Имплементация и сравнительное тестирование алгоритма проблемно-ориентированного планирования потоковых приложений в облачных средах PO-HEFT*

Г.И. Радченко, И.А. Лыжин, Е.А. Неповинных

Южно-Уральский государственный университет

Специфика современных вычислительных экспериментов подразумевает, что ресурсы облачной вычислительной среды часто используются для решения большого числа задач, отличающихся лишь значениями относительно небольшого набора параметров моделирования. Такие наборы задач могут появляться, например, при реализации многовариантных расчетов, ориентированных на поиск значений параметров, обеспечивающих оптимизацию определенных характеристик вычислительной модели. Приложения такого рода составляют большой процент загрузки современных суперкомпьютерных и распределенных вычислительных систем, что влечет необходимость в создании методов и алгоритмов эффективного распределения ресурсов таких систем для оптимизации решения таких задач. Целью данной работы является реализация проблемно-ориентированного алгоритма планирования потоков работ для поддержки научных расчетов (Scientific Workflow) PO-HEFT и сравнение его с другими алгоритмами планирования потоков работ.

Ключевые слова: проблемно-ориентированное планирование, облака, облачные вычисления, моделирование, алгоритм планирования, HEFT, PO-HEFT

1. Введение

Последние тридцать лет наблюдается революционный поворот в основах научной деятельности и инжиниринга. Вычислительные методы стали “третьей ветвью” научного подхода, наряду с теорией и экспериментом. Вычислительные методы применяются в приложениях, которые связаны с анализом данных, визуализацией результатов экспериментов [8]. Использование методов суперкомпьютерного моделирования и интеллектуального анализа данных обеспечивает получение качественно новых результатов во всех отраслях знаний, позволяя проводить численные исследования физических, биологических, социальных и др. процессов, предоставляя реальную альтернативу дорогостоящим (или невозможным) экспериментам [6, 11].

Типовой сценарий вычислительного эксперимента – это повторяющийся цикл, состоящий из

1) передачи данных на суперкомпьютер для анализа или моделирования;
2) проведения вычислений;
3) управления хранением полученных результатов.

Таким образом, такой типовой сценарий может быть успешно реализован посредством так называемых «потоков работ». В работе [9] предлагается следующее определение: *поток работ* — это автоматизация процессов, заключающаяся в оркестрации набора сервисов, агентов или акторов, возможности которых должны быть объединены для решения определенной задачи или для определения нового сервиса. Наиболее общим вариантом представления потока работ является ориентированный граф, в котором узлы соответствуют действиям по обработке данных, а ребра представляют собой зависимости по данным [26]. Но чаще всего, потоки работ представляются в виде ациклического ориентированного графа (Directed Acyclic Graph – DAG) [37] или даже в виде последовательностей (конвейера) действий. Циклические ориентированные графы (Directed Cyclic Graph – DCG) значительно более трудны в реализации, поскольку требуется обеспечить поддержку итерационных вычислительных процессов [36].

* Данная работа выполнялась при поддержке РФФИ (грант № 14-07-00420) и Гранта Президента РФ (№ МК-7524.2015.9).

При этом, специфика реализации вычислительных экспериментов часто подразумевает, что ресурсы распределенной вычислительной среды часто используются для решения большого числа задач из узкой предметной области, отличающихся лишь значениями относительно небольшого набора параметров моделирования.

Для повышения эффективности использования вычислительных ресурсов создаются проблемно-ориентированные распределенные вычислительные среды, обеспечивающие решение задач в рамках конкретных предметных областей (математика, вычислительная гидродинамика, вычислительная химия, вычислительная инженерия и др.) [1, 15, 24]. Такое ограничение позволяет использовать информацию о предметной области для прогнозирования вычислительных характеристик задач при планировании и распределении заданий, увеличивая эффективность использования доступных вычислительных ресурсов [21].

В статье [21] предлагается подход к реализации проблемно-ориентированного планирования вычислительных ресурсов на основе алгоритма PO-HEFT. Целью данной работы является тестовая реализации алгоритма PO-HEFT и анализ его эффективности по сравнению с существующими алгоритмами планирования потоков работ.

Статья организована следующим образом. В разделе 2 представлен обзор существующих подходов к планированию и прогнозированию ресурсов выполнения потоков работ. Также, нами представляется анализ существующих платформ, обеспечивающих возможность моделирования распределенных вычислительных сред, включая облачные системы и системы потоков работ. В разделе 3 представлена реализация алгоритма планирования PO-HEFT. Далее, в разделе 4 представлена архитектура используемой системы моделирования проблемно-ориентированной среды. Раздел 5 посвящен реализации и сравнительному тестированию алгоритмов проблемно-ориентированного планирования потоков работ. В заключении представлены краткие выводы по проделанной работе и возможные направления развития исследования.

2. Обзор методов планирования и прогнозирования потоков работ

Алгоритмы, связанные с управлением потоками работ, можно разбить на 2 категории: алгоритмы планирования и алгоритмы прогнозирования. Алгоритмы планирования отвечают за распределение задач между доступными вычислительными ресурсами. Алгоритмы прогнозирования позволяют оценить время, которое потребуется для выполнения отдельных задач и всего потока в целом. Алгоритмы планирования могут использовать алгоритмы прогнозирования для более оптимального распределения задач между ресурсами.

2.1. Алгоритмы планирования независимых задач

В [12] проводится сравнение алгоритмов планирования для независимых задач, включая такие алгоритмы как *Opportunistic Load Balancing (OLB), Minimum Execution Time (MET), Minimum Completion Time (MCT), The Min-Min Heuristic, The Max-Min Heuristic* и др. Однако для планирования потоков работ необходимо учитывать связи по управлению и по данным между задачами, составляющими поток.

Рассмотрим основные алгоритмы планирования потоков работ. Алгоритм *Heterogeneous Earliest Finish Time (HEFT)* [30] является одним из наиболее распространенных алгоритмов планирования потоков работ на сегодняшний день. Рассмотрим работу алгоритма. Пусть $|T_x|$ — размер задачи T_x, а R - множество вычислительных ресурсов со средней вычислительной мощностью $|R| = \sum_{i=1}^{n} |R_i|/n$. Тогда среднее время выполнения задачи на всех доступных ресурсах рассчитывается, как:

$$E(T_x) = \frac{|T_x|}{|R|},$$

Пусть $\bar{T}_{xy}$ – объем пересылаемых данных между задачами T_x и T_y, а R - множество доступных ресурсов со средней мощностью пересылки данных $\bar{R} = \sum_{i=1}^{n} \bar{R}_i/n$. Тогда усредненная

оценка затрат на пересылку данных между задачами T_x и T_y для всех пар ресурсов:

$$D(T_{xy}) = \frac{\bar{T}_{xy}}{\bar{R}},$$

В этом случае, приоритет вычислительного блока может быть определен, как:

$$rank(T_x) = E(T_x) + \max_{T_y \in succ(T_x)} (D(T_{xy}) + rank(T_y)),$$

где $succ(T_x)$ – множество всех задач, которые зависят от задачи T_x.

Таким образом, приоритет задачи напрямую определяется приоритетом всех зависимых от нее задач. Назначение задач на ресурсы происходит следующим образом: задача с большим приоритетом при условии выполнения всех задач, от которых она зависит, назначается на вычислительный ресурс, обеспечивающий меньшее время выполнения задачи.

Ряд коллективов предложил модификации алгоритма HEFT, ориентированные на решение определенных проблем при планировании потоков работ. Так, *PDHEFT* [16] позволяет избежать затраты на передачу данных между узлами путем дублирования задач; *PO-HEFT* [21] обеспечивает оценку вычислительных характеристик задачи на основе ее типа и значений входных параметров, прогнозируя время выполнения задачи, пределы масштабирования и количество передаваемых данных; *Dynamic Data-Resource-Selection (DDRS)* [22] обеспечивает динамический перерасчет приоритетов задач через определенные интервалы.

Можно также отметить ряд алгоритмов планирования, не являющихся развитием HEFT:

- *Problem Oriented Scheduling (POS)* [27] – обеспечивает планирование ресурсов в распределенных проблемно-ориентированных вычислительных средах. Отличительной особенностью алгоритма POS является то, что при планировании ресурсов он учитывает знания о специфике предметной области, позволяя обеспечить запуск одной задачи на нескольких процессорных ядрах с учетом ограничений по масштабируемости данной задачи.
- *Dominant Sequence Clustering (DSC)* [35] – состоит их двух фаз. На первой фазе производится поиск критического пути в графе и происходит кластеризация задач вдоль этого пути. Задачи одного кластера выполняются на одном узле. Затем первая фаза рекурсивно применяется к оставшимся задачам. На второй фазе происходит слияние некоторых кластеров для уменьшения количества используемых ресурсов.
- *Critical Path First (CPF)* [17]– в графе потока работ находит критический путь (путь с максимальными временными затратами). Задачи, расположенные на найденном критическом пути, определяет на один вычислительный узел, чтобы избежать дополнительных затрат на коммуникацию. Остальные задачи распределяет по остальным узлам, стараясь не увеличить общее время выполнения потока.

2.2. Алгоритмы прогнозирования вычислительных характеристик задач и потоков работ

Анализ основных направлений исследований в области планирования потоковых приложений в распределенных вычислительных средах показывает, что одной из наиболее актуальных задач сегодня является задача проблемно-ориентированного планирования и прогнозирования нагрузки вычислительной среды [14, 28, 29]. Это связано с тем, что основной проблемой всех указанных выше алгоритмов является сложность получения достоверной информации о вычислительных характеристиках задачи в потоке работ до того, как задача была завершена. Поэтому совместно с алгоритмами планирования применяются различные техники и алгоритмы прогнозирования характеристик вычислительных задач. В качестве объекта прогнозирования могут выступать такие характеристики вычислительного процесса, как время выполнения задачи или потока, количество ресурсов, которое потребуется для выполнения задачи, объем входных и выходных данных, степень масштабирования задачи и другие параметры.

Существующие подходы и техники оценивания времени выполнения задачи можно разделить на три категории: статические, динамические и гибридные. В статических подходах оценка всего потока и отдельных его задач производится до его непосредственного выполнения. В динамических подходах оценка времени выполнения задачи производится непосредственно перед ее выполнением с учетом текущего состояния системы и доступных ресурсов. Однако,

наилучшим подходом следует считать гибридный, который совмещает в себе преимущества статического и динамического.

В статье [26] вводится понятие MAPE-K цикла (анг. «Monitoring, Analysis, Planning, Execution and Knowledge» - «Мониторинг, Анализ, Планирование, Исполнение и Знание»), связанного с динамическим планированием. Изначально оценивается весь поток и строится начальный план его выполнения. По завершении каждой задачи оцененные значения заменяются реальными, производится переоценка потока и, возможно, перепланировка. Это позволяет сразу исправлять ошибки оценки и не распространять их на весь поток.

В статье [33] для оценки времени выполнения задачи сначала строится регрессионная модель, показывающая зависимость количества операций CPU от размера входных данных. Затем эта модель в комбинации со статическими (частота CPU) и динамическими (например, объем свободной оперативной памяти) параметрами позволяет оценить количество времени, которое потребуется для выполнения задачи на узле.

2.3. Моделирование потоков работ

Системы управления потоками работ для поддержки научных расчетов (Scientific Workflow Management System - SWMS) – это системы, поддерживающие разработку, развертывание и анализ результатов исполнения потоков работ. Они обеспечивают автоматизацию цикла вычислительного эксперимента, упрощая исследователям и инженерам использование распределенных вычислительных ресурсов, и позволяя им сфокусироваться на решении прикладных задач, а не на вопросах управления вычислительным процессом [8]. SWMS позволяет определить абстрактный поток работ, сформировать план исполнения потока работ (Workflow Execution Plan – WEP), с учетом возможностей оптимизации и параллельного выполнения отдельных действий и подпотоков, обеспечить управление источниками данных, отправить поток работ на исполнение и получить результаты расчета. Можно выделить ряд платформ, наиболее используемых научными группами в области управления потоками работ, такие системы как Pegasus [7, 10], Taverna [20, 32], Askalon [31], Kepler [18], Triana [25] и др.

Для моделирования процесса работы алгоритмов управления потоками работы могут быть применены системы моделирования. Одним из самых распространенных систем для моделирования платформ управления потоками работ является система WorkflowSim [5], которая является расширением симулятора облачной среды CloudSim [4].

CloudSim – это инструмент для симуляции облачных вычислительных сред. Он поддерживает моделирование таких компонентов облачных вычислительных сред, как центры обработки данных, виртуальные машины и политики распределения ресурсов. Платформа CloudSim является хорошо расширяемой, в связи с чем был реализован ряд независимых проектов, таких как CloudSimEx [34], WorkflowSim [5], Cloud2Sim [13], DynamicCloudSim [3] и др., ориентированных на улучшение производительности либо расширение функциональных возможностей системы.

3. Реализация алгоритма PO-HEFT

Нами предлагается списочный алгоритм проблемно-ориентированного планирования приложений в облачных средах с учетом их вычислительных профилей PO-HEFT [21]. Предложенный алгоритм основан на алгоритме Heterogeneous Earliest-Finish-Time (HEFT), но содержит модификации при вычислении уровня узла задачи и учитывает входящую коммуникационную стоимость его родительских задач.

Облачная система исполнения потоков работ в самом общем виде состоит из вычислительного кластера, облачной платформы, развернутой на этом вычислительном кластере и работающей по принципу Infrastructure as a Service (IaaS) и платформы управления потоками работ, обеспечивающей запуск задач в определенном порядке на вычислительном кластере. Будем считать, что в рамках проблемно-ориентированной облачной вычислительной среды также определено [21, 23]:

- $\mathcal{F}$ – множество всех функций f, которые реализуются в предметной области проблемно-ориентированной облачной вычислительной среды;
- $\mathfrak{M}$ – множество образов виртуальных вычислительных машин $\mathfrak{m}$, доступных для развертывания на узлы;
- $\pi: \mathfrak{m} \to \mathbb{Z}_{>0}$ – характеристика производительности образа виртуальной машины;
- $\hat{\tau}(f, \mathcal{I}^{in}, \pi)$ – оператор оценки ожидаемого времени выполнения функции f при заданном множестве входных информационных объектов $\mathcal{I}^{in}$ на машине, с характеристикой производительности π;
- $\hat{v}(f, \mathcal{I}^{in})$ – оператор оценки общий размер в байтах всех выходных информационных объектов при выполнении функции f при значении входных параметров $\mathcal{I}^{in}$.

Тогда отдельная задача T_x – это ничто иное как отдельный экземпляр функции $f_x \in \mathcal{F}$ с определенным набором входных информационных объектов $\mathcal{I}^{in}$:

$$T_x = f_x(\mathcal{I}_x^{in})$$

Определим R – как множество доступных для развертывания виртуальных машин со средней производительностью мощностью

$$|R| = \sum_{i=1}^{n} \frac{|R_i|}{n} = \sum_{i=1}^{n} \frac{\pi_i}{n}.$$

В этом случае, для оценки времени выполнения $E(T_x)$ можно применить оператор оценки ожидаемого времени выполнения функции:

$$E(T_x) = \hat{\tau}(f_x, \mathcal{I}_x^{in}, |R|). \quad (1)$$

В рамках модели проблемно-ориентированных сервисов необходимо учитывать объем данных, возвращаемых каждой задачей T_x. Для этого может быть использован оператор оценки ожидаемого выхода $\hat{v}(f, \mathcal{I}^{in})$. Следовательно, в рамках проблемно-ориентированной модели для оценки времени передачи данных между двумя задачами может быть использована формула:

$$D(T_{xy}) = \hat{v}(f_x, \mathcal{I}_x^{in}) * \overline{R}_{xy}, \quad (2)$$

Где $\overline{R}_{xy}$ представляет собой пропускную способность канала передачи данных в облачной вычислительной системе.

Таким образом, приоритет вычислительного блока может быть определен, как:

$$rank(T_x) = E(T_x) + \max_{T_y \in succ(T_x)} (D(T_{xy}) + rank(T_y)) \quad (3)$$

где $succ(T_x)$ – множество всех задач, которые зависят от задачи T_x.

На рисунке 1 приведен псевдокод алгоритма PO-HEFT.

```
ПРОЦЕДУРА: PO-HEFT
ВХОДНЫЕ ДАННЫЕ: ГрафЗадач G(T, E), значения входных параметров I^in, СписокРаспре-
деленийЗадач, МножествоРесурсов R
НАЧАЛО
    для каждого t ∈ T из графа задач G
        Оценить время выполнения задачи в соответствии с операцией (1)
    для каждого e ∈ E из графа задач G
        Оценить время передачи данных в соответствии с (2)
    Запустить обход в ширину в обратной последовательности задач и рассчитать
ранг для каждой задачи в соответствии с (3)
    пока в T есть незавершенные задачи
        СписокЗадач получить завершенные задачи из графа задач G
        Назначить Задачу (СписокЗадач, R)
        Обновить СписокРаспределенийЗадач
КОНЕЦ

ПРОЦЕДУРА: НАЗНАЧИТЬ ЗАДАЧУ
ВХОДНЫЕ ДАННЫЕ: СписокЗадач, Множество Ресурсов R
НАЧАЛО
     Отсортировать СписокЗадач в порядке убывания значения ранга задачи
     для каждого t ∈ СписокЗадач
            r ←получить ресурс из R, способный завершить задачу t раньше
           распределить задачу t на ресурс r
           обновить статус r
КОНЕЦ
```

Рис. 1. Алгоритм планирования PO-HEFT потоковых приложений в проблемно-ориентированных вычислительных средах

Для реализации операторов $\hat{\tau}$ и $\hat{v}$ применим эвристический метод прогнозирования, основанный на анализе результатов предыдущих запусков аналогичных задач на основе алгоритма *k-ближайших соседей:*

Шаг 1. Выбрать из базы данных информацию о всех предыдущих запусках функции f.

Шаг 2. Для каждого предыдущего запуска вычислить расстояние относительно прогнозируемого запуска по значениям входных параметров. Для расчета расстояния между значениями входных параметров будем применять стандартное Евклидово расстояние:

$$\rho\left(\mathcal{I}^{in}, \mathcal{I}_j^{in}\right) = \sqrt{\sum_{i=0}^{n}\left(I_i^{in} - I_{j_i}^{in}\right)},$$

где I_i^{in} и $I_{j_i}^{in}$ – размеры i-го параметра для текущего и j-го запуска соответственно.

Шаг 3. Выбрать k предыдущих запусков, расстояние параметров которых ближе всего к текущему значению параметров запуска

Шаг 4. Вычислить среднее время выполнения и средние размеры выходных файлов для выбранных k предыдущих запусков.

4. Реализация испытательного стенда

4.1 Архитектура системы исполнения потоков работ

Для реализации эксперимента по анализу эффективности работы алгоритмов планирования, была определена структура системы исполнения потоков работ, состоящая из следующих компонентов: парсер DAX, подсистема прогнозирования, планировщик, workflow engine, база данных, хранящая статистику с предыдущих запусков. Кроме того, на каждой виртуальной машине на вычислительном кластере запущен профилировщик, который собирает всю необходимую статистику и сохраняет ее в базу данных. Детальная схема всей системы представлена на рисунке 2. В рамках данной работы реализуется подсистема прогнозирования, которая затем используется для реализации алгоритма PO-HEFT.

Разберем более подробно отдельные компоненты, составляющие разрабатываемую программную систему.

1. *Парсер DAX* – компонент, принимающий файл XML, содержащий информацию о задачах, входящих в поток, и их зависимостях, и возвращающий список задач с зависимостями в некотором внутреннем представлении, с которым работают остальные компоненты системы.
2. *База предыдущих запусков* – база данных, содержащая статистику о предыдущих запусках отдельных задач. Статистика по задаче включает в себе параметры запуска, размер выходных данных и время исполнения.
3. *Подсистема прогнозирования* –система для оценки время выполнения и размера выходных данных при конкретных параметрах запуска на основе информации из базы предыдущих запусков.
4. *Планировщик* – компонент, принимающий на вход список задач с зависимостями и временными оценками. Запрашивает у облачной платформы информацию о доступных ресурсах и возвращает план выполнения потока работ.
5. *Workflow Engine* – компонент, действует по предоставленному плану выполнения потока работ и отправляет задачи на выполнение только при условии, что все родительские задачи уже завершены.
6. *Профилировщик* – компонент, работающий на каждой виртуальной машине и собирающий статистику о выполнении задач.

Нами разработаны подсистема прогнозирования и база значений предыдущих запусков. В качестве системы базы данных использовалась SQL база данных PostgreSQL. Подсистема прогнозирования написана на языке программирования Java. Остальные компоненты испытательного стенда для сравнительного анализа алгоритмов планирования реализованы на базе симулятора WorkflowSim.

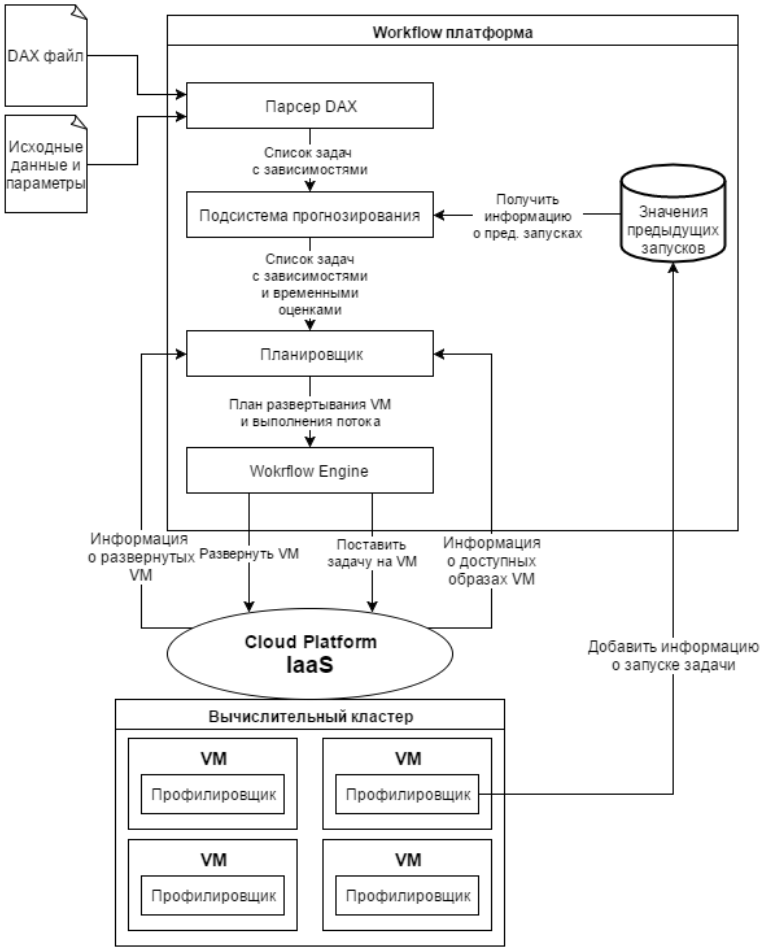

Рис. 2. Архитектура системы исполнения потоков работ

4.2. Модель базы данных истории запусков

Для прогнозирования времени выполнения и объема выходных данных задач, необходимо иметь статистику по предыдущим запускам задач данного типа. Для хранения этой статистики необходима база данных, которая будет наполняться характеристиками запусков задач и использоваться системой прогнозирования. В рамках разработанного испытательного стенда, в базу заносится следующая информация о запусках задач на вычислительном кластере (рис. 3):

- tasks – имена задач, которые могут выполняться в системе;
- executions – статистика о запусках задач, а включая время исполнения задачи, измеряемое в миллисекундах;
- input_sizes – размеры входных файлов;
- output_sizes – размеры выходных файлов.

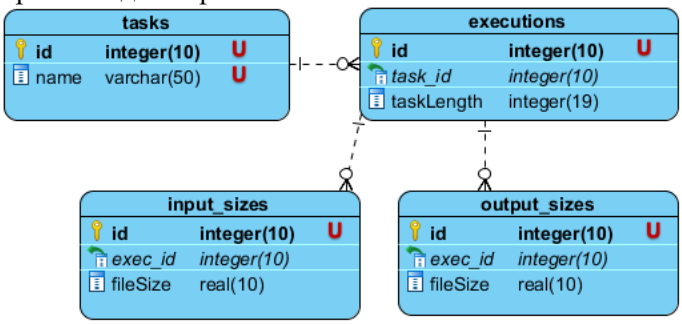

Рис. 3. Схема базы данных истории запусков

5. Тестирование алгоритмов проблемно-ориентированного планирования

5.1. Тестирование алгоритма прогнозирования на отдельных задачах

Для тестирования системы прогнозирования был использован набор из 45 потоков работ проекта Pegasus [2, 19]. Каждый из потоков работ представлен XML-файлом с расширением «dax», что является аббревиатурой от «Directed Acyclic graph Xml». Общее количество задач во всех собранных потоках работ составило около 24 000. Этот набор был разбит на обучающий и тестовый блок. Обучающий блок был загружен в базу предыдущих запусков, на тестовом был запущен алгоритм оценки характеристик задачи, описанный в разделе 3. Значение параметра k было установлено равным 10. Для каждого запуска вычислялась относительная погрешность (истинные значения, равные нулю, пропускались) по формуле:

$$\delta^i = \frac{|T^i_{predicted} - T^i_{real}|}{T^i_{real}}.$$

В результате была вычислена средняя относительная погрешность по всем прогнозируемым запускам. Все собранные данные о задачах разбивались в разном соотношении на обучающие и тестовые блоки, и для каждого разбиения было произведено тестирование на предсказание отдельных задач. Результаты тестирования приведены в таблице 1 и на рис. 4.

Таблица 1. Результаты тестирования алгоритма прогнозирования на отдельных задачах

№ теста	Отношение разбиения «обучающая выборка/тестовая выборка»	Относительная погрешность оценки времени работы (runtime)	Относительная погрешность оценки объема выходных данных (outsize)
1	50/50	6.7%	6.3%
2	60/40	6.2%	5.8%
3	70/30	6.0%	5.6%
4	80/20	5.5%	5.3%
5	90/10	5.0%	4.8%

Как видно из результатов тестирования, точность предсказания времени работы задачи и объема выходных данных на представленной выборке составляет от 7 до 5%, уменьшаясь с увеличением размера обучающей выборки.

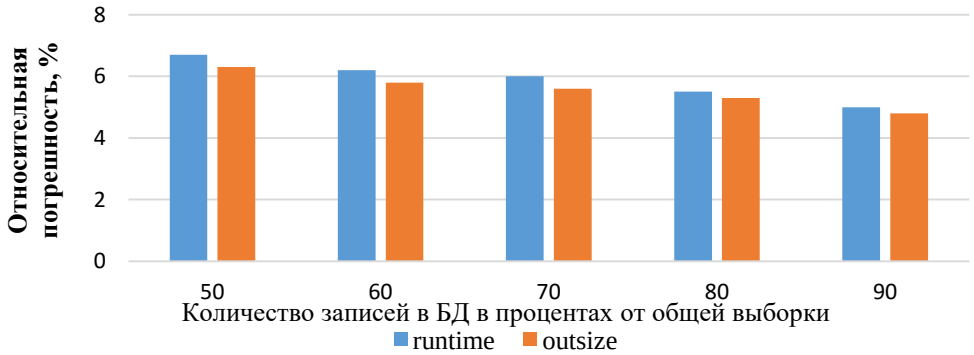

Рис. 4. Результаты тестирования алгоритма прогнозирования на отдельных задачах

5.2. Тестирование алгоритма планирования PO-HEFT на потоках работ

Нами была произведена симуляция исполнения потоков работ на платформе WorkflowSim с использованием следующих алгоритмов:

— *HEFT*, реализованный с использованием точных значений времени выполнения и размера выходных данных, полученных из DAX-файлов;

— *PO-HEFT*, реализующий оценку времени выполнения и объема выходных данных на основе информации, полученной от системы прогнозирования;
— *RANDOM*, обеспечивающий случайное распределение задач по узлам.

Была использована стандартная реализация алгоритмов *HEFT* и *RANDOM* из пакета WorkflowSim. Для тестирования были взяты потоки работ CyberShake (на 30, 50, 100 и 1000 задач), Epigenomics (на 24, 46, 100 и 997 задач) и Inspiral (на 30, 50, 100 и 1000 задач) [19]. Для каждого потока в качестве тестового брался самый большой поток, а меньшие потоки постепенно загружались в базу предыдущих запусков для оценки качества прогнозирования.

В первой серии тестовых запусков, исследование проводилось на наборе из 5 одинаковых виртуальных машин со следующими параметрами:
— MIPS (million instructions per second) = 1000;
— Bandwidth (пропускная способность) = 1000 MB/s.

Для получения консистентных результатов, алгоритм RANDOM запускался 5 раз, после чего в качестве результата определялось среднее значение. Результаты тестирования алгоритмов HEFT, PO-HEFT и RANDOM представлены в таблице 2 и на рисунке 5.

Таблица 2. Сравнение алгоритмов в гомогенной вычислительной среде

Алгоритм / Поток работ	Время исполнения потока работ (в скобках указано относительное замедление по сравнению с HEFT)					
	HEFT	PO-HEFT		RANDOM	
CyberShake_1000	4 754	4 957	(+4.3%)	5 368	(+12.9%)
Epigenomics_997	776 051	789 115	(+1.7%)	885 290	(+14.1%)
Inspiral_1000	45 716	4 6791	(+2.3%)	50 860	(+11.3%)

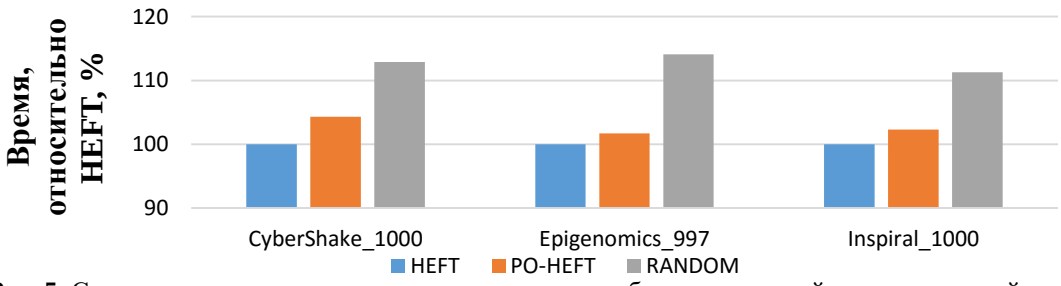

Рис. 5. Сравнение алгоритмов планирования потоков работ в гомогенной вычислительной среде

Также, для оценки влияния значения параметра K, была проведена серия экспериментов с использование алгоритма PO-HEFT и различными значениями параметра K (1, 3, 5, 10) в алгоритме K-ближайших. Результаты тестирования алгоритма PO-HEFT в зависимости от количества информации в базе данных и от параметра K представлены в таблице 3.

Таблица 3. Результаты тестирования алгоритма PO-HEFT на потоках CyberShake_1000, Epigenomics_997, Inspiral_1000

Поток	K=1	K=3	K=5	K=10
CyberShake_1000	4 843	4 843	4 942	5 034
Epigenomics_997	789 076	789 053	789 394	789 087
Inspiral_1000	47 612	46 524	47 301	46 910

Результаты первой серии эксперимента позволяют сделать следующие выводы:
— на анализируемых потоках работ, даже случайное распределение задач (алгоритм RANDOM) по вычислительным узлам в гомогенной вычислительной среде дает относительно хорошие результаты планирования, всего на 10-15% хуже чем алгоритм HEFT;
— результаты алгоритма PO-HEFT, в среднем, показывают результаты на 2-4% хуже чем результаты алгоритма HEFT;
— задачи в анализируемых потоках работ характеризуются сильной прямой зависимостью между объемом входных данных, временем исполнения задачи и объемом выходных дан-

ных. В связи с этим, в среднем, алгоритм PO-HEFT показывает одинаково хорошие результаты при использования значений параметра K=1..10 в алгоритме K-ближайших соседей.

Во второй серии тестирования было создано 5 виртуальных машин с параметрами, представленными в таблице 4.

Таблица 4. Параметры виртуальных машин второй серии экспериментов

	VM_0	VM_1	VM_2	VM_3	VM_4
MIPS	200	400	600	800	1 000
Bandwidth (MB/s)	200	400	600	800	1 000

Для второй серии экспериментов, значения параметра K для алгоритма PO-HEFT было принято равным 10. Усредненные результаты тестирования алгоритмов HEFT, PO-HEFT и RANDOM представлены в таблице 5 и на рисунке 6.

Таблица 5. Сравнение алгоритмов во второй серии тестирования

Алгоритм / Поток работ	Время исполнения потока работ (в скобках указано относительное замедление по сравнению с HEFT)				
	HEFT	PO-HEFT		RANDOM	
CyberShake_1000	7 795	8 106	(+3.8%)	23 716	(+204%)
Epigenomics_997	1 294 702	1 331 352	(+2.8%)	3 995 491	(+209%)
Inspiral_1000	79 130	79 051	(-0.1%)	230 608	(+191%)

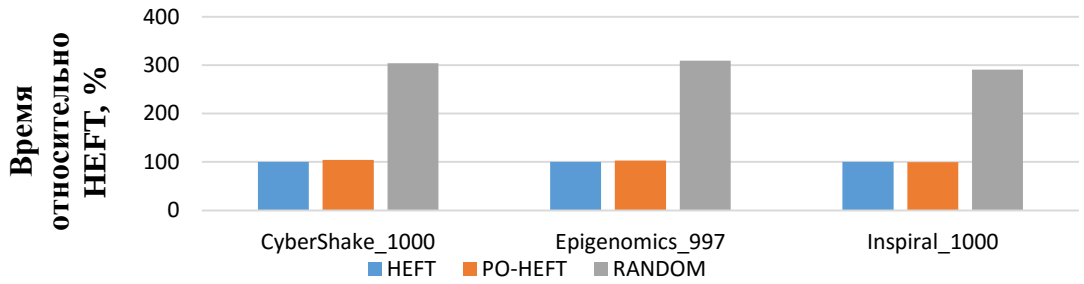

Рис. 6. Сравнение алгоритмов во второй серии тестирования

На основании полученных результатов можно сделать вывод, что разрабатываемый алгоритм PO-HEFT показал незначительно более низкое (1-4%) время выполнения потока работ в сравнение с алгоритмом HEFT. При этом, алгоритм HEFT, в процессе планирования использует априорную информацию о времени выполнения задачи и объеме выходных данных. В то же время эффективность алгоритма PO-HEFT в несколько раз превосходит алгоритм RANDOM в случае, когда в облачной вычислительной среде возможно использование виртуальных машин, обладающих различным уровнем вычислительных возможностей.

6. Заключение

В рамках работы был разработан прототип системы прогнозирования потоков работ в проблемно-ориентированных вычислительных средах и протестирован алгоритм PO-HEFT. В виду отсутствия полноценной статистики запусков реальных задач, прототип был протестирован только на абстрактных потоках работ, для которых известен только размер входного файла, но не известны другие параметры.

В дальнейшем планируется провести исследования по сравнительной эффективности алгоритмов планирования PO-HEFT, POS и DSC на основе информации о параметрах запуска реальных потоков работ, исполняемых на ресурсах суперкомпьютера ЮУрГУ. Но уже сейчас можно сказать, что алгоритм PO-HEFT реализуем и сравним по эффективности планирования с алгоритмом HEFT.

Литература

1. *Afanasiev A., Sukhoroslov O., Voloshinov V.* Mathcloud: Publication and reuse of scientific applications as RESTful web services // Lecture Notes in Computer Science. 2013. Vol. 7979. P. 394–408.
2. *Bharathi S. et al* Characterization of scientific workflows // Third Workshop on Workflows in Support of Large-Scale Science, November 17, 2008, Austin, TX, USA. IEEE Computer Society, 2008. P. 1–10.
3. *Bux M., Leser U.* DynamicCloudSim: Simulating heterogeneity in computational clouds // Future Generation Computer Systems. 2015. Vol. 46. P. 85–99.
4. *Calheiros R. N. et al* 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. P. 23–50.
5. *Chen W., Deelman E.* WorkflowSim: A toolkit for simulating scientific workflows in distributed environments // 8th International Conference on E-Science, October 8-9, 2012, Chicago, Illinois, USA. IEEE Computer Society, 2012. P. 1–8.
6. *Davis P. K., Henninger A. E.* Analysis, Analysis Practices, and Implications for Modeling and Simulation / P.K. Davis, A.E. Henninger, Rand Corporation, 2007. 54 p.
7. *Deelman E. et al* Pegasus, a workflow management system for science automation // Future Generation Computer Systems. 2015. Vol. 46. P. 17–35.
8. *Deelman E. et al* Workflows and e-Science: An overview of workflow system features and capabilities // Future Generation Computer Systems. 2009. Vol. 25, No 5. P. 528–540.
9. *Fox G. C., Gannon D.* Special Issue: Workflow in Grid Systems // Concurrency and Computation: Practice and Experience. 2006. Vol. 18, No 10. P. 1009–1019.
10. *Gil Y., Ratnakar V., Deelman E.* Wings for pegasus: Creating large-scale scientific applications using semantic representations of computational workflows // Proceedings of the National Conference on Artificial Intelligence (Vol. 22, No. 2), July 22–26, 2007, Vancouver, British Columbia, Canada. 2007. P. 1767–1774.
11. *Glotzer S. C.* International assessment of research and development in simulation-based engineering and science / S.C. Glotzer, Imperial College Press, 2011. 312 p.
12. *Kannan R. et al* Managing and Processing Big Data in Cloud Computing / R. Kannan, R.U. Rasool, H. Jin, S.R. Balasundaram, edited by R. Kannan et al, IGI Global, 2016.
13. *Kathiravelu P., Veiga L.* Concurrent and Distributed CloudSim Simulations // IEEE 22nd International Symposium on Modelling, Analysis & Simulation of Computer and Telecommunication Systems, September 9-11, 2014, Paris, France. IEEE Computer Society, 2014. P. 490–493.
14. *Kliazovich D. et al* CA-DAG: Communication-Aware Directed Acyclic Graphs for Modeling Cloud Computing Applications // IEEE 6th International Conference on Cloud Computing, June 27 - July 2, 2013, Santa Clara, CA, USA. IEEE, 2013. P. 277–284.
15. *Knyazkov K. V. et al* CLAVIRE: e-Science infrastructure for data-driven computing // Journal of Computational Science. 2012. Vol. 3, No 6. P. 504–510.
16. *Lee Y. C., Zomaya A. Y.* A Productive Duplication-Based Scheduling Algorithm for Heterogeneous Computing Systems // Proceedings of First International Conference on High Performance Computing and Communications, September 21-23, 2005, Sorrento, Italy. Springer, 2005. P. 203–212.
17. *Lee Y. C., Zomaya A. Y.* Stretch Out and Compact: Workflow Scheduling with Resource Abundance // 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, May 13-16, 2013, Delft, Netherlands. 2013. P. 219–226.
18. *Ludäscher B. et al* Scientific workflow management and the Kepler system // Concurrency and

Computation: Practice and Experience. 2006. Vol. 18, No 10. P. 1039–1065.

19. *Mehta G., Juve G., Chen W.* Workflow Generator. URL: https://confluence.pegasus.isi.edu/display/pegasus/WorkflowGenerator (дата обращения: 11.06.2016).
20. *Missier P. et al* Taverna, reloaded // Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 2010. Vol. 6187. P. 471–481.
21. *Nepovinnykh E. A., Radchenko G. I.* Problem-Oriented Scheduling of Cloud Applications : PO-HEFT Algorithm Case Study // 39th International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2016 - Proceedings, May 30 - June 3, 2016, Opatija, Chroatia. IEEE Computer Society, 2016. P. 196–201.
22. *Pandey S., Buyya R.* Scheduling of scientific workflows on data grids // Proceedings CCGRID 2008 - 8th IEEE International Symposium on Cluster Computing and the Grid, 19-22 May, 2008, Lyon, France. 2008. P. 548–553.
23. *Radchenko G.* Model of problem-oriented cloud computing environment // Proceedings of the Institute for System Programming of the RAS. 2015. Vol. 27, No 6. P. 275–284.
24. *Radchenko G., Hudyakova E.* A service-oriented approach of integration of computer-aided engineering systems in distributed computing environments // UNICORE Summit 2012, Proceedings, May 30-31, 2012, Dresden, Germany. 2012. P. 57–66.
25. *Shiroor A. et al* Scientific workflow management systems and workflow patterns // International Journal of Business Process Integration and Management. 2010. Vol. 5, No 1. P. 63.
26. *Silva R. F. et al* Online Task Resource Consumption Prediction for Scientific Workflows // Parallel Processing Letters. 2015. Vol. 25, No 3. P. 25.
27. *Sokolinsky L. B., Shamakina A. V.* Methods of resource management in problem-oriented computing environment // Programming and Computer Software. 2016. Vol. 42, No 1. P. 17–26.
28. *Tchernykh A. et al* Online Bi-Objective Scheduling for IaaS Clouds Ensuring Quality of Service // Journal of Grid Computing. 2016. Vol. 14, No 1. P. 5–22.
29. *Tchernykh A. et al* Towards Understanding Uncertainty in Cloud Computing Resource Provisioning // Procedia Computer Science. 2015. Vol. 51. P. 1772–1781.
30. *Topcuoglu H., Hariri S., Min-You W.* Performance-effective and low-complexity task scheduling for heterogeneous computing // IEEE Transactions on Parallel and Distributed Systems. 2002. Vol. 13, No 3. P. 260–274.
31. *Wieczorek M., Prodan R., Fahringer T.* Scheduling of scientific workflows in the ASKALON grid environment // ACM SIGMOD Record. 2005. Vol. 34, No 3. P. 56.
32. *Wolstencroft K. et al* The Taverna workflow suite: designing and executing workflows of Web Services on the desktop, web or in the cloud // Nucleic Acids Research. 2013. Vol. 41, No W1. P. W557–W561.
33. *Wu Q., Datla V. V.* On Performance Modeling and Prediction in Support of Scientific Workflow Optimization // 2011 IEEE World Congress on Services, 2011. IEEE, 2011. P. 161–168.
34. *Yang M., Rutherfoord B., Jung E.* Learning Cloud Computing and Security Through Cloudsim Simulation // Information Security Education Journal. 2014. Vol. 1, No 2. P. 62–69.
35. *Yang T., Gerasoulis A.* DSC: Scheduling Parallel Tasks on an Unbounded Number of Processors // IEEE Transactions on Parallel and Distributed Systems. 1994. Vol. 5, No 9. P. 951–967.
36. *Yu J., Buyya R.* A Taxonomy of Workflow Management Systems for Grid Computing // Journal of Grid Computing. 2005. Vol. 3, No 3–4. P. 171–200.
37. *Zhang J. et al* Task mapper and application-aware virtual machine scheduler oriented for parallel computing // Journal of Zhejiang University SCIENCE C. 2012. Vol. 13, No 3. P. 155–177.

Implementation and Testing of the PO-HEFT Problem-oriented Workflow Planning Algorithm for Cloud Environments

G.I. Radchenko, I.A. Lyzhin, E.A. Nepovinnyh

South Ural State University

Modern computational experiments imply that the resources of the cloud computing environment are often used to solve a large number of tasks, which differ only in the values of a relatively small set of simulation parameters. Such sets of tasks may occur while implementing multivariate calculations aimed at finding the simulation parameter values, which optimize certain characteristics of the computational model. Applications of this type make a large percentage of modern HPC systems load, which implies a need for methods and algorithms for efficient allocation of resources in order to optimize systems for solving such problems. The aim of this work is to implement a PO-HEFT problem-oriented scientific workflow planning algorithm and to compare it with other workflow planning algorithms.

Keywords: problem-oriented solving environment, workflow, cloud, simulation, planning algorithm, HEFT, PO-HEFT

References

1. *Afanasiev A., Sukhoroslov O., Voloshinov V.* Mathcloud: Publication and reuse of scientific applications as RESTful web services // Lecture Notes in Computer Science. 2013. Vol. 7979. P. 394–408.
2. *Bharathi S. et al* Characterization of scientific workflows // Third Workshop on Workflows in Support of Large-Scale Science, November 17, 2008, Austin, TX, USA. IEEE Computer Society, 2008. P. 1–10.
3. *Bux M., Leser U.* DynamicCloudSim: Simulating heterogeneity in computational clouds // Future Generation Computer Systems. 2015. Vol. 46. P. 85–99.
4. *Calheiros R. N. et al* 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. P. 23–50.
5. *Chen W., Deelman E.* WorkflowSim: A toolkit for simulating scientific workflows in distributed environments // 8th International Conference on E-Science, October 8-9, 2012, Chicago, Illinois, USA. IEEE Computer Society, 2012. P. 1–8.
6. *Davis P. K., Henninger A. E.* Analysis, Analysis Practices, and Implications for Modeling and Simulation / P.K. Davis, A.E. Henninger, Rand Corporation, 2007. 54 p.
7. *Deelman E. et al* Pegasus, a workflow management system for science automation // Future Generation Computer Systems. 2015. Vol. 46. P. 17–35.
8. *Deelman E. et al* Workflows and e-Science: An overview of workflow system features and capabilities // Future Generation Computer Systems. 2009. Vol. 25, No 5. P. 528–540.
9. *Fox G. C., Gannon D.* Special Issue: Workflow in Grid Systems // Concurrency and Computation: Practice and Experience. 2006. Vol. 18, No 10. P. 1009–1019.
10. *Gil Y., Ratnakar V., Deelman E.* Wings for pegasus: Creating large-scale scientific applications using semantic representations of computational workflows // Proceedings of the National Conference on Artificial Intelligence (Vol. 22, No. 2), July 22–26, 2007, Vancouver, British Columbia, Canada. 2007. P. 1767–1774.

11. *Glotzer S. C.* International assessment of research and development in simulation-based engineering and science / S.C. Glotzer, Imperial College Press, 2011. 312 p.

12. *Kannan R. et al* Managing and Processing Big Data in Cloud Computing / R. Kannan, R.U. Rasool, H. Jin, S.R. Balasundaram, edited by R. Kannan et al, IGI Global, 2016.

13. *Kathiravelu P., Veiga L.* Concurrent and Distributed CloudSim Simulations // IEEE 22nd International Symposium on Modelling, Analysis & Simulation of Computer and Telecommunication Systems, September 9-11, 2014, Paris, France. IEEE Computer Society, 2014. P. 490–493.

14. *Kliazovich D. et al* CA-DAG: Communication-Aware Directed Acyclic Graphs for Modeling Cloud Computing Applications // IEEE 6th International Conference on Cloud Computing, June 27 - July 2, 2013, Santa Clara, CA, USA. IEEE, 2013. P. 277–284.

15. *Knyazkov K. V. et al* CLAVIRE: e-Science infrastructure for data-driven computing // Journal of Computational Science. 2012. Vol. 3, No 6. P. 504–510.

16. *Lee Y. C., Zomaya A. Y.* A Productive Duplication-Based Scheduling Algorithm for Heterogeneous Computing Systems // Proceedings of First International Conference on High Performance Computing and Communications, September 21-23, 2005, Sorrento, Italy. Springer, 2005. P. 203–212.

17. *Lee Y. C., Zomaya A. Y.* Stretch Out and Compact: Workflow Scheduling with Resource Abundance // 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, May 13-16, 2013, Delft, Netherlands. 2013. P. 219–226.

18. *Ludäscher B. et al* Scientific workflow management and the Kepler system // Concurrency and Computation: Practice and Experience. 2006. Vol. 18, No 10. P. 1039–1065.

19. *Mehta G., Juve G., Chen W.* Workflow Generator. URL: https://confluence.pegasus.isi.edu/display/pegasus/WorkflowGenerator (дата обращения: 11.06.2016).

20. *Missier P. et al* Taverna, reloaded // Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 2010. Vol. 6187. P. 471–481.

21. *Nepovinnykh E. A., Radchenko G. I.* Problem-Oriented Scheduling of Cloud Applications : PO-HEFT Algorithm Case Study // 39th International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2016 - Proceedings, May 30 - June 3, 2016, Opatija, Chroatia. IEEE Computer Society, 2016. P. 196–201.

22. *Pandey S., Buyya R.* Scheduling of scientific workflows on data grids // Proceedings CCGRID 2008 - 8th IEEE International Symposium on Cluster Computing and the Grid, 19-22 May, 2008, Lyon, France. 2008. P. 548–553.

23. *Radchenko G.* Model of problem-oriented cloud computing environment // Proceedings of the Institute for System Programming of the RAS. 2015. Vol. 27, No 6. P. 275–284.

24. *Radchenko G., Hudyakova E.* A service-oriented approach of integration of computer-aided engineering systems in distributed computing environments // UNICORE Summit 2012, Proceedings, May 30-31, 2012, Dresden, Germany. 2012. P. 57–66.

25. *Shiroor A. et al* Scientific workflow management systems and workflow patterns // International Journal of Business Process Integration and Management. 2010. Vol. 5, No 1. P. 63.

26. *Silva R. F. et al* Online Task Resource Consumption Prediction for Scientific Workflows // Parallel Processing Letters. 2015. Vol. 25, No 3. P. 25.

27. *Sokolinsky L. B., Shamakina A. V.* Methods of resource management in problem-oriented computing environment // Programming and Computer Software. 2016. Vol. 42, No 1. P. 17–26.

28. *Tchernykh A. et al* Online Bi-Objective Scheduling for IaaS Clouds Ensuring Quality of Service //

Journal of Grid Computing. 2016. Vol. 14, No 1. P. 5–22.

29. *Tchernykh A. et al* Towards Understanding Uncertainty in Cloud Computing Resource Provisioning // Procedia Computer Science. 2015. Vol. 51. P. 1772–1781.
30. *Topcuoglu H., Hariri S., Min-You W.* Performance-effective and low-complexity task scheduling for heterogeneous computing // IEEE Transactions on Parallel and Distributed Systems. 2002. Vol. 13, No 3. P. 260–274.
31. *Wieczorek M., Prodan R., Fahringer T.* Scheduling of scientific workflows in the ASKALON grid environment // ACM SIGMOD Record. 2005. Vol. 34, No 3. P. 56.
32. *Wolstencroft K. et al* The Taverna workflow suite: designing and executing workflows of Web Services on the desktop, web or in the cloud // Nucleic Acids Research. 2013. Vol. 41, No W1. P. W557–W561.
33. *Wu Q., Datla V. V.* On Performance Modeling and Prediction in Support of Scientific Workflow Optimization // 2011 IEEE World Congress on Services, 2011. IEEE, 2011. P. 161–168.
34. *Yang M., Rutherfoord B., Jung E.* Learning Cloud Computing and Security Through Cloudsim Simulation // Information Security Education Journal. 2014. Vol. 1, No 2. P. 62–69.
35. *Yang T., Gerasoulis A.* DSC: Scheduling Parallel Tasks on an Unbounded Number of Processors // IEEE Transactions on Parallel and Distributed Systems. 1994. Vol. 5, No 9. P. 951–967.
36. *Yu J., Buyya R.* A Taxonomy of Workflow Management Systems for Grid Computing // Journal of Grid Computing. 2005. Vol. 3, No 3–4. P. 171–200.
37. *Zhang J. et al* Task mapper and application-aware virtual machine scheduler oriented for parallel computing // Journal of Zhejiang University SCIENCE C. 2012. Vol. 13, No 3. P. 155–177.