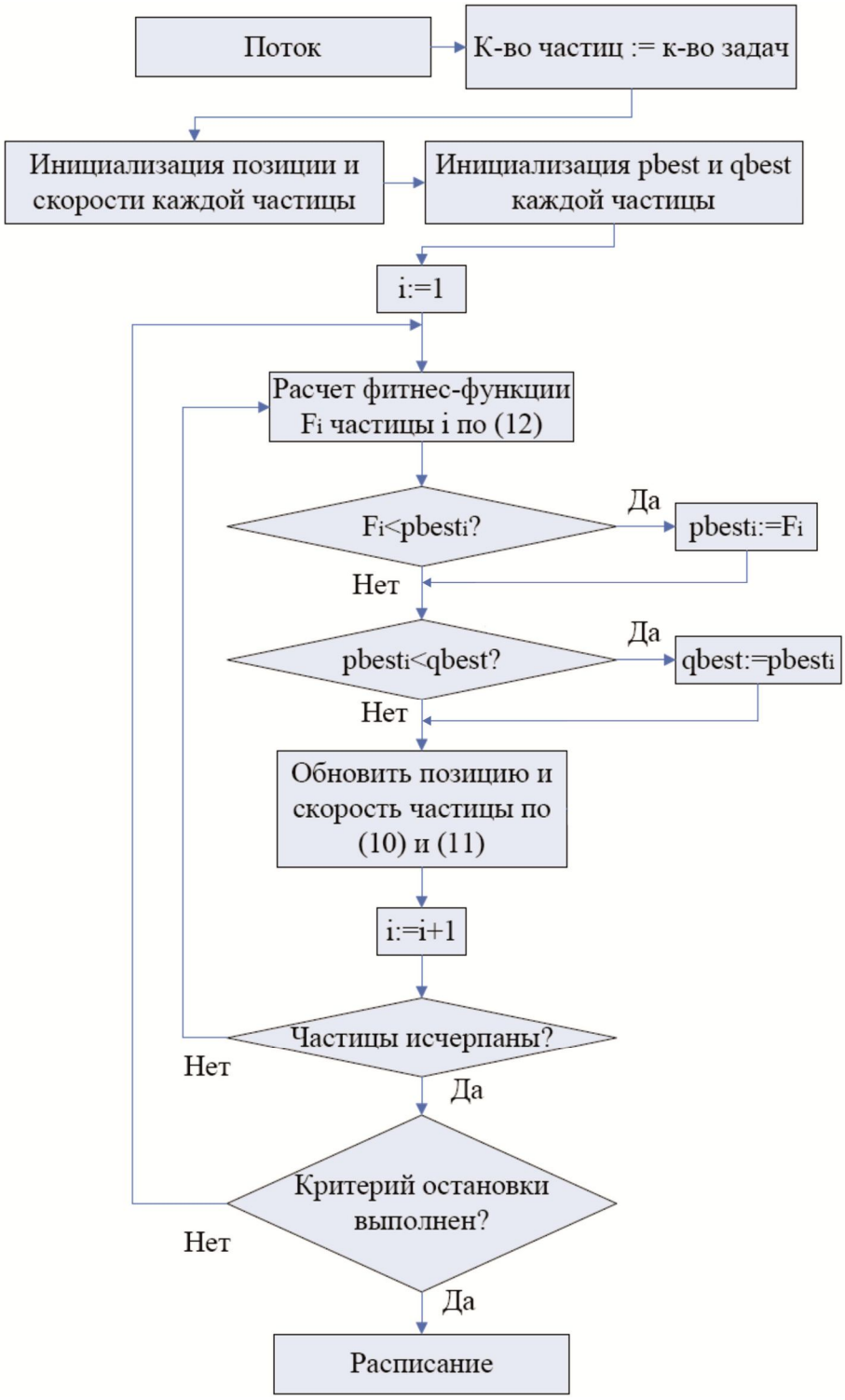

Рисунок 2 – Структура алгоритма RDPSO
Figure 2 – Block diagram of RDPSO algorithm

В алгоритме PSO для оценки качества решения целевая функция оценивается после вычисления $TCT(w)$ и $R_{index}(w)$ с использованием уравнений (4) и (9) соответственно.

Результаты экспериментов, реализованные для оценки эффективности предложенного подхода, показали его эффективность. Разработана имитационная модель, основанная на инструменте CloudSim [16] для моделирования облака с несколькими платформами центров обработки данных. Использованы различные рабочие процессы из разных областей с разной структурой и реализован стандартный подход для сравнения. Протестирован предложенный подход с помощью моделирования, но для большей адекватности использованы конфигурации виртуальных машин, предлагаемые поставщиком общедоступных облаков, и приложения на основе реальных рабочих процессов.

Заключение

Представлен подход к планированию RDPSO, основанный на PSO. Ставилась задача минимизировать время выполнения и время передачи данных, одновременно повышая надежность рабочего процесса в нескольких облачных центрах обработки данных. Учитывалась неоднородность ресурсов, сбои виртуальных машин и сбои связей.

Разработан модифицированный алгоритм роя частиц настройки параметров расписания облачных вычислений, отличающийся использованием весовой фитнес-функции и обеспечивающий минимизацию времени передачи данных, времени выполнения и повышение надежности выполнения рабочих процессов.

Проведена оценка алгоритма с использованием реальных рабочих процессов. Результаты показывают, что RDPSO превосходит алгоритм PSO. Время выполнения улучшается у RDPSO в большинстве случаев и общая надежность повышается. Предложенный подход позволяет реализовать передачу с меньшим объемом данных, чем алгоритм RHEFT, даже с использованием очень трудоемких рабочих процессов.

Библиографический список

1. **Ullman J. D.** NP-complete scheduling problems // Journal of Computer and System Sciences, 1975, vol. 10, no. 3, pp.384-393.

2. **Chen W-N., Zhang J.** An ant colony optimization approach to a grid workflow scheduling problem with various QoS requirements // IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 2009, vol. 39, no. 1, pp. 29-43.

3. **Mell P., Grance T.** The NIST Definition of Cloud Computing, National Institute of Standards and Technology. http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf. 2011.

4. **Chen W-N., Zhang J**. A set-based discrete PSO for cloud workflow scheduling with user-defined QoS constraints // 2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC). (2012), pp. 773-778.

5. **Wu F., Wu Q., Tan Y.** Workflow scheduling in cloud: a survey // The Journal of Supercomputing, 2015, vol. 71, no. 9, pp. 3373-3418.

6. **Hakem M., Butelle F.** Reliability and scheduling on systems subject to failures // International Conference on Parallel Processing, 2007. (ICPP 2007), pp. 38-48.

7. **Dogan A., Ozguner F.** Biobjective scheduling algorithms for execution time-reliability trade-off in heterogeneous computing systems // The Computer Journal, vol. 48, no. 3, pp. 300-314.

8. **Rehani N., Garg R.** Meta-heuristic based reliable and green workflow scheduling in cloud computing. International Journal of System Assurance Engineering and Management, 2017, vol. 9, no. 4, pp. 1-10.

9. **Sih G. C., Lee E.A.** A compile-time scheduling heuristic for interconnection-constrained heterogeneous processor architectures // IEEE Transactions on Parallel and Distributed systems, 1993, vol. 4, no. 2, pp. 175-187.

10. **Dongarra J. J., Jeannot E., Saule E., Shi Z.** Bi-objective scheduling algorithms for optimizing makespan and reliability on heterogeneous systems // Proceedings of the Nineteenth Annual ACM Symposium on Parallel Algorithms and Architectures, 2007, pp. 280-288, ACM.

11. **Topcuoglu H., Hariri S., Wu M-Y.** Performance-effective and low-complexity task scheduling for heterogeneous computing // IEEE Transactions on Parallel and Distributed Systems, 2002, vol. 13, no. 3, pp. 260-274.

12. **Kianpisheh S., Charkari N. M., Kargahi M.** Reliability-driven scheduling of time/cost-constrained grid workflows // Future Generation Computer Systems. 2016, vol. 55, no. C, pp. 1-16.

13. **Qin X., Jiang H.** A novel fault-tolerant scheduling algorithm for precedence constrained tasks in real-time heterogeneous systems // Parallel Computing. 2006, vol. 32, no. 5, pp. 331-356.

14. **Kennedy J.** Particle swarm optimization // Encyclopedia of Machine Learning. Springer, New York, NY, USA. 2011, pp. 760-766.

15. **Bouali L., Oukfif K., Bouzefrane S., Oulebsir-Boumghar F.** A hybrid algorithm for DAG application scheduling on computational grids. International Conference on Mobile, Secure and Programmable Networking, 2015, pp. 63-77, Springer.

16. **Calheiros R. N., Ranjan R., Beloglazov A., De Rose C. A., Buyya R.** CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms // Software: Practice and Experience. 2011, vol. 41, no. 1, pp. 23-50.

UDC 004.7

ALGORITHM FOR CONFIGURING CLOUD COMPUTING SCHEDULE PARAMETERS BASED ON PARTICLE SWARM OPTIMIZATION

A. A. Spitsyn, Associate Professor, Military Training and Research Center of the Air Force «Professor N. E. Zhukovsky and Yu. A. Gagarin Air Force Academy», Voronezh, Russian Federation;
orcid.org/0000-0002-8650-6736, e-mail: csit@bk.ru

Infrastructure as a service (IaaS) clouds offer huge opportunities for solving large-scale tasks. Running workflows in such environments can be time-consuming if they are not properly scheduled. Although workflow planning in industry is widely studied, most approaches are focused on two user requirements for the quality of service, namely, execution time and cost. It is also necessary to take into account other important features of cloud computing, heterogeneous cloud data centers. ***The aim is*** *to optimize data transfer time while minimizing execution time and improving reliability. Based on the simulation, the results show significant improvement in terms of execution time, transmitted data, and reliability compared to the proven HEFT (Heterogeneous Earliest Execution Time) method for reference workflows.*

Key words: *IaaS cloud, workflow, scheduling, quality of service, data transfer time, reliability, heterogeneous centers, execution time.*

DOI: 10.21667/1995-4565-2021-75-44-52

References

1. **Ullman J. D.** NP-complete scheduling problems. *Journal of Computer and System Sciences*. 1975, vol. 10, no. 3, pp. 384-393.

2. **Chen W-N., Zhang J.** An ant colony optimization approach to a grid workflow scheduling problem with various QoS requirements. *IEEE Transactions on Systems, Man, and Cybernetics*, Part C (Applications and Reviews). 2009, vol. 39, no. 1, pp.29-43.

3. **Mell P., Grance T.** *The NIST Definition of Cloud Computing, National Institute of Standards and Technology*. http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf. 2011.

4. **Chen W-N., Zhang J.** A set-based discrete PSO for cloud workflow scheduling with user-defined QoS constraints. *2012 IEEE International Conference on Systems, Man, and Cybernetics* (*SMC*). 2012, pp. 773-778.

5. **Wu F., Wu Q., Tan Y.** Workflow scheduling in cloud: a survey. *The Journal of Supercomputing*. 2015, vol. 71, no. 9, pp.3373-3418.

6. **Hakem M., Butelle F.** Reliability and scheduling on systems subject to failures. *International Conference on Parallel Processing.* 2007. ICPP 2007, pp. 38-38.

7. **Dogan A., Ozguner F.** Biobjective scheduling algorithms for execution time-reliability trade-off in heterogeneous computing systems. *The Computer Journal*, vol. 48, no. 3, pp. 300-314.

8. **Rehani N., Garg R.** Meta-heuristic based reliable and green workflow scheduling in cloud computing. *International Journal of System Assurance Engineering and Management*. 2017, vol. 9, no. 4, pp. 1-10.

9. **Sih G. C., Lee E. A.** A compile-time scheduling heuristic for interconnection-constrained heterogeneous processor architectures. *IEEE Transactions on Parallel and Distributed systems*. 1993, vol. 4, no. 2, pp. 175-187.

10. **Dongarra J. J., Jeannot E., Saule E., Shi Z.** Bi-objective scheduling algorithms for optimizing makespan and reliability on heterogeneous systems. *Proceedings of the Nineteenth Annual ACM Symposium on Parallel Algorithms and Architectures*. 2007, pp. 280-288, ACM.

11. **Topcuoglu H., Hariri S., Wu M-Y.** Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE Transactions on Parallel and Distributed Systems*. 2002, vol. 13, no. 3, pp. 260-274.

12. **Kianpisheh S., Charkari N. M., Kargahi M.** Reliability-driven scheduling of time/cost-constrained grid workflows. *Future Generation Computer Systems*. 2016, vol. 55, no. C, pp.1-16.

13. **Qin X., Jiang H.** A novel fault-tolerant scheduling algorithm for precedence constrained tasks in real-time heterogeneous systems. *Parallel Computing*, 2006, vol. 32, no. 5, pp. 331-356.

14. **Kennedy J.** Particle swarm optimization. *Encyclopedia of Machine Learning. Springer, New York, NY, USA*. 2011, pp.760-766.

15. **Bouali L., Oukfif K., Bouzefrane S., Oulebsir-Boumghar F.** A hybrid algorithm for DAG application scheduling on computational grids. *International Conference on Mobile, Secure and Programmable Networking*. 2015, pp. 63-77, Springer.

16. **Calheiros R. N., Ranjan R., Beloglazov A., De Rose C. A., Buyya R.** CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience.* 2011, vol. 41, no. 1, pp. 23-50.