Раздел. II. Распределенные и облачные вычисления

УДК 004.4’2+004.89 DOI 10.23683/2311-3103-2018-8-59-69

А.Г. Феоктистов, Р.О. Костромин, И.А. Сидоров, С.А. Горский

МУЛЬТИАГЕНТНЫЙ АЛГОРИТМ ПОСТРОЕНИЯ ОСТАТОЧНОЙ СХЕМЫ РЕШЕНИЯ ЗАДАЧИ В РАСПРЕДЕЛЕННЫХ ПАКЕТАХ ПРИКЛАДНЫХ ПРОГРАММ*

В настоящее время базовые программные средства, реализующие технологии организации расчетов в высокопроизводительных вычислительных системах, обеспечивают потенциальную основу для массового создания и использования параллельных и распределенных приложений. Активно развиваются и применяются на практике инструментальные средства для создания пакетов прикладных программ, а также систем поддержки рабочих процессов (workflow). Однако анализ их практического применения позволяет сделать вывод о необходимости повышения отказоустойчивости процессов решения задач, включающих наборы взаимосвязанных подзадач, в распределенных пакетах прикладных программ. В особенности данная проблема актуализируются при решении задач в гетерогенной распределенной вычислительной среде, в качестве основных компонентов которой выступают кластеры, в том числе гибридные кластеры с разнородными узлами, а высокопроизводительные серверы, системы хранения данных, персональные компьютеры и другие вычислительные элементы дополняют инфраструктуру среды. В статье представлен адаптивный мультиагентный алгоритм, предназначенный для перераспределения заданий по ресурсам такой среды при возобновлении процесса решения задач в распределенных пакетах прикладных программ после отказов программных и аппаратных средств. Работа данного алгоритма базируется в отличие от известных на методах конкретизирующего программирования для построения и выполнения остаточной схемы решения задачи, а также метамониторинга ресурсов среды. Сравнительный анализ результатов эксперимента по полунатурному моделированию поддержки отказоустойчивости процесса выполнения схем решения задач распределенных пакетов прикладных программ различными метапланировщиками продемонстрировал преимущество предложенного подхода к мультиагентному управлению в гетерогенной распределенной вычислительной среде.

Распределенный пакет прикладных программ; схема решения задачи; мультиагентное управление; отказоустойчивость.

A.G. Feoktistov, R.O. Kostromin, I.A. Sidorov, S.A. Gorsky

MULTI-AGENT ALGORITHM FOR CREATING A RESIDUAL PROBLEM-SOLVING SCHEME IN DISTRIBUTED APPLIED SOFTWARE PACKAGES

Nowadays, basic software tools that implement technologies for organizing computations in high-performance computing systems provide a potential basis for the mass creation and use of parallel and distributed applications. Tools for creating applied software packages and workflow support systems are being actively developed and applied in practice. However, an analysis of their practical application allows us to conclude that it is necessary to increase the fault-tolerance of problem-solving processes in distributed applied software packages for problems that include

* Исследование выполнено при поддержке РФФИ, проект № 16-07-00931-а, а также Президиума РАН, программа № 30, проект «Методы, алгоритмы и инструментальные средства децентрализованного группового решения задач в вычислительных и управляющих системах».

sets of interrelated subproblems. In particular, this problem becomes urgent when we solve problems in a heterogeneous distributed computing environment. Clusters, including hybrid clusters with heterogeneous nodes, are the main components of such an environment. High-performance servers, storage systems, personal computers, and other computing elements complement the infrastructure of the environment. The paper presents an adaptive multi-agent algorithm, which is intended for the redistribution of jobs on the resources of such an environment. The algorithm is used when restarting the problem-solving process in distributed applied software packages after the failure of software and hardware. In contrast to the well-known algorithms for maintaining fault-tolerance of distributed computing that are used in workflow management systems, the work of this algorithm is based on the use of program specialization methods for creating and executing a residual problem-solving scheme. It also actively applies meta-monitoring of computational resources. Comparative analysis of the experimental results on the semi-natural modeling the support of the fault-tolerance of the scheme-executing process for solving problems of distributed applied software packages by various meta-schedulers demonstrated the advantage of the proposed approach to multi-agent management in the heterogeneous distributed computing environment.

Distributed applied software package; problem-solving scheme; multi-agent management; fault-tolerance.

Введение. В настоящее время решение сложных задач с помощью суперкомпьютеров обуславливает необходимость обеспечения отказоустойчивости процесса высокопроизводительных вычислений, являющейся чрезвычайно актуальной проблемой [1]. В этой связи целью исследования является повышение отказоустойчивости процессов решения задач, включающих множество взаимосвязанных подзадач, в распределенных пакетах прикладных программ, ориентированных на их применение в гетерогенных вычислительных средах [2].

Схема решения задачи в распределенном пакете прикладных программ представляется абстрактной программой, тесно коррелирующей с понятием рабочего процесса (workflow [3]). В общем случае она описывает процесс решения набора взаимосвязанных подзадач. Как правило, подзадачи характеризуются различными требованиями к ресурсам среды и представляются отдельными вычислительными заданиями, связи между которыми определяются схемой решения задачи.

В системах, поддерживающих управление workflow, используются разные подходы к обеспечению отказоустойчивости: миграция [4–6] и перезапуск процессов [4–9] на уровне отдельных задач на кластере с использованием аппарата контрольных точек, использование альтернативных ресурсов [7–10], а также продолжение выполнение workflow в надежде, что отказавший ресурс, на котором выполнялась одна из подзадач этого workflow, будет восстановлен и продолжит выполнение вычислений [4–7, 10].

В статье предложен новый мультиагентный алгоритм для распределения заданий по ресурсам гетерогенной распределенной вычислительной среды (ГРВС), интегрирующей Grid и облачные вычисления [11], в случае отказа программно-аппаратного обеспечения в процессе выполнения схемы решения задачи и наличия незавершенных (остаточных) вычислений. Остаточная схема решения задачи формируется с использованием методов конкретизирующего программирования [12, 13]. В отличие от алгоритмов, используемых в вышеупомянутых системах управления workflow, предложенный алгоритм реализует адаптивное мультисценарное перераспределение заданий по ресурсам с использованием специальной системы метамониторинга [14], обеспечивающей выявление и частичное устранение отказов. Это позволяет существенно повысить отказоустойчивость вычислительного процесса [15].

Вычислительная модель. Обозначим через P, M и S соответственно множества параметров, программных модулей и схем решения задач вычислительной модели, описывающей предметную область пакета прикладных программ. Модули

из M реализуют отношения вычислимости на множестве параметров P так, что с каждым модулем $m_i \in M$ связаны множества его входных и выходных параметров $P_i^{in}, P_i^{out} \subset P$, $i \in \overline{1,n_m}$. Множество P_i^{in} включает параметры, значения которых должны быть известны, чтобы вычислить значения параметров, входящих во множество P_i^{out}.

Схема решения задачи $s \in S$ определяется тройкой $< M_s, X_0, Y_0 >$, где $M_s \subset M$ – множество модулей, которые нужно выполнить в процесс решения задачи, $X_0 \subset P$ – множество исходных параметров, значения которых заданы, $Y_0 \subset P$ – множество целевых параметров, значения которых нужно вычислить. В общем случае S является поливариантной схемой решения задачи и описывает альтернативные алгоритмы выполнения вычислительного процесса.

На рис. 1 приведены примеры поливариантной схемы решения задачи (а) и двух ее специализированных вариантов: с использованием для решения задачи модуля m_5 (б) и модуля m_6 (в), которые вычисляют значения одного и того же параметра, но применяют разные алгоритмы или обладают различными требованиями к ресурсам и показателями эффективности решения задачи.

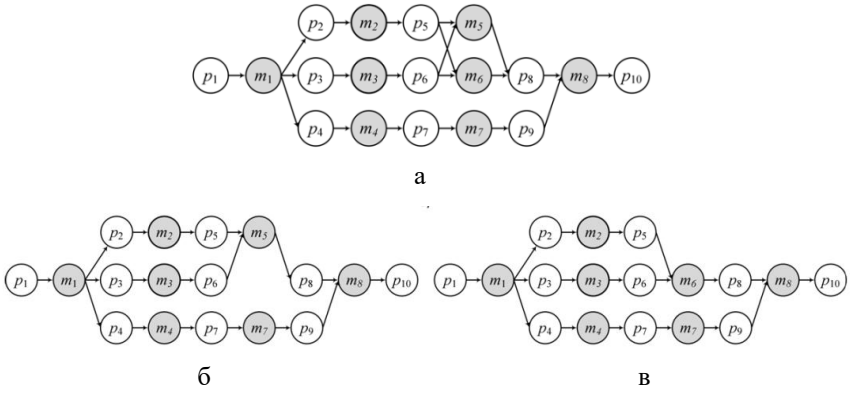

Рис. 1. Схемы решения задачи: поливариантная (а), схема с использованием модуля m_5 (б), схема с использованием модуля m_6 (в)

Выполнение схемы решения задачи S осуществляется путем ее интерпретации. Пусть в процессе интерпретации схемы S выполнено множество $M_e \subset M_s$ модулей, $|M_e| = n_e$. Тогда множество $M_u = M_s \setminus M_e$ будет включать модули схемы S, которые уже выполняются или ожидают своего запуска, $|M_e| = n_u$. Схему S_u, определяемую тройкой $< M_u, X_u, Y_u >$, будем называть остаточной схемой решения задачи (рис. 2). Множества X_u и Y_u, представляющие соответственно ее входные и выходные параметры, определяются по следующим формулам

$$X_u = X_o \cup \left(\bigcup_{l=1}^{n_e} P_{i_l}^{out} \right), \; Y_u = Y_o \cap \overline{\left(X_o \cup \left(\bigcup_{l=1}^{n_e} P_{i_l}^{out} \right) \right)},$$

где $i_l \in \overline{1,n_m} : m_{i_l} \in M_e$. Следует отметить, что множество X_u представляет промежуточные результаты вычислений.

На рис. 2 изображен фрагмент исходной схемы (рис. 1,а) с параметрами и модулями, относящимися к остаточной схеме решения задачи. Параметры $p_1 - p_6$ представляют собой результат промежуточных вычислений. Темно-серым цветом заливки выделены модули m_4 и m_5, которые выполняются. Модули m_7 и m_8 ожидают своего запуска. Модуль m_6 реализует вычислительную избыточность схемы и может быть использован в том случае, если решение задачи с использованием модуля m_5 будет невозможно.

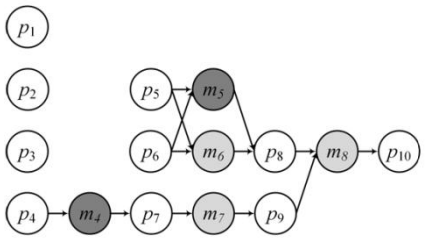

Рис. 2. Остаточная схема решения задачи

Мультиагентная система. Управление вычислениями реализуется МАС [16]. Пользователь пакета формулирует постановку задачи на вычислительной модели. Агент пользователя строит схему решения задачи, формирует виртуальное сообщество агентов, представляющих вычислительные ресурсы, и генерирует задание по выполнению схемы для агентов виртуального сообщества. В виртуальное сообщество включаются агенты, которые представляют ресурсы, подходящие по своим характеристикам для выполнения модулей схемы решения задачи. Соответствие характеристик ресурсов свойствам модулей осуществляется на основе заданной классификации заданий [17]. Задание описывает требования к среде, определяемые выполняемой прикладной программой (модулем). Систему классификации заданий создает администратор среды на основе своего экспертного опыта и с учетом вычислительной истории заданий, выполнявшихся на ресурсах среды.

При формировании виртуального сообщества агентов автоматически строится топология его коммуникационной сети. Топология строится в виде ориентированного дерева $G=<V,U>$, где V – множество вершин графа, представляющих агентов, U – множество ребер графа, представляющих коммуникационные связи между этими агентами. Дерево G описывается булевой матрицей смежности B размерности $n_a \times n_a$, элемент которой $b_{ij}=1$ означает, что существует дуга $\left(v_i, v_j\right)$, n_a – число агентов виртуального сообщества. Ориентированность дерева позволяет исключить возможность коллизий при передаче сообщений между агентами.

С целью координации работы виртуального сообщества его агенты выбирают своего лидера. Работа алгоритма выбора лидера характеризуется следующими особенностями: алгоритм является модификацией распределенного древесного волнового алгоритма [18]; любой из агентов, которому соответствует листовая вершина дерева G, может инициировать работу алгоритма; агенты взаимодействуют посредством посылки сообщений через коммуникационные каналы; коммуникационные протоколы гарантируют невозможность пропажи сообщений из-за ошибок коммуникаций или отката к контрольной точке; агенты используют дополнительные служебные сообщения (например, подтверждения о

доставке сообщений); алгоритм функционирует в фоновом режиме – агенты продолжают решение своих текущих задач при взаимодействии между собой путем обмена сообщениями.

Выбор одного из волновых алгоритмов для реализации выбора лидера виртуального сообщества агентов обусловлен рядом важных свойств, присущих алгоритмам данного класса [18]: завершение работы за конечное число шагов, обеспечение принятия определенного решения, причинно-следственная обусловленность принятого решения всеми участниками процесса. Результаты сравнения оценок сложности различных распределенных волновых алгоритмов по числу сообщений приведены в табл. 1, где k – это число обменов сообщениями алгоритма обхода, на котором базируется волновой алгоритм угасания. Оценки, приведенные в табл. 1, позволяют сделать вывод о преимуществе древесного алгоритма по числу обменов сообщениями в сравнении с другими волновыми алгоритмами.

Таблица 1

Оценка сложности алгоритмов по числу сообщений

Алгоритм	Топология коммуникационной среды	Оценка
Древесный	Дерево	$2 \times n_a - 2$
Ченя – Робертса	Кольцо	$O(n_a \log n_a)$
Петерсона/Долева – Клейва – Роде	Кольцо	$1.5 \times n_a \times \log n_a$
Угасания	Топология заранее неизвестна	$k \times n_a$
«Задиры»	Топология заранее неизвестна	$O(n_a^2)$

Алгоритм. Построение схемы решения задачи, назначение ресурсов для выполнения ее модулей и управление вычислительным процессом осуществляется МАС с использованием рыночных механизмов регулирования спроса и предложения ресурсов [16]. Для выполнения схемы формируется виртуальное сообщество агентов, представляющих ресурсы, которые наилучшим образом подходят по своим характеристикам модулям схемы.

За обнаружение отказов и их диагностику отвечает система метамониторинга. Информационно-вычислительная модель системы диагностики представлена в виде следующей структуры $S = < O, P, T, Z, F, R, PR, L, I >$, где O – множество исследуемых объектов в узлах, P – множество измеряемых характеристик (параметров) объектов, T – множество типов значений характеристик, Z – множество логических параметров, F – множество контрольно-диагностических операций (действий), R – множество типов операций, PR – множество продукций, L – журнал диагностики, I – периоды запуска диагностики вычислительного узла, соответствующие режимам эксплуатации узла в разные временные интервалы.

В системе метамониторинга рассматриваются следующие характеристики узлов, модулей и агентов: характеристики объемов вычислительной нагрузки компонентов узла (нагрузки процессоров, ядер, оперативной памяти, сетевых элементов, систем хранения данных и других структурных элементов); характеристики физического состояния компонентов узла (температура процессоров и материнских плат, работоспособность систем бесперебойного питания, жестких дисков и других структурных элементов); характеристики процесса выполнения модуля (приоритет и статус процесса, использованное процессорное время, объем используемой оперативной памяти, число обращений к жесткому диску и сетевым элементам и другие сведения); характеристики работы агента (аналогичные выполнению модуля на узле, дополненные характеристиками взаимодействия агента с другими агентами).

К основным типам отказов, идентифицируемых в системе метамониторинга относятся: нештатное завершение вычислительного процесса; нехватка оперативной памяти для выполнения процесса; сбои операций чтения/записи данных; недоступность требуемого объема свободного места в системе хранения данных для записи результатов вычислений; отсутствие доступа к предметным базам данных; нарушение взаимодействия с другими узлами и агентами среды (в том числе сбои передачи данных); отказы операционной системы узла, приводящие к потере его работоспособности; отказы аппаратных средств (систем охлаждения, сетевых интерфейсов, модулей памяти, процессоров и других устройств).

Виды, признаки и причины отказов объектов вычислительной среды с различной степенью их детализации описаны в [19]. В статье рассматриваются следующие основные объекты среды: вычислительные узлы, агенты и модули распределенного пакета прикладных программ.

В соответствии с выделенными объектами учитываются следующие отказы: отказ узла (узел находится в нерабочем состоянии либо не отвечает в течение заданного периода времени); отказ агента (агент не отвечает в течение заданного периода времени); отказ модуля (аварийное завершение выполнения модуля или некорректное вычисление его выходных параметров). Эти отказы обобщают различные причины и признаки неисправностей рассматриваемых объектов. Отказы приводят к необходимости выполнения модуля схемы решения задачи в резервном узле или другим агентом, передачи управления модулем иному агенту или выполнения другого модуля.

На рис. 3 приведены основные этапы работы алгоритма: идентификация отказа, обработка отказа, выбор сценария поддержки отказоустойчивости процесса выполнения схемы решения задачи и построение остаточной схемы.

Рис. 3. Основные этапы работы алгоритма

На первом этапе производится идентификация и локализация отказа с помощью системы метамониторинга. Затем агентами данной системы запускается процедура обработки отказа, включая частичное устранение его последствий и реконфигурацию совокупности доступных ресурсов, а также сбор и передачу необходимой информации агентам виртуального сообщества. На основе полученных данных агенты осуществляют выбор сценария поддержки отказоустойчивости процесса выполнения схемы. На заключительном этапе агенты формируют остаточную схему решения задачи в соответствии с выбранным сценарием на основе методов редукции избыточных вычислений.

Пусть Z – это множество простых и составных логических параметров, принимающих значения из множества $\{0,1,\theta\}, Z \subset P$. Составной параметр реализует логическое выражение, сформированное из простых параметров и логических операторов. Он не определен, если не определен хотя бы один из простых параметров. F и PR – это множества действий по обработке отказов и продукций, задающих условия применения действий: $pr_i : z_j \Rightarrow f_k$, где $z_j \Rightarrow f_k$ – ядро продукции, интерпретируемое как выбор действия $f_k \in F$ по обработке отказа в соответствии с условием $z_j \in Z$. Каждая продукция имеет приоритет.

Связи между продукциями и логическими параметрами в левых частях их ядер представлены булевой матрицей W размерности $n_p \times n_z$, где n_p – число продукций, n_z – число логических параметров. Элемент матрицы $w_{i,j} = 1$ означает, что продукция pr_i использует параметр z_j. Связи между продукциями и действиями представлены булевой матрицей C размерности $n_p \times n_f$, где n_f – число действий. Элемент матрицы $c_{i,k} = 1$ означает, что продукция pr_i описывает условия выполнения действия f_k. Зависимости между действиями представим булевой матрицей D размерности $n_f \times n_f$. Элемент матрицы $d_{i,j} = 1$ означает, что действие f_i зависит от действия f_j.

Работа алгоритма строится следующим образом:

1. Формулировка непроцедурной постановки задачи: «вычислить Y по X», где X – это множество параметров, содержащих информацию о диагностируемых объектах, а Y – множество параметров, содержащих информацию о результатах диагностики.
2. Формирование множества $PR^* = \left\{ pr_{i_1}, pr_{i_2}, ..., pr_{i_q} \right\}$ продукций, для которых определены истинные значения параметров $z_{j_1}, z_{j_2}, ..., z_{j_q}$: $w_{i_h, j_h} = 1, \ \forall h = \overline{1,q}$.
3. Если $PR^* = \varnothing$, то переход на шаг 7 (задача неразрешима).
4. Иначе $\forall h = \overline{1,q}$:
   a. выполнение действия f_k: $c_{i_h,k} = 1, \ P_k^{in} \subseteq X$;
   b. если действие f_k выполнено, то $X = X \cup P_k^{out}$, $F = F \setminus \{ f_k \}$, $PR = PR \setminus \{ p_{i_h} \}$.
5. Если $Y \subseteq X$, то переход на шаг 7 (задача решена).
6. Иначе переход на шаг 2.
7. Завершение работы алгоритма.

Экспериментальные исследования. Проведен сравнительный анализ отказоустойчивости процесса вычислений путем полунатурного моделирования. Синтетический поток заданий сформирован на основе вычислительной истории, полученной при решении крупномасштабных практических задач. Выполнение схем осуществлялось под управлением MAC, а также мета-планировщиков GridWay и CondorDAGMan.

В процессе работы систем моделировались все рассматриваемые типы отказов, интенсивность которых составляла около одного сбоя в час. Хотя, их число превышает реальную интенсивность на кластере, такая частота позволяет нагляднее показать достоинства и недостатки каждой из систем. Рис. 4 демонстрирует успешную обработку отказов узлов всеми тремя системами. К окончанию работы модели, не было ни одной незавершенной схемы решения задачи. На рис. 5 показаны преимущества MAC по обработке отказов компонентов системы управления. В таких случаях агент-лидер брал на себя управление узлом отказавшего агента. Он продолжал выполнения модулей в узле без ожидания перезапуска отказавшего агента. В то же время GridWay и CondorDAGMan ожидали восстановления своих отказавших компонентов и в некоторых случаях перезапускали модули. В случае отказа модуля, только MAC поддерживала отказоустойчивость, когда в схеме был предусмотрен альтернативный алгоритм решения задачи. Эксперименты проводились в ГРВС, организованной на базе ресурсов Иркутского суперкомпьютерного центра СО РАН [20], с использованием до 700 ядер.

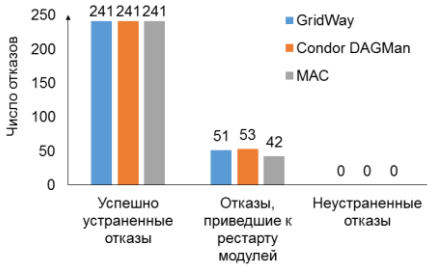

Рис. 4. Отказы узлов

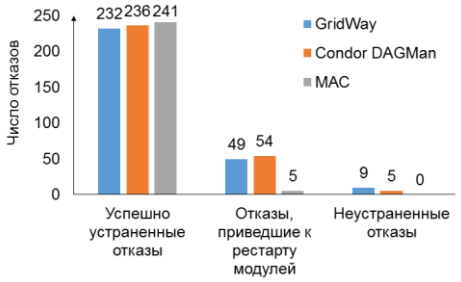

Рис. 5. Отказы систем управления

Заключение. В статье представлен адаптивный мультиагентный алгоритм, предназначенный для перераспределения заданий по ресурсам ГРВС при возобновлении процесса решения задач в распределенных пакетах прикладных программ после отказов программно-аппаратных средств. Работа данного алгоритма базируется в отличие от известных на методах конкретизирующего программирования для построения и выполнения остаточной схемы решения задачи, а также метамониторинга ресурсов среды.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Бондаренко А.А., Якобовский М.В.* Обеспечение отказоустойчивости высокопроизводительных вычислений с помощью локальных контрольных точек // Вестник Южно-Уральского государственного университета. Серия: Вычислительная математика и информатика. – 2014. – Т. 3, No. 3. – С. 20-36.
2. *Феоктистов А.Г., Сидоров И.А., Горский С.А.* Автоматизация разработки и применения распределенных пакетов прикладных программ // Проблемы информатики. – 2017, No. 4. – С. 61-78.
3. *Banti A., Kacsuk P., Kozlovszky M.* Classification of scientific workflows based on reproducibility analysis // Proceedings of the 39th International Convention on information and communication technology, electronics and microelectronics (MIPRO-2016). – Riejka: IEEE, 2016. – P. 327-331.
4. *Mhashilkar P., Miller Z., Kettimuthu R., Garzoglio G., Holzman B., Weiss C., Duan X., Lacinski L.* End-To-End Solution for Integrated Workload and Data Management using GlideinWMS and Globus Online // Journal of Physics: Conference Series. – 2012. – Vol. 396, No. 3. – P. 2076-2085.

5. *Talia D.* Workflow Systems for Science Concepts and Tools // ISRN Software Engineering. – 2013. – Vol. 2013. – P. 1-15.
6. *Deelman E., Peterka T., Altintas I., Carothers C.D., van Dam K.K., Moreland K., Parashar M., Ramakrishnan L., Taufer M., Vetter J.* The future of scientific workflows // The International Journal of High Performance Computing Applications. – 2017. Vol. 32, No. 1.1. – P. 159-175.
7. *Ostermann S., Plankensteiner K., Prodan R., Fahringer T., Iosup A.* Workflow monitoring and analysis tool for ASKALON // Proceedings of 3rd CoreGRID Workshop on Grid Middleware. – Spain: Springer, 2008. – P. 73-86.
8. *Zhao Y., Raicu I., Foster I.* Scientific Workflow Systems for 21st Century, New Bottle or New Wine? // IEEE Congress on Services - Part I. – Honolulu, HI: IEEE, 2008. – P. 467-471.
9. *Rodriguez M.A., Buyya R.* Deadline Based Resource Provisioning and Scheduling Algorithm for Scientific Workflows on Clouds // IEEE Transactions on Cloud Computing. – 2014. Vol. 2, No. 2. – P. 222-235.
10. *Anwar N., Deng H.* Elastic Scheduling of Scientific Workflows under Deadline Constraints in Cloud Computing Environments // Future Internet. – 2018. – Vol. 10, No. 1. – P. 1-23.
11. *Feoktistov A., Sidorov I., Sergeev V., Kostromin R., Bogdanova V.* Virtualization of Heterogeneous HPC-clusters Based on OpenStack Platform // Вестник Южно-Уральского государственного университета. Серия: Вычислительная математика и информатика. – 2017. – Т. 6, № 2. – С. 37-48.
12. *Ершов А.П.* Научные основы доказательного программирования // Вестник АН СССР. – 1984. – № 10. – С. 9-19.
13. *Ershov A.P.* On Mixed Computation: Informal Account of the Strict and Polyvariant Computation Schemes // Control Flow and Data Flow: Concepts of Distributed Programming. – Berlin A.O.: Springer-Verlag, 1985. – P. 107-120.
14. *Sidorov I.A.* Methods and Tools to Increase Fault Tolerance of High-Performance Computing Systems // Proceedings of the 39th International Convention on information and communication technology, electronics and microelectronics (MIPRO-2016). – Riejka: IEEE, 2016. – P. 242-246.
15. *Feoktistov A.G., Sidorov I.A.* Logical-Probabilistic Analysis of Distributed Computing Reliability // Proceedings of the 39th International Convention on information and communication technology, electronics and microelectronics (MIPRO-2016). – Riejka: IEEE, 2016. – P. 247-252.
16. *Феоктистов А.Г., Костромин Р.О., Дядькин Ю.А.* Управление заданиями в гетерогенной распределенной вычислительной среде на основе знаний // Вестник компьютерных и информационных технологий. – 2018. – № 2. – С. 10-17.
17. *Bychkov I., Feoktistov A., Kostromin R., Sidorov I., Edelev A., Gorsky S.* Machine Learning in a Multi-Agent System for Distributed Computing Management // Data Science. Information Technology and Nanotechnology 2018. CEUR-WS Proceedings. – 2018. – Vol. 2212. – P. 89-97.
18. *Tel G.* Introduction to Distributed Algorithms: Solutions and Suggestions. – Cambridge University Press, 2000. – 596 p.
19. *Balaji P., Buntinas D., Kimpe D.* Fault Tolerance Techniques for Scalable Computing // Scalable Computing and Communications: Theory and Practice. – Hoboken: Wiley-IEEE Press, 2013. – P. 212-245.
20. ЦКП Иркутский суперкомпьютерный центр СО РАН. – Режим доступа: http://hpc.icc.ru/ (дата обращения: 03.11.2018).

REFERENCES

1. *Bondarenko A.A., Yakobovski M.V.* Obespechenie otkazoustojchivosti vysokoproizvoditel'nyh vychislenij s pomoshch'yu lokal'nyh kontrol'nyh tochek [Fault tolerance for HPC by using local checkpoints], *Vestnik Yuzhno-Ural'skogo gosudarstvennogo universiteta. Seriya: Vychislitel'naya matematika i informatika* [Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering], 2014, Vol. 3, No. 3, pp. 20-36.
2. *Feoktistov A.G., Sidorov I.A. Gorky S.A.* Avtomatizatsiya razrabotki i primeneniya raspredelennykh paketov prikladnykh programm [Automation of development and application of distributed applied software packages], *Problemy informatiki* [Problems of Informatics], 2017, No. 4, pp. 61-78.

3. *Banti A., Kacsuk P., Kozlovszky M.* Classification of scientific workflows based on reproducibility analysis, *Proceedings of the 39th International Convention on information and communication technology, electronics and microelectronics (MIPRO-2016)*, Riejka: IEEE, 2016, pp. 327-331.
4. *Mhashilkar P., Miller Z., Kettimuthu R., Garzoglio G., Holzman B., Weiss C., Duan X., Lacinski L.* End-To-End Solution for Integrated Workload and Data Management using GlideinWMS and Globus Online, *Journal of Physics: Conference Series*, 2012, Vol. 396, No. 3, pp. 2076-2085.
5. *Talia D.* Workflow Systems for Science Concepts and Tools, *ISRN Software Engineering*, 2013, Vol. 2013, pp. 1-15.
6. *Deelman E., Peterka T., Altintas I., Carothers C.D., van Dam K.K., Moreland K., Parashar M., Ramakrishnan L., Taufer M., Vetter J.* The future of scientific workflows, *The International Journal of High Performance Computing Applications*, 2017, Vol. 32, No. 1.1, pp. 159-175.
7. *Ostermann S., Plankensteiner K., Prodan R., Fahringer T., Iosup A.* Workflow monitoring and analysis tool for ASKALON, *Proceedings of 3rd CoreGRID Workshop on Grid Middleware*, Spain: Springer, 2009, pp. 73-86.
8. *Zhao Y., Raicu I., Foster I.* Scientific Workflow Systems for 21st Century, New Bottle or New Wine?, *IEEE Congress on Services - Part I*, Honolulu, HI: IEEE, 2008, pp. 467-471.
9. *Rodriguez M.A., Buyya R.* Deadline Based Resource Provisioning and Scheduling Algorithm for Scientific Workflows on Clouds *IEEE Transactions on Cloud Computing*, 2014, Vol. 2, No. 2, pp. 222-235.
10. *Anwar N., Deng H.* Elastic Scheduling of Scientific Workflows under Deadline Constraints in Cloud Computing Environments *Future Internet*, 2018, Vol. 10, No. 1, pp. 1-23.
11. *Feoktistov A., Sidorov I., Sergeev V., Kostromin R., Bogdanova V.* Virtualization of Heterogeneous HPC-clusters Based on OpenStack Platform, *Vestnik Yuzhno-Ural'skogo gosudarstvennogo universiteta. Seriya: Vychislitel'naya matematika i informatika* [Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering], 2017, Vol. 6, No. 2, pp. 37-48.
12. *Ershov A.P.* Nauchnye osnovy dokazatel'nogo programmirovaniya [Scientific basis of evidence-based programming], *Vestnik AN SSSR* [Herald of the Russian Academy of Sciences], 1984, No. 10, pp. 9-19.
13. *Ershov A.P.* On Mixed Computation: Informal Account of the Strict and Polyvariant Computation Schemes, *Control Flow and Data Flow: Concepts of Distributed Programming*, Berlin A.O.: Springer-Verlag, 1985, pp. 107-120.
14. *Sidorov I.A.* Methods and Tools to Increase Fault Tolerance of High-Performance Computing Systems, *Proceedings of the 39th International Convention on information and communication technology, electronics and microelectronics (MIPRO-2016)*, Riejka: IEEE, 2016, pp. 242-246.
15. *Feoktistov A.G., Sidorov I.A.* Logical-Probabilistic Analysis of Distributed Computing Reliability, *Proceedings of the 39th International Convention on information and communication technology, electronics and microelectronics (MIPRO-2016)*, Riejka: IEEE, 2016, pp. 247-252.
16. *Feoktistov A.G, Kostromin R.O., Dyadkin Y.A.* Upravlenie zadaniyami v geterogennoy raspredelennoy vychislitel'noy srede na osnove znaniy [Knowledge Based Management of Jobs in Heterogeneous Distributed Computing Environment], *Vestnik komp'iuternykh i informatsionnykh tekhnologii* [Herald of computer and information technologies], 2018, No. 2, pp. 10-17.
17. *Bychkov I., Feoktistov A., Kostromin R., Sidorov I., Edelev A., Gorsky S.* Machine Learning in a Multi-Agent System for Distributed Computing Management, *Data Science. Information Technology and Nanotechnology 2018*, CEUR-WS Proceedings, 2018, Vol. 2212. pp. 89-97.
18. *Tel G.* Introduction to Distributed Algorithms: Solutions and Suggestions, *Cambridge University Press*, 2000, 596 p.
19. *Balaji P., Buntinas D., Kimpe D.* Fault Tolerance Techniques for Scalable Computing, *Scalable Computing and Communications: Theory and Practice*, Hoboken: Wiley-IEEE Press, 2013, pp. 212-245.
20. Irkutsk Supercomputer Centre of SB RAS. Available at: http://hpc.icc.ru/ (accessed 3 November 2018).

Статью рекомендовал к опубликованию д.т.н. А.В. Петров.

Феоктистов Александр Геннадьевич – Институт динамики систем и теории управления им. В.М. Матросова СО РАН; e-mail: agf65@yandex.ru; 664033, г. Иркутск, ул. Лермонтова, 134; тел.: +79247116704; к.т.н.; доцент; в.н.с.

Костромин Роман Олегович – e-mail: romang70055@gmail.com; тел.: +79041150109; аспирант.

Сидоров Иван Александрович – e-mail: yvan.sidorov@gmail.com; тел.: +79027668364; к.т.н.; научный сотрудник.

Горский Сергей Алексеевич – e-mail: gorskysergey@mail.ruж; тел.: +79149230010; к.т.н.

Feoktistov Aleksandr Gennadyevich – Matrosov Institute for System Dynamics and Control Theory of SB RAS; e-mail: agf65@yandex.ru; 134, Lermontov street, Irkutsk, 664033, Russia; phone: +79247116704; cand. of eng. sc.; associate professor; leading researcher.

Kostromin Roman Olegovich – e-mail: romang70055@gmail.com; phone: +79041150109; post-graduate student.

Sidorov Ivan Aleksandrovich – e-mail: yvan.sidorov@gmail.com; phone: +79027668364; cand. of eng. sc.; research officer.

Gorsky Sergey Alexeevich – e-mail: gorskysergey@mail.ru; phone: +79149230010; cand. of eng. sc.

УДК 007.52 DOI 10.23683/2311-3103-2018-8-69-83

Л.А. Мартынова

МЕТОД РАЗРЕШЕНИЯ КОНФЛИКТА В МУЛЬТИАГЕНТНОЙ СИСТЕМЕ УПРАВЛЕНИЯ АВТОНОМНОГО НЕОБИТАЕМОГО ПОДВОДНОГО АППАРАТА С ИСПОЛЬЗОВАНИЕМ РАСПРЕДЕЛЕННЫХ ВЫЧИСЛЕНИЙ*

Целью исследования является повышение эффективности функционирования автономного необитаемого подводного аппарата (АНПА) за счет разрешения конфликта в его мультиагентной системе управления, связанного с энергопотреблением подсистемами АНПА. При переходе от исключительного использования аккумуляторной батареи к дополнительному использованию аккумуляторной батареи возникла необходимость разрешения противоречия между предоставлением энергоресурса разнородными источниками и его потребления. Сложность решения задачи заключалась в непредсказуемости использования различных скоростных режимов, оказывающих влияние на энергорасход АНПА. Предложенный метод разрешения конфликта основан на декомпозиции потребителей энергоресурса и прогнозировании возможности выполнения проставленной перед АНПА задачи, связанной с преодолением заданного расстояния в пределах заданного времени. Для этого в работе разработан метод, в основу которого положены алгоритмы: – прогнозирования достаточности энергоресурса для преодоления заданной дистанции; – определения допустимого текущего расхода энергоресурса и соответствующего ему скоростного режима; – оценки времени, необходимого для преодоления оставшейся дистанции. Перечисленные алгоритмы характеризуются набором параметров, которые были оптимизированы в зависимости от сложившихся условий в ходе выполнения аппаратом задания. В качестве критерия оптимальности использована вероятность преодоления заданной дистанции в пределах заданного времени. При оптимизации параметров учитывались: – текущий уровень заряда аккумуляторной батареи; – текущий уровень запаса электрохимического генератора; – время, в течение которого аппарат уже преодолел часть заданной дистанции. По этим данным последовательно определялись: – оставшаяся дистанция; – время, затраченное на преодоление оставшейся дистанции; – запас энергоресурса, израсходован-

* Работа выполнена при поддержке РФФИ, проект № 17-08-00666.