УДК 004.9 DOI: 10.26102/2310-6018/2020.31.4.023

Распределение ресурсов и планирование заданий в облачной среде на основе алгоритма оптимизации роя частиц и R-фактора

А.А. Спицын, Д.И. Мутин
Военный учебно-научный центр Военно-воздушных сил «Военно-воздушная академия имени профессора Н.Е. Жуковского и Ю.А. Гагарина», Воронеж, Российская Федерация
МГТУ Станкин, Москва, Российская Федерация

Резюме: Облачные вычисления – это мощная технология вычислений, которая предоставляет гибкие услуги пользователю в любом месте. Управление ресурсами и планирование задач являются важнейшими перспективами облачных вычислений. Одной из главных проблем облачных вычислений было планирование задач. Обычно планирование задач и управление ресурсами в облаке – это сложная задача оптимизации во время рассмотрения потребностей в качестве обслуживания. Огромные работы в рамках планирования задач фокусируются только на вопросах крайних сроков и оптимизации затрат, а также избегают значения доступности, надежности и надежности. Основная цель данного исследования – разработка оптимизированного алгоритма эффективного распределения ресурсов и планирования в облачной среде. В этом исследовании используется алгоритм PSO и R-фактора. Основная цель алгоритма PSO заключается в том, чтобы задачи планировались на виртуальных машинах для сокращения времени ожидания и пропускной способности системы. PSO – это метод, порожденный социальным и коллективным поведением роев живых существ в природе, и в котором частицы ищут проблемное пространство, чтобы предсказать близкое к оптимальному или оптимальное решение. Разработан гибридный алгоритм, сочетающий PSO и R-фактор с целью снижения время обработки, сделать промежуток и стоимость выполнения задачи одновременно. Результаты испытаний и моделирования показывают, что предложенный метод обладает большей эффективностью, чем ранее распространенные подходы.

Ключевые слова: облачные вычисления, распределение ресурсов, алгоритм роя частиц, управление заданиями, моделирование.

Для цитирования: Спицын А.А., Мутин Д.И. Распределение ресурсов и планирование заданий в облачной среде на основе алгоритма оптимизации роя частиц и R-фактора. *Моделирование, оптимизация и информационные технологии.* 2020;8(4). Доступно по: https://moitvivt.ru/ru/journal/pdf?id=869 DOI: 10.26102/2310-6018/2020.31.4.023

Resource allocation and task planning in a cloud environment based on the particle swarm and R-factor optimization algorithm

A.A. Spitsyn, D.I. Mutin
Military educational scientific center air force "air force Academy named after Professor N. E. Zhukovsky and Y.A. Gagarin», Russian Federation
MSTU Stankin, Moscow, Russian Federation

Abstract: Cloud computing is a powerful computing technology that provides flexible services to the user anywhere. Resource management and task planning are the most important perspectives of cloud computing. One of the main challenges of cloud computing was scheduling tasks. Typically, task planning and resource management in the cloud is a complex optimization task while considering quality

of service requirements. Huge work within task planning focuses only on issues of deadlines and cost optimization, and avoids the importance of availability, reliability, and reliability. The main goal of this study is to develop an optimized algorithm for efficient resource allocation and planning in a cloud environment. This study uses the PSO and R-factor algorithm. The main purpose of the PSO algorithm is to have tasks scheduled on virtual machines to reduce latency and system throughput. PSO is a method generated by the social and collective behavior of swarms of living things in nature, and in which particles search for a problem space to predict a near-optimal or optimal solution. A hybrid algorithm has been developed that combines PSO and R-factor in order to reduce processing time, make the gap and the cost of performing the task at the same time. The results of tests and simulations show that the proposed method is more effective than previously common approaches.

Keywords: cloud computing, resource allocation, particle swarm algorithm, task management, modeling.

For citation: Spitsyn A.A., Mutin D.I. Resource allocation and task planning in a cloud environment based on the particle swarm and R-factor optimization algorithm *Modeling, optimization and information technology*. 2020;8(4). Available from: https://moitvivt.ru/ru/journal/pdf?id=869 DOI: 10.26102/2310-6018/2020.31.4.023 (In Russ).

Введение

Облако – это распределенная и параллельная вычислительная система, состоящая из набора виртуализированных и взаимосвязанных ПК, которые динамически представляются и подготавливаются как более чем один унифицированный вычислительный ресурс на основе SLA (Service level agreement), устанавливаемого путем обсуждения между пользователями и поставщиками облачных услуг [1].

Облачные вычисления [2] – это крупномасштабная модель распределенных вычислений, которая опирается на экономический размер облачного оператора, который является динамичным, виртуализированным и абстрактным. Основная информация об облачных вычислениях – это способность справиться с объмом вычислений, хранилищ и различных видов услуг и платформ, которые выделяются внешним пользователям по требованию через интернет. Облачные вычисления [5] – это быстро развивающаяся парадигма вычислений с целью освобождения пользователей облака от управления аппаратным и программным обеспечением, информационными ресурсами и сетями и переноса этого бремени на облачный сервис.

Распределенная и параллельная система состоит из набора виртуальных и взаимосвязанных ПК, которые динамически подготавливаются и представляются как более чем один унифицированный ресурс вычислений на основе SL, настроенного путем переговоров между клиентами и поставщиком услуг. Существует [6] в основном 3 вида услуг, предлагаемых облаком. Во-первых, это IaaS (инфраструктура как услуга), которая предлагает пользователям облачную инфраструктуру – это инфраструктура для различных целей, а именно вычислительные ресурсы и система хранения данных. Во-вторых, платформа как услуга (PaaS), которая предлагает платформу клиентам, чтобы они могли делать свои приложения на этой платформе. В-третьих, программное обеспечение как услуга (SaaS), которая предоставляет программное обеспечение пользователям, так что пользователям не требуется устанавливать программное обеспечение на свои машины, и они могут использовать программное обеспечение непосредственно из облака.

Планирование задач [7] является одной из наиболее важных и существенных проблем в облачных вычислениях, и многие исследователи пытались предсказать оптимальное решение для планирования задач на существующих ресурсах в облачном окружении. Но проблема планирования задач – это NP-полная задача. В настоящее время эвристические алгоритмы оптимизации широко используются при решении NP-полных

задач. Эволюционный алгоритм используется для предсказания субоптимального решения проблемы за значительное время вычислений. Для быстрого достижения решения используется несколько эвристических методов.

Планирование задач играет важную роль в развитии надежности и гибкости облачных систем. На Рисунке 1 показан процесс планирования задач.

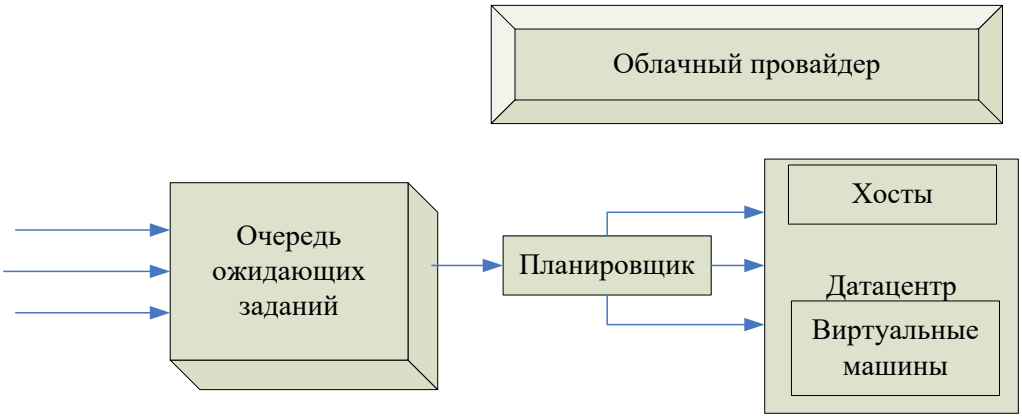

Рисунок – 1. Процесс планирования задач
Figure – 1. Task scheduling process

Основная причина планирования задач для ресурсов в соответствии с заданным периодом времени включает в себя прогнозирование наилучшей и полной последовательности, в которой различные задачи могут быть выполнены для обеспечения удовлетворительного и наилучшего результата для пользователя. Ресурсы в любом типе облачных вычислений часто распределяются динамически в соответствии с потребностями и последовательностью выполнения задачи, что приводит к планированию задач в облаке быть динамической проблемой, где последовательность может быть полезна во время обработки задач.

Планирование задач должно быть динамическим, чтобы поток задач и пути их выполнения были неопределенными, и в то же время ресурсы были неопределенными, потому что существует несколько задач, присутствующих для совместного использования задач одновременно в одно и то же время.

Проблема планирования задач состоит [8] из M машин и N задач. Каждая задача должна быть обработана одной из виртуальных машин M таким образом, чтобы в конечном итоге вся продолжительность планирования была бы сокращена.

Алгоритм планирования концентрируется на параметрах качества, а именно стоимость выполнения задач, период решения. Каждая задача может быть выполнена на одном ресурсе и не может быть приостановлена до конца. Поскольку алгоритм планирования статичен, ожидаемое время выполнения задачи j на I ресурсах считается заранее заданным. Цель алгоритма планирования состоит в том, чтобы представить каждое задание каждому ресурсу, чтобы сократить время потока и продолжительность выполнения заданий.

Выделение физических и/или виртуальных ресурсов [9] – это основной тип управления в облачной среде, поскольку его эффективность напрямую влияет на стоимость и производительность всей системы. Таким образом, неэффективное распределение ресурсов имеет отрицательный эффект влияния на стоимость и производительность. Основная цель распределения ресурсов состоит в том, чтобы наилучшим образом использовать ресурсы инфраструктуры и интегрировать их для достижения большей пропускной способности при решении проблем

крупномасштабных вычислений.

Ресурсы вычислений в таких средах распределяются, когда потребитель посылает запрос со своими потребностями. В [11] исследовано три различных эвристических алгоритма, генетический алгоритм, поиск блокировок и имитационный отжиг. Существующие решения в каждом поколении оцениваются индивидами и функцией пригодности, которые имеют лучшую ценность пригодности, генерируя новые решения с помощью операторов кроссовера и мутации. Одним из новейших эвристических алгоритмов является оптимизация роя частиц.

Частицы обозначаются как группа виртуальных машин, выделенных для выполнения задач. Каждая частица в поведении роя имеет 2 главных признака, а именно скорость v, указывающая на скорость движения, и положение x, обозначающее рекомендуемое местоположение.

Для частицы p лучшее решение известно как лучший личный опыт (pbest), в то время как среди всех частиц популяции лучшим решением является лучший групповой опыт (gbest). В любой момент времени на положение частицы влияет ее личное лучшее положение и положение другой лучшей частицы в глобальном проблемном пространстве.

Метаэвристический метод PSO (Particle Swarm Optimization) с собственным адаптивного поиска на основе методов оптимизации предложен в [3]. Алгоритм PSO не похож [4] на другие алгоритмы, основанные на популяции, а именно на генетические алгоритмы, которые не имеют прямой индивидуальной рекомбинации популяции. Алгоритм оптимизации роя частиц концентрируется на снижении общей вычислительной стоимости приложения рабочего процесса. В качестве показателя производительности используется полная стоимость выполнения приложения.

Основная цель – снизить общую стоимость выполнения рабочих процессов приложений в среде облачных вычислений. Оптимизация роя частиц на основе отображения ресурса задачи может обеспечить трехкратную экономию затрат в отличие от BRS (Лучший выбор ресурсов) на основе сопоставления для приложения рабочего процесса. Кроме того, оптимизация роя частиц уравновешивает нагрузку на вычислительные ресурсы, распределяя задачи по возможным ресурсам. Таким образом, можно сделать вывод, что планирование задач и управление ресурсами должны быть тщательно проанализированы и оптимизированы в облачной среде для достижения снижения затрат и времени, что приводит к повышению качества и надежности.

1. Сопутствующие работы

Технология виртуализации обычно используется для виртуализации отдельных серверов в различные серверы, что не только создает операционную среду для платформы облачных вычислений на базе виртуальных машин, но и существенно повышает ее эффективность.

В [16, 17] исследовано QoS (качество обслуживания) на основе GHPSO (генетическая гибридная частица Swarm Optimization) для планирования приложений облачных активов.В генетическом гибридном рое частиц оптимизация мутации и гибрида наследственного алгоритма вставляется в алгоритм PSO. Результаты моделирования показывают, что генетическая гибридная оптимизация роя частиц обеспечивает более высокую производительность по сравнению со стандартным алгоритмом PSO, используемым при предельных затратах в течение заданного времени выполнения.

В [18, 19] предложен улучшенный алгоритм для выполнения субоптимизации или оптимизации планирования облака. MGA (улучшенный генетический алгоритм)

используется для автоматизированного подхода к бронированию. Тесты подтверждают, что скорость MGA почти вдвое превышает традиционную.

Генетический алгоритм часто лучше формирует стратегии планирования и использования стоимости активов, чем инфраструктура с открытым исходным кодом в качестве облачных фреймворков сервиса. В [20] усовершенствован алгоритм планирования, основанный на затратах, для эффективного представления задач возможным исполнителям в облаке. Этот алгоритм планирования измеряет как производительность вычислений, так и стоимость активов, а также повышает соотношение коммуникации и вычислений, собирая клиентские задачи в соответствии с возможностями обработки конкретного облачного актива и отправляя собранные задания в актив.

В [21] предложена оптимизация роя частиц для построения алгоритма решения проблемы балансировки нагрузки в виртуальной среде. Целью данного исследования является ограничение времени выполнения занятий. В [22, 23] оптимизация роя частиц предлагается для уменьшения среднего времени выполнения работ и повышения коэффициента доступности активов.

В [24, 25] новый адаптивный алгоритм управления заданиями предлагается использовать для расположения экземпляров виртуальных машин на расположении облачных физических активов и уменьшать деградацию системы.

В [26] предложен планировщик для регистрации размещения виртуальной машины в соответствии с текущей нагрузкой физических активов и ограничения используемых ресурсов. Такое ограничение может привести к массовому снижению затрат. Кроме того, чем больше потребляемая мощность, тем больше рассеивается тепло, и таким образом возрастает вероятность аппаратных сбоев.

Оптимизация роя частиц используется для формирования групповых и глобальных задач оптимизации. Динамически варьируемая оптимизация роя частиц инерционного веса быстро создает и удовлетворяет почти оптимальным решениям. Преимущества оптимизации роя частиц заключаются в быстрой сборке и большей точности выходных данных. Оптимизация роя частиц лучше всего применима к глобальному поиску.

2. Полученные результаты

Разработан алгоритм PSO, а R-фактор и PSO объединены для эффективного распределения ресурсов и планирования в облачной среде.

2.1. Оптимизация роя частиц

Оптимизация роя частиц состоит из частиц Роя, каждая из которых указывает на решение проблемы. В этом исследовании частицы указывают на группу виртуальных машин, которые выделяются для выполнения задач. Каждая частица в поведении Роя имеет 2 основных символа, а именно позицию x, которая обозначает рекомендуемое место, и скорость v, которая указывает скорость движения. Оптимизация роя частиц использует адаптивное движение, которое преобразует положение частицы на каждой итерации. Шаги, используемые в алгоритме PSO, следующие:

(1) На первом шаге в алгоритме PSO производится инициализация параметров.

MAX_ST = 1000;
PF = 500;
NoT = 10;
PS = 100;
NoI = 40.

ST обозначает время моделирования, а PF – частоту паузы, когда на каждой паузе новый исполнитель должен отправлять облачные пакеты и виртуальные машины в возможный центр обработки данных. Неизвестно количество задач, PS обозначает размер популяции.

(2) Следующим шагом в алгоритме PSO является передача начальных параметров функции полезности.

(3) после этого частицы инициализируются.

(4) Затем Рой инициализируется.

```
MinP (0);
MaxP (this.NoDC – 1);
MaxMinV (0.5);
setPt (this.Pt);
for (integer a = 0; I < NoI; a++)
this.swarm.evolve ();
if (a % 10 == 0) {
GB (Min Cost)
Makespan = this.ff.evaluate (bestPt.getGB)
```

P обозначает позицию, а NoDC – количество центров обработки данных. Pt обозначает частицы, а V – скорость. ff указывает на фитнес-функцию фитнеса, а GB – на глобальный оптимум.

Фитнес-функция:

(1) первым шагом в функции полезности является получение матрицы времени выполнения. Запись [a, b] содержит время выполнения задачи x в центре обработки данных y.

Возвращает матрицу времени выполнения.

(2) второй шаг состоит в том, чтобы получить матрицу времени взаимодействия. Запись [a, b] содержит время передачи задачи x в центр обработки данных y.

Возвращает матрицу времени взаимодействия.

(3) Третий шаг предназначен для текущего количества задач.

```
Makespan = Math.maximum (makespan, DCWT[DCId]);
```

(4) Четвертый шаг – инициализация матриц времени связи и выполнения.

```
For (int a = 0; a < NoT; a++) {
For (int b = 0 ; b < NoDC; b++) {
ExecutionT [a] [b] = Math.random () _Max_ET;
CommunicationT [a] [b] = Math.random() _
Max_CT+20;
```

T – время, a и b – целые числа, ET – время выхода, а CT – время связи.

2.2. R-фактор

Основное назначение класса «R factor» заключается в выделении виртуальной машины для вычисления «R factor». Этот класс будет работать для распределения между двумя центрами данных.

Первый центр обработки данных будет рассматриваться как локальный центр обработки данных, а второй – как облачный / удаленный центр обработки данных. Облачный дата-центр будет использоваться только в том случае, если R-фактор будет выше некоторого порогав соответствии со следующим алгоритмом:

(1) Инициализация параметров

```
R = 2; инициирующее значение по умолчанию
L_Threshold= 1;
```

U_Threshold= 10;

NoH (используется для вычисления R-фактора) = 1;

NoM (используется для вычисления R-фактора) = 1;

L - нижний порог, U – верхний порог, NoH - количество просмотров, NoM – число потерь.

(2) Создание нового объекта APS виртуальной машины, где APS обозначает простую политику приложения.

(3) Выделение хоста для данной виртуальной машины.

Возвращает $true, если хост выделен; $false в противном случае

If виртуальная машина не была создана

Do (до тех пор, пока хост не будет найден или пока все они не будут перебраны)

Int morefree= Int.Min_

(4) Вычисление R-фактора.

R = (double) H/M;

Применить ограничения для выделения **/

If (R > L_Threshold && R < U_Threshold)

// разрешить свободное распределение

} иначе, если (P < L_Threshold) {

// принудительное выделение облака

Ida = 1;

Попытка создания виртуальной машины на определенном хосте

If (result) { //если виртуальная машина была успешно создана

на хосте

Фиксируем успешное значение R-фактора для локальных ресурсов

Результат = True;

Прерывание;

(5) Регистрация значения отказа R-фактора для локальных ресурсов и возврат результатов.

3. Анализ

На Рисунке 2 показана блок-схема оптимизированного алгоритма эффективного распределения ресурсов и планирования в облачной среде.

Сначала инициализируем параметры виртуальной машины:

long size = 10000; // размер изображения (MB)

int ram = 4096; // память ВМ (MB)

int mips = 1000;

long bw = 4000;

int pesNumber = 4; // количество процессоров

String vmm = “Xen”; // имяVMM

Затем создаются виртуальные машины, в которых также создается контейнер для хранения:

long length = 40000;

long fileSize = 300;

long outputSize = 300;

int pesNumber = 1.

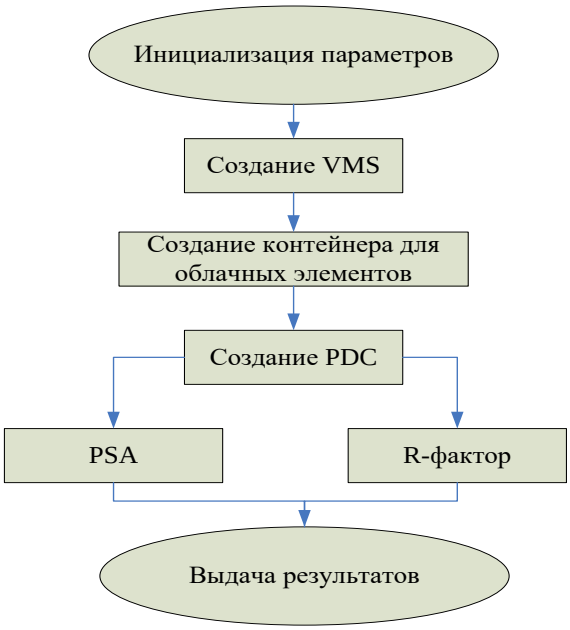

Рисунок 2 – Блок-схема алгоритма
Figure 2 – Block diagram of the algorithm

Вслед за облаками создается центр обработки данных, который включает в себя более одного ядра. Центр обработки данных настроен для использования с распределением виртуальных машин на основе R-фактора и этот алгоритм присутствует внутри имени файла VmAllocationPolicyRFactor.java. Этот класс будет функционировать для распределения между двумя центрами обработки данных. Первый центр обработки данных будет рассматриваться как локальный центр обработки данных, а второй -как удаленный/облачный центр обработки данных.

Облачный дата-центр будет использоваться только в случае, когда R-фактор превышает определенный порог. Кластеры в системе можно разделить на два типа, а именно облачный узел и локальный узел. Облачный узел строится с использованием ресурса облака, и ресурсы имеют отношение к функциональности локального узла. Локальные узлы строятся с использованием локальных ресурсов и являются одной из важных частей системы. Разница между локальным узлом и облачным узлом заключается в количестве узлов в облаке, которое меняется в зависимости от состояния загрузки системы и допустимого количества ресурсов. С точки зрения ресурсного обеспечения система в данном исследовании использует облачные ресурсе и локальные ресурсы. Есть и гибридный режим ресурса.

Часто существуют определенные требования, которых нет в кэше, но они должны быть отправлены на исходный сервер из-за особенностей кэша. Эти запросы называются miss request, и на время их ответа влияет состояние сети и пропускная способность исходного сервера.

Когда есть перегрузка сети или физический сервер перегружен, тогда miss request будут такими же, как если нагрузка в системе уменьшится.

Относительный индекс производительности R представляет собой отношение среднего времени обслуживания запросов к среднему времени отклика исходного запроса на пропуск сервера за данный период времени.

Алгоритм оптимизации роя частиц используется для определения того, какие облака будут распределены на какую виртуальную машину, так что алгоритм пытается снизить стоимость распределения, которая составляет время распределения. На Рисунке

3 показана блок-схема алгоритма PSO.

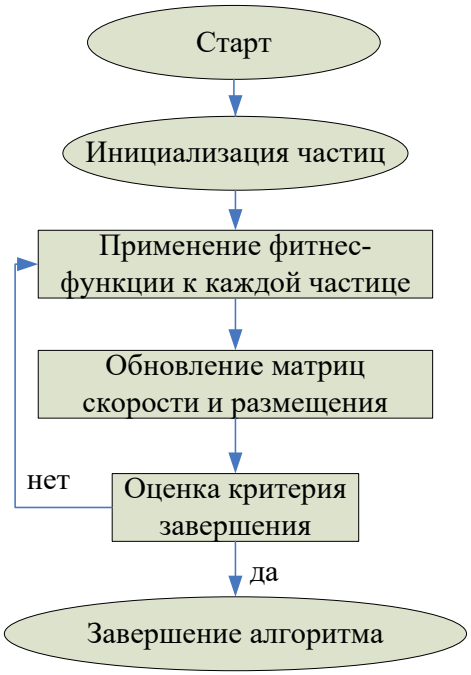

Рисунок 3 – Блок-схема алгоритма PSO
Figure 3 – Block diagram of the PSO algorithm

В PSO итерации используются для нахождения t позиции каждой частицы, которая называется личным лучшим (Pb), достигнутым частицей I, и глобальным лучшим (Pg). Это помогает в прогнозировании наилучшего решения за короткое время вычислений [13]. После того, как все задачи расположены в облачном окружении, алгоритм оптимизации используется для оценки минимальных значений времени ожидания заданий [14]. Эти минимальные значения используются для обеспечения правильного порядка выполнения задач, что, в свою очередь, сокращает общее время ожидания. В нашем исследовании после получения правильного оптимального порядка задач алгоритм, генерирующий очереди, используется для прогнозирования порога, а затем отправляет задачу в эту очередь. Затем расписание планирует задачи на соответствующий ресурс. Основная цель оптимизация роя частиц заключается [15] в выделении запроса пользователя на соответствующий ресурс. В нашем исследовании для эффективного планирования задачи в облачном окружении процесс планирования задач нуждается в оптимальном алгоритме, который учитывает ресурсы и задачу. Алгоритм оптимизации роя частиц учитывает как задачу, так и ресурс и помогает сохранить ресурс занятым и сократить время обработки задач.

Для анализа было отобрано 500 задач с 20 виртуальными машинами для объединения R-фактора и частиц Роя. Первая задача находится между задачами и виртуальными машинами. При выполнении задач с использованием десяти виртуальных машин время отклика виртуальных машин становится стабильным через 40 с. При использовании 20 виртуальных машин среднее время отклика уменьшается до 20 с.

Заключение

В работе исследуется проблема планирования и управления ресурсами. Алгоритмы оптимизации роя частиц являются наиболее известными алгоритмами планирования задач в распределенных системах. Для повышения производительности алгоритма оптимизации роя частиц рекомендуется использовать модифицированный

алгоритм оптимизации роя частиц, в котором он объединяется для генерации начальной популяции. Эксперименты показывают, что как R-фактор, так и алгоритм PSO показывают правильные результаты. Этот алгоритм может быть использован в среде облачных вычислений для эффективного планирования задач на уже существующих ресурсах, чтобы сократить время выполнения задач. В будущих работах планируется сосредоточиться на развитии процесса разделения приложений или больших задач на небольшие подзадачи, разрабатывая процесс для большей скорости и точного распределения в зависимости от длины задачи. Это приведет к повышению точности и скорости распределения ресурсов и планирования задач, а тем самым и к повышению эффективности облачных вычислительных систем.

ЛИТЕРАТУРА/ REFERENCES

1. Buyya R., Yeo C.S., Venugopal S., Broberg J., Brandic I. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems.* 2009;25(6):599-616.
2. Hayes B. Cloud computing. *Communications of the ACM.* 2008;51(7):9-11.
3. Kennedy J., Eberhart R. Particle swarms optimization. *IEEE International Conference on Neural Networks.* 1995;4:1942-1948.
4. Pandey S., Wu L., Guru S.M., Buyya R. A Particle Swarm Optimization-Based Heuristic for Scheduling Workflow Applications in Cloud Computing Environments. *2010 24th IEEE international Conference on Advanced Information Networking and Applications (AINA),* IEEE. 2010:400-407.
5. Liu P. Cloud computing definition and characteristics. *China cloud computing. 2009.* http://www.chinacloud.cn. 2009;2(25).
6. Kumar P., Verma A. Independent task scheduling in cloud computing by improved genetic algorithm. *International Journal of Advanced Research in Computer Science and Software Engineering.* 2012;2(5).
7. Roy P., Mejbah M., Das N. Heuristic based task scheduling in multiprocessor systems with genetic algorithm by choosing the eligible processor. *International Journal of Distributed and Parallel Systems (IJDPS).* 2012;3(4).
8. Chalack S.A., Razavi S.N., Harounabadi A. Job scheduling on the grid environment using max-min firefly algorithm. *International Journal of Computer Applications Technology and Research.* 2014;3(1):63-67.
9. Vinothina V., Sridaran R., Ganapathi P. A survey on resource allocation strategies in cloud computing. *International Journal of Advanced Computer Science and Applications,* 2012;3(6):97-104.
10. Alkayal E.S., Jennings N.R., Abulkhair M.F. Efficient Task Scheduling Multi-Objective Particle Swarm Optimization in Cloud Computing. *2016 IEEE 41st Conference on Local Computer Networks Workshops (LCN Workshops).* November; IEEE. 2016:17-24.
11. Buyya A.R.,Nath B. Nature's Heuristics for Scheduling Jobs on Computational Grids. *8th IEEE International Conference on Advanced Computing and Communications (ADCOM 2000),* India. 2000.
12. Zhang L., Chen Y., Yang B. Task Scheduling Based on PSO Algorithm in Computational Grid. *2006 Proceedings of the 6th International Conference on Intelligent Systems Design and Applications,* Jinan, China. 2006.
13. Kalra M., Singh S. A review of metaheuristic scheduling techniques in cloud computing. *Egyptian Informatics Journa.* 2015;16(3):275-295.

14. Verma A., Kaushal S. Bi-criteria priority based particle swarm optimization workflow scheduling algorithm for cloud. *2014 Recent Advances in Engineering and Computational Sciences (RAECS)*. IEEE. 2014:1-6.
15. Beegom A.A., Rajasree M.S. A Particle Swarm Optimization Based Pareto Optimal Task Scheduling in Cloud Computing. *International Conference in Swarm Intelligence*, Cham, Springer. 2014:79-86.
16. Jun Xue S., Wu W. Scheduling workflow in cloud computing based on hybrid particle swarm algorithm. *Indonesian Journal of Electrical Engineering*. 2012;10(7):1560-1566.
17. Guo L., Zhao S., Shen S., Jiang C. Task scheduling optimization in cloud computing based on heuristic algorithm. *Journal of Networks.* 2012;(7):547-553.
18. Varalakshmi P., Ramaswamy A., Balasubramanian A., Vijaykumar P. An optimal workflow based scheduling and resource allocation in cloud. *Advances in Computing and Communications, First International Conference, ACC.* 2011;411-420.
19. Zhong H., Tao K., Zhang X. An Approach to Optimized Resource Scheduling Algorithm for Open-Source Cloud Systems. *Fifth Annual China Grid Conference*. 2010.
20. Selvarani S., Sadhasivam G. Improved Cost-Based Algorithm for Task Scheduling in Cloud Computing. *IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*. 2010.
21. Liu Z., Wang X. A pso-based algorithm for load balancing in virtual machines of cloud computing environment. *Advances in Swarm Intelligence*, ser. *Lecture Notes in Computer Science*, Berlin, Heidelberg, Springer. 2012;7331:142-147.
22. Zhan S., Huo H. Improved PSO-based task scheduling algorithm in cloud computing. *Journal of Information and Computational Science.* 2012;9(13):3821-3829.
23. Liu J., Guo Luo X., Zhang X.M.F. Job scheduling algorithm for cloud computing based on particle swarm optimization. *Advanced Materials Research.* 2013;662:957-960.
24. Jeyarani R., Nagaveni N., Vasanth Ram R. Design and implementation of adaptive power-aware virtual machine provisioner (APA-VMP) using swarm intelligence. *Future Generation Computer Systems.* 2012;28(5):811-821.
25. Liu Y., Zhu H. A survey of the research on power management techniques for high-performance systems. *Software Practice and Experience.* 2010;40(11):943-964.
26. Feller E., Rilling L., Morin C. Energy-Aware Ant Colony Based Workload Placement in Clouds. *12th International Conference on Grid Computing, ser. Grid*, IEEE Computer Society. 2011;11:26-33.

ИНФОРМАЦИЯ ОБ АВТОРЕ / INFORMATIONS ABOUT AUTHORS

Спицын Андрей Алексеевич, Военный учебно-научный центр Военно-воздушных сил «Военно-воздушная академия имени профессора Н.Е. Жуковского и Ю.А. Гагарина», Воронеж, Российская Федерация.
e-mail: an.spitsin@yandex.ru

Spitsin Andrey Alekseevich, Military training and research center of the Air force «Air Force Academy named after Professor N.E. Zhukovsky and Y.A. Gagarin», Voronezh, Russian Federation.

Мутин Денис Игоревич, д. т. н., Московский государственный технологический университет Станкин, Москва, Российская Федерация.
e-mail: 5015646@mail.ru

Mutin Denis Igorevich, doctor of technical Sciences, Moscow state technological University Stankin, Moscow, Russian Federation.