УДК 004.722

Методология концептуализации и классификации потоков заданий масштабируемых приложений в разнородной распределенной вычислительной среде

Феоктистов А. Г.

Постановка задачи: *современные распределенные вычислительные среды характеризуются высокой конкуренцией пользователей за общие ресурсы этих сред, возникающей в процессе выполнения пользовательских заданий. Проблема оптимального выделения ресурсов заданиям актуализирует решение задач выявления, описания, классификации и применения информации о характеристиках ресурсов и заданий. В настоящее время традиционные системы управления ресурсами для распределенных вычислительных сред не обладают всеми необходимыми средствами для эффективного решения вышеперечисленных задач.* ***Целью работы*** *является разработка методологии концептуализации и классификации потоков заданий масштабируемых приложений в распределенных вычислительных средах, включающей специализированные модели и методы представления и использования знаний о заданиях и ресурсах, а также технологию практического применения этих моделей и методов.* ***Используемые методы****: в процессе классификации заданий и их потоков применяется признаковое описание классифицируемых объектов с использованием числовых и нечисловых характеристик. В случае классификации заданий в качестве признаков используются вычислительные характеристики заданий, при классификации потоков заданий – структурные и поведенческие характеристики этих потоков. Распознавание свойств заданий и их потоков осуществляется с помощью набора специализированных характеристических функций. Детальная настройка требований, содержащихся в классифицированных заданиях, осуществляется на основе методов конкретизирующего программирования.* ***Новизна:*** *аналоги предложенных в статье средств классификации заданий и их потоков в используемых на практике системах управления ресурсами для распределенных вычислительных сред автору неизвестны.* ***Результат****: использование представленных средств классификации заданий и их потоков в качестве надстройки к системе управления ресурсами позволяет существенно улучшить результаты распределения этих ресурсов.* ***Практическая значимость****: результаты моделирования показывают, что применение разработанных систем классификации заданий и их потоков позволяет оптимизировать распределение вычислительных ресурсов и обеспечивает существенное повышение ряда важных показателей эффективности функционирования распределенной вычислительной среды. Практическое использование этих систем подтверждает полученные модельные результаты.*

Ключевые слова: *распределенная вычислительная среда, масштабируемые приложения, потоки заданий, концептуализация и классификация.*

Ведение

На основе анализа тенденций развития, организации и использования современных разнородных распределенных вычислительных сред (РВС) [1-4] можно сделать вывод, что в последнее десятилетие особую актуальность приобрели исследования, связанные с проблемой усиления предметной ориентации технологий организации таких сред. Движение в этом направлении обусловлено необходимостью все более эффективного интегрированного использования разнородных ресурсов РВС, а также высокоуровневой поддержки пользователей-«предметников» в процессе разработки и запуска масштабируемых приложений. Особое внимание специалистов в области высокопроизводительных вычислений уделяется развитию фундаментальных основ организации распределенных вычислений при решении

крупномасштабных научных задач различных предметных областей, а также разработке новых методов и алгоритмов управления масштабируемыми вычислениями и организации вычислительного процесса в на базе гетерогенных ресурсов РВС в условиях динамически изменяющейся производительности этих ресурсов [5-7]. Высокая конкуренция заданий за общие ресурсы РВС приводит к необходимости всестороннего учета характеристик этих ресурсов в процессе их распределения с целью достижения требуемого качества выполнения задания [8]. Особую актуальность имеют исследования связанные с вопросами многокритериальной оценки качества вычислительных работ (сервисов), обеспечиваемого ресурсами в процессе выполнения задания [9-11].

Необходимость создания масштабируемых приложений возникает при выполнении многовариантных расчетов, решении широкого спектра переборных задач и многих других. Предполагается, что масштабируемое приложение включает набор прикладных программ для параллельного решения задачи с помощью различных вычислительных единиц (например, ядер) разнородных узлов РВС и порождает комбинированный поток, объединяющий задания для этих прикладных программ. При этом вычислительная нагрузка, связанная с решением задачи, распределяется между вычислительными единицами разнородных узлов РВС, а время выполнения заданий комбинированного потока уменьшается обратно пропорционально числу используемых вычислительных единиц с учетом их производительности в составе конкретного узла. Создание системы управления комбинированными потоками заданий для распределенной программной системы является нетривиальной и весьма актуальной проблемой. Для успешного решения этой проблемы необходимо, чтобы пользовательские приложения такого рода включали возможности, во-первых, мониторинга состояния узлов РВС (их доступности, готовности, надежности, параметров очередей, статусов запущенных заданий и др.) и гибкого управления заданиями (учета требований к вычислительной системе, запуска, рестарта и миграции заданий, поддержки механизмов создания контрольных точек), во-вторых, динамической декомпозиции исходной задачи на подзадачи на основе анализа алгоритмов решения задачи и вычислительных характеристик узлов, назначения этих узлов для решения в них подзадач и последующей генерации потоков заданий для прикладных программ, размещенных в выбранных узлах.

Основные понятия классификации заданий

Задача в самом общем смысле – это ситуация, определяющая действия некоторой решающей системы, в состав которой входят и люди, и автоматы (машины) [12]. Чтобы осуществить решение задачи, такая система должна обладать средствами и способами решения. Основными этапами процесса решения задачи являются [13]:

- постановка задачи;
- построение плана решения задачи;
- выполнение процесса решения задачи в соответствии с планом;

- анализ результатов решения задачи.

К вычислительным задачам, относятся задачи, процесс решения которых осуществляется путем взаимодействия пользователя (специалиста предметной области) с вычислительной системой. Процесс решения вычислительной задачи характеризуется, как правило, относительно небольшим набором данных и большим объемом вычислений, выполняемых над этими данными, и включает два основных этапа:

- формирование задания вычислительной системе для решения задачи;
- выполнение этого задания в вычислительной системе.

Задание представляет собой спецификацию процесса решения задачи, содержащую информацию о требуемых вычислительных ресурсах, исполняемых прикладных программах, входных/выходных данных, а также другие сведения, необходимые вычислительной системе для успешного выполнения процесса решения задачи. Множество заданий пользователя, поступающих в вычислительную систему, составляют поток заданий, который в дальнейшем может сливаться с потоками заданий других пользователей, образуя новые потоки. Поток заданий характеризуется следующими свойствами: мощность и структура потока, порядок поступления и обслуживания заданий, степень изменения перечисленных свойств во времени, характер взаимосвязи заданий, уровень платформенной независимости приложений, выполнение которых требуется в задании. Задания могут объединяться в пакеты – конечные наборы заданий, обладающие однородными характеристиками.

Вычислительная задача называется ресурсоемкой, если для ее решения специалисту предметной области не хватает ресурсов доступных ему компьютеров: производительности процессоров, объемов оперативной и дисковой памяти, пропускной способности телекоммуникационной среды. В этом случае число операций, выполняемых вычислительной системой над каждой переменной в соответствии с алгоритмом решения ресурсоемкой задачи, должно, как правило, на несколько порядков превышать быстродействие этих компьютеров.

Процесс решение ресурсоемких вычислительных задач может требовать применения вычислительных систем разного уровня – от относительно простых, имеющих в своем составе типовые вычислительные модули, до специализированных высокопроизводительных серверов и суперкомпьютеров [14]. Ускорение процесса решения ресурсоемких вычислительных задач достигается за счет применения параллельных и распределенных вычислений. Распределенные вычисления – это способ решения задач с использованием нескольких вычислительных устройств. Парадигма параллельных вычислений включает всю совокупность вопросов, относящихся к разработке параллельных алгоритмов решения задачи, организации параллельного использования вычислительных ресурсов для решения задачи и гибкому управлению параллельными процессами решения задачи с целью достижения наибольшей эффективности использования вычислительной системы. Любые вычислительные устройства можно считать параллельной вычислительной

системой, если они работают одновременно и их можно использовать для решения одной задачи. Параллельные или распределенные вычисления являются высокопроизводительными, если существенно ускоряют процесс решения задачи.

При решении задачи в распределенной или параллельной вычислительной системе требуется, во-первых, построить план решения задачи и, во-вторых, распределить ресурсы системы с целью эффективного выполнения процесса решения задачи. Эффективность использования того или иного ресурса для выполнения конкретного задания зависит от степени соответствия ресурса требованиям (вычислительным характеристикам) задания. Когда ресурсы вычислительной системы разнородны, возникает необходимость классификации зданий с целью повышения эффективности процесса распределения ресурсов: если каждому ресурсу поставлены в соответствие определенные классы заданий, то выбор ресурса для выполнения задания конкретного класса существенно упрощается.

Наиболее простым способом классификации объектов является их признаковое описание с использованием числовых и/или нечисловых признаков. В случае классификации заданий в качестве признаков используются вычислительные характеристики заданий.

Вычислительные характеристики заданий

В настоящее время вычислительные кластеры являются неотъемлемым технологическим элементом процесса эффективного решения фундаментальных и прикладных задач. Вычислительные кластеры организуются на базе различных программно-аппаратных платформ, архитектур и коммуникационных сред, и, вследствие этого, существенно отличаются по своим показателям производительности, эффективности и надежности. При объединении нескольких кластеров в единую вычислительную среду (интегрированную кластерную систему), представляющую собой частный случай вычислительной Grid, возникает проблема оптимизации распределения по кластерам потока заданий, поступающих в эту среду.

Формирование и выполнение задания осуществляется с помощью специальных утилит той системы управления ресурсами, которая используется в вычислительной среде. К таким системам относятся: во-первых, глобальные планировщики (метапланировщики) заданий на уровне Grid типа GridWay [15]; во-вторых, локальные системы управления прохождением заданий (СУПЗ) вычислительных кластеров, такие, например, как PBS [16] или Condor [17].

Анализ результатов исследований (таблица 1), затрагивающих в той или иной степени вопросы классификации вычислительных задач и способов их постановки [12, 14, 18, 19], а также рассмотрение способов спецификации процессов решения задач в известных метапланировщиках Grid и кластерных СУПЗ позволяет выявить целый ряд важных характеристик заданий. В их числе: способ решения задачи (использование готовой программы, построение алгоритма решения задачи на основе библиотек стандартных программ,

исполнение программы в режиме интерпретации или компиляции); место размещения (в машине пользователя или в машине системы) программы, используемой для решения задачи; число прогонов программы; наличие взаимосвязанных подзаданий; вид параллелизма алгоритма решения задачи (мелкозернистый, крупноблочный и смешанный параллелизм), степень ресурсоемкости вычислений (малые, средние и большие задания); необходимость управления процессом вычислений в пакетном или интерактивном режимах; срочность вычислений и ряд других особенностей процесса выполнения заданий. Значение той или иной характеристики может быть явно задано пользователем вычислительной среды, формирующим свое задание, получено из переменной окружения этой среды, установлено системой управления заданиями как значение по умолчанию или взято из конфигурационных и статистических файлов (системных или пользовательских).

Таблица 1 – Подходы к классификации вычислительных задач

Подход к классификации заданий	Основы классификации
Классификация задач с точки зрения реализации в алгоритмах их решения тех или иных численных методов (см., например, [18, 19]).	Методы численного дифференцирования и интегрирования, решения систем линейных алгебраических уравнений, решения дифференциальных уравнений в частных производных и др.
Классификация задач по способам решения задач (см., например, [19-24]).	Использование готовой программы; программирование алгоритма решения задачи; построение алгоритма решения задачи на основе библиотек стандартных программ; синтез программ.
Классификация задач по их количественным характеристикам (см., например, [20, 25]).	Малые, средние и большие задачи; задачи вычислительные (мало данных, много вычислений) и задачи обработки данных (много данных, мало вычислений).
Классификация по возможности одновременного выполнения вычислений (см., например, [14]).	Параллельные и последовательные вычисления.
Классификация по степени параллелизма алгоритмов решения задач (см., например, [26]).	Мелкозернистый, крупноблочный и смешанный параллелизм.
Классификация вычислительных заданий (см., например, [27]).	Просты, взаимосвязанные, интерактивные и др. задания.

Вычислительные характеристики потоков заданий, задаются администратором РВС, а сами потоки заданий связываются с приложениями, порождающими эти потоки.

Подходы к классификации вычислительных заданий в системах управления заданиями

Известные на сегодняшний день метапланировщики Grid базируются на двух способах распределения ресурсов [28]: путем взаимодействия метапланировщика с кластерными СУПЗ (например, GridWay) либо за счет реализации планировщика, занимающегося поиском свободных ресурсов в Grid для выполнения конкретного приложения (например, AppLeS [29]). В обоих случаях, как правило, они не производят классификацию заданий на основе характеристик заданий и осуществляют оптимизацию выделения ресурсов, разделяемых между всеми заданиями потока, для выполнения конкретного задания, оперируя информацией, касающейся только этого задания. Такой подход, зачастую, не позволяет добиться высоких показателей функционирования интегрированной кластерной системы в целом: ее пропускной способности, эффективности загрузки ресурсов и времени выполнения некоторого набора задач. Отдельные системы, которые поддерживают классификацию заданий в том или ином виде (например, система управления заданиями программного комплекса gLite [30]), имеют, зачастую, достаточно ограниченный, нерасширяемый набор встроенных типов (классов) заданий и не осуществляют предварительную виртуальную декомпозицию вычислительных ресурсов относительно этих типов с целью оптимизации процесса выбора ресурса для выполнения заданий определенного типа.

Ниже приведены примеры классификации заданий в некоторых используемых на практике системах управления заданиями.

В системе Condor явно выделяются два класса заданий: простые и взаимосвязанные. Задания первого класса запускаются с помощью утилиты *condor_sabmit* Задания второго класса запускаются с помощью утилиты *condor_sabmit_dag*. Дальнейшая классификация осуществляется неявно – путем указания значений различных параметров в паспорте задания.

Программный комплекс gLite разрабатывалась в качестве промежуточного программного обеспечения проекта Enabling Grids for E-sciencE (EGEE) [31], завершенного в 2010 г. В настоящее время развитие и поддержка системы gLite осуществляется в рамках платформы European Middleware Initiative (EMI) [32], используемой для поддержки крупномасштабных научных исследований, например, осуществляемых в рамках проекта Worldwide Large Hadron Collider Computing Grid [33].

Основная и наиболее развитая часть в составе комплекса gLite – это система управления заданиями Workload Management System (WLMS) [30]. Ее назначение – поддержка выполнения программ на распределенных компьютерах, организованных в единую систему. Программный код задания (программа) выполняется на исполнительном компьютере без участия пользователя (в пакетном режиме). Сам код в рядовом случае не требует адаптации к условиям распределенной вычислительной среды. В некоторых случаях может потребоваться его дополнение прологом/эпилогом, которые выполняют подготовительные/завершающие операции, например, доставку

обрабатываемых данных. Задание представляется WLMS в виде формализованного описания, составленного на языке Job Description Language (JDL). Ресурсы распределенной вычислительной среды используются коллективно множеством пользователей, поэтому задание не обязательно начинает выполняться сразу после запуска: оно может ждать освобождения ресурсов, занятых другими заданиями. Ожидающие ресурсов задания хранятся в очередях. WLMS поддерживает работу как с простыми заданиями, так и с составными.

При классификации заданий в WLMS тип задания описывается двумя атрибутами: Type и JobType. Атрибут Type имеет, соответственно, значения «Job» или «DAG». Значение «DAG» (Direct Acyclic Graph of dependent jobs) атрибута Type определяет задание, в котором должны быть выполнены в определенной последовательности ряд простых заданий. Для простого задания применим второй атрибут – JobType, который может принимать следующие значения:

- «Normal» – обыкновенное задание;
- «Interactive» – интерактивное задание;
- «MPICH» – параллельное задание;
- «Checkpointable» – задание с контрольными точками;
- «Partitionable» – сериализуемое (многовариантное) задание.

Перечисленные характеристики могут сочетаться друг с другом, например:

JobType = {«Checkpointable», «MPICH»}.

Более гибкие возможности реализованы в инструментальном комплексе DISCENT [27]. В данном инструментарии для классификации заданий используется структура (*a* / *b* / *c* / *d*), где символы *a*, *b*, *c* и *d* имеют следующее назначение:

- *a* – характеристика алгоритма решения задачи с точки зрения наличия подзадач; фиксированными значениями для *a* являются 1 (отсутствие подзадач) и *K* (наличие подзадач);
- *b* – характеристика алгоритма решения задач с точки зрения наличия в нем параллелизма; фиксированными значениями для *b* являются *L* (крупноблочный параллелизм алгоритма), *F* (мелкозернистый параллелизм алгоритма) и *S* (последовательный алгоритм); недопустимыми комбинациями значений *a* и *b* являются (*a* = 1 / *b* = *L* / …) и (*a* = *K* / *b* = *F* / …);
- *c* – характеристика процесса решения задачи с точки зрения необходимости выполнения многовариантных расчетов; фиксированными значениями для *c* являются 1 (один единственный вариант данных) и *N* (наличие множества вариантов данных);
- *d* – характеристика процесса решения задачи с точки зрения места размещения выполняемого программного приложения; фиксированными значениями для *d* являются *G* (выполнение

приложения, размещенного в узлах РВС) и *R* (выполнение удаленного приложения).

Ниже приведены примеры классов заданий, определяемых с помощью этой системы классификации:

- (*a* = 1 / *b* ≠ *L* / *c* / *d* = *R*) – стандартные задания (скомпилированные программы пользователей), для выполнения которых используется штатный набор функций СУПЗ;
- (*a* / *b* / *c* = *N* / *d*) – многовариантные задания, для решения которых необходимо выполнение программы с различными наборами данных;
- (*a* / *b* / *c* / *d* = *G*) – задания, в которых требуется выполнение приложений, размещенных в узлах РВС;
- (*a* = *K* / *b* = *L* / *c* / *d*) – задания, требующие выполнения набора взаимозависимых подзаданий, включенных в процесс решения одной общей задачи.

Модель системы классификации заданий

Пусть имеется конечное множество характеристик заданий $H = \{h_1, h_2, ..., h_k\}$. Каждая характеристика h_i описывается информационной структурой, имеющей следующие компоненты: D_i – область допустимых значений характеристики h_i, включающая символ неопределенности θ; $r_i \geq 1$ – целочисленный ранг характеристики h_i, определяющий степень важности данной характеристики; $w_i \geq 0$ – вес характеристики h_i, представляющий собой численное выражение важности данной характеристики. Будем считать, что элементы множества H частично упорядочены по их рангу по убыванию: $r_i \geq r_{i+1}$, $\forall i \in \{1, 2, ..., k-1\}$.

Определим конечное множество классов заданий $C = \{c_1, c_2, ..., c_m\}$. Каждый класс c_j определяется базовым (обязательным) и дополнительным (необязательным) наборами характеристик из H. Характеристики базового и дополнительного набора для классов заданий удобно представить в виде булевых матриц A и B размерности $k \times m$, элементы которых $a_{ij} = 1$ или $b_{ij} = 1$ означают, что характеристика h_i принадлежит базовому или дополнительному набору, используется в определении класса c_j и для этого класса имеет конкретизированную область допустимых значений $D_{ij}^* \subseteq D_i \setminus \{\theta\}$. Если $a_{ij} \vee b_{ij} = 0$, то $D_{ij}^* \equiv \{\theta\}$. Матрицы A и B должны удовлетворять следующим условиям $\bigvee_{j=1}^{m} \bigwedge_{i=1}^{k} \overline{a_{ij}} = 0$; $\bigvee_{i=1}^{k} \bigvee_{j=1}^{m} (a_{ij} \wedge b_{ij}) = 0$.

Формирование множества характеристик H (определение имен, областей допустимых значений, рангов и весов характеристик) и создание на их основе множества классов заданий C (определение для каждого класса множеств обязательных и необязательных характеристик из множества H с указанием их областей значений, допустимых для конкретного класса) выполняется администратором интегрированной кластерной системы. Как правило,

характеристикам классов заданий, используемым в качестве обязательных характеристик, присваиваются более высокие ранги и веса. При поступлении задания в интегрированную кластерную систему классификатор заданий автоматически проверяет, какие характеристики из множества H использованы в спецификации этого задания, и определяет для каждой из них область значений, требуемую для выполнения задания в его спецификации.

Задание со всеми своими характеристиками представляется булевым вектором x размерности k. Между индексами элементов вектора x и индексами характеристик из H установлено взаимно однозначное соответствие. Значение i-го элемента вектора x определяется следующим образом:

$$x_i = \begin{cases} 0, \textit{если } D'_i \equiv \{\theta\}, \\ 1, \textit{если } D'_i \subseteq D_i \setminus \{\theta\}, \end{cases}$$

где D'_i – это область значений характеристики h_i, требуемых для выполнения данного задания. Вектор x должен удовлетворять условию $\bigwedge_{i=1}^{k} \overline{x_i} = 0$.

Соответствие требуемых областей допустимых значений характеристик задания областям допустимых значений характеристик j-го класса заданий из C устанавливается с помощью характеристической функции

$$\chi_j(x) = \begin{cases} 0, \textit{если } \exists i : (a_{ij} \vee b_{ij} = 1) \wedge (D'_i \not\subseteq D^*_{ij}), \\ 1, \textit{в противном случае}, \end{cases}$$

где $i \in \{1, 2, ..., k\}$, $j \in \{1, 2, ..., m\}$.

Для выполнения первичной классификации задания достаточно применения функции χ. В результате первичной классификации задание может быть соотнесено сразу с несколькими классами заданий из C.

Однако, может возникнуть необходимость в более детальной конкретизации классификации задания, полученной с помощью функции χ. В этом случае следует принимать во внимание число характеристик задания, требуемые области допустимых значений которых удовлетворяют соответствующим областям допустимых значений характеристик того или иного класса, и учитывать ранги и веса этих характеристик.

Ниже введен ряд вспомогательных функций.

Будем говорить, что i-я характеристика используется как в спецификации задания, так и в описании класса c_j, если выполняется условие

$$\overline{x_i} \vee \overline{a_{ij}}\,\overline{b_{ij}} = 0. \qquad (1)$$

Область требуемых значений i-ой характеристики соответствует области допустимых значений этой характеристики для класса c_j, если выполняется условие

$$D'_i \subseteq D^*_{ij}. \qquad (2)$$

Функция $\rho_j(x)$ вычисляет оценку возможности (вероятность) отнесения задания к классу c_j на основе характеристик этого задания, удовлетворяющих

по всем параметрам характеристикам данного класса c_j. Значение функции определяется выражением $\rho_j(x) = \frac{V_j}{S}$, где $V_j = \frac{k_j}{n_j}$, $S = \sum_{l=1}^{m} \frac{k_l}{n_l}$, k_j (k_l) – число характеристик задания, удовлетворяющих условиям (1) и (2) относительно j-го (l-го) класса, n_j и n_l – число характеристик, определяющих классы c_j и c_l соответственно, $k_j, k_l \in \{0, 1, ..., k\}$, $n_j, n_l \in \{1, ..., m\}$. Нетрудно заметить, что $\rho_j(x) \in [0,1]$ и $\sum_{j=1}^{m} \rho_j(x) = 1$.

Функция $\sigma_j(x)$ вычисляет агрегированный числовой показатель важности характеристик задания для класса c_j с учетом рангов этих характеристик. Значение функции определяется выражением $\sigma_j(x) = n_t s_t + \sum_{l=1}^{t-1}(n_l s_l + k s_{l+1})$, где $s_1, s_2, \ldots, s_t$ – упорядоченные по убыванию значения рангов, n_t и n_l – число характеристик задания, удовлетворяющих условиям (1) и (2), а также дополнительным условиям $r_i = s_t$ и $r_i = s_l$ соответственно; $\sigma_j(x) \geq 1$.

Функция $\omega_j(x)$ вычисляет сумму весов характеристик задания для класса c_j. Значение функции определяется выражением $\omega_j(x) = \sum_{i \in I} w_i$, где I – это множество индексов характеристик задания, удовлетворяющих условиям (1) и (2); $\omega_j(x) \geq 0$.

Функция $\phi_j(x, z)$ вычисляет оценку возможности (вероятность) отнесения задания к классу c_j с учетом его вычислительной истории z – информационной структуры, содержащей статистику о выполнении в интегрированной кластерной системе заданий с аналогичными характеристиками. Данная информация извлекается из статистических и информационных файлов СУПЗ, применяемых в интегрированной кластерной системе. Значение функции определяется выражением

$$\varphi_j(x, z) = \begin{cases} \frac{1}{m}, & \text{если } \forall l \in \{1, 2, ..., m\}\ \gamma_l(x, z) = 0, \\ \frac{\gamma_j(x, z)}{\sum_{l=1}^{m} \gamma_l(x, z)}, & \text{в противном случае,} \end{cases}$$

где $\gamma_l(x, z)$ и $\gamma_j(x, z)$ – функции, возвращающие число выполненных в интегрированной кластерной системе заданий классов c_l и c_j с аналогичными характеристиками. Нетрудно заметить, что $\phi_j(x, z) \in [0,1]$ и $\sum_{j=1}^{m} \phi_j(x, z) = 1$.

Определим также верхние границы эквивалентности значений функций ρ, σ, ω и ϕ – δ_ρ, δ_σ, δ_ω и δ_ϕ соответственно. Два значения одной и той же

функции будем считать эквивалентными, если модуль их разницы будет меньше соответствующей верхней границы эквивалентности, либо равен ей.

Пусть с помощью функции χ сформирован булев вектор y размерности m, элемент которого $y_j = 1$ означает, что задание относится к j-му классу. Введем следующие характеристические функции, конкретизирующие принадлежность задания j-му классу с учетом перечисленной выше дополнительной информации о характеристиках задания:

$$\chi_j^{\rho}(x,y) = \begin{cases} 0, \textit{ если } \max\limits_{\forall l: y_l = 1}\{\rho_l(x)\} - \rho_j(x) > \delta_{\rho}, \\ 1, \textit{ в противном случае}, \end{cases}$$

$$\chi_j^{\sigma}(x,y) = \begin{cases} 0, \textit{ если } \max\limits_{\forall l: y_l = 1}\{\sigma_l(x)\} - \sigma_j(x) > \delta_{\sigma}, \\ 1, \textit{ в противном случае}, \end{cases}$$

$$\chi_j^{\omega}(x,y) = \begin{cases} 0, \textit{ если } \max\limits_{\forall l: y_l = 1}\{\omega_l(x)\} - \omega_j(x) > \delta_{\omega}, \\ 1, \textit{ в противном случае}, \end{cases}$$

$$\chi_j^{\varphi}(x,y,z) = \begin{cases} 0, \textit{ если } \max\limits_{\forall l: y_l = 1}\{\varphi_l(x,z)\} - \varphi_j(x,z) > \delta_{\varphi}, \\ 1, \textit{ в противном случае.} \end{cases}$$

Следует заметить, что наличие у задания отдельно взятой характеристики с высокой значимостью (как обязательной, так и необязательной) не гарантирует отнесение этого задания к классу, в описании которого присутствует данная характеристика. Классификация осуществляется с учетом агрегированной значимости всех характеристик задания для каждого класса заданий.

Алгоритм классификации задания

Рассмотрим алгоритм классификации задания. На вход алгоритму поступает вектор x, представляющий задание. На выходе получаем вектор y, содержащий информацию о классах, к которым относится данное задание. Алгоритм включает следующие этапы работы:

I. Инициализация элементов вектора y: $y_j = 0$, $\forall j \in \{1, 2, ..., m\}$.

II. Выполнение первичной классификации задания – проверка принадлежности задания каждому из m классов: $y_j = \chi_j(x)$, $\forall j \in \{1, 2, ..., m\}$.

III. Если $y_j = 0$, $\forall j \in \{1,2,\ldots,m\}$, то завершение работы алгоритма (задание не может быть классифицировано).

IV. Иначе – выполнение конкретизации результатов классификации задания, полученной на этапе II.

а) Редукция множества классов, к которым относится задание, с учетом числа характеристик классов: $y_j = \chi_j^{\rho}(x,y)$, $\forall j : y_j = 1$.

б) Редукция множества классов, к которым относится задание, с учетом рангов характеристик $y_j = \chi_j^{\sigma}(x,y)$, $\forall j : y_j = 1$.

в) Редукция множества классов задания, к которым относится

задание, с учетом весов характеристик $y_j = \chi_j^{\omega}(x, y)$, $\forall j : y_j = 1$.

г) Редукция множества классов задания, к которым относится задание, с учетом вычислительной истории данного задания $y_j = \chi_j^{\phi}(x, y, z)$, $\forall j : y_j = 1$.

V. Завершение работы алгоритма (задание классифицировано).

Модель системы классификации потоков заданий

Модель системы классификации потоков заданий строится аналогично модели системы классификации заданий. Пусть имеется конечное множество характеристик потоков заданий $\tilde{H} = \{\tilde{h}_1, \tilde{h}_2, ..., \tilde{h}_k\}$. Каждая характеристика $\tilde{h}_i$ описывается информационной структурой, имеющей следующие компоненты: $\tilde{D}_i$ – область допустимых значений характеристики $\tilde{h}_i$, включающая символ неопределенности θ.

Определим конечное множество классов потоков заданий $\tilde{C} = \{\tilde{c}_1, \tilde{c}_2, ..., \tilde{c}_m\}$. Каждый класс $\tilde{c}_j$ определяется набором обязательных характеристик из $\tilde{H}$. Характеристики классов потоков заданий удобно представить в виде булевой матрицы $\tilde{A}$ размерности $k \times m$, элемент которой $\tilde{a}_{ij} = 1$ означает, что характеристика $\tilde{h}_i$ используется в определении класса $\tilde{c}_j$ и для этого класса имеет конкретизированную область допустимых значений $\tilde{D}_{ij}^* \subseteq \tilde{D}_i \setminus \{\theta\}$. Матрица $\tilde{A}$ должна удовлетворять следующим условиям:

$$\bigvee_{j=1}^{m} \bigwedge_{i=1}^{k} \overline{\tilde{a}_{ij}} = 0, \quad (3)$$

$$\bigvee_{j=1}^{m} \bigvee_{l=1}^{m} \bigwedge_{i=1}^{k} \overline{(\tilde{a}_{ij} + \tilde{a}_{il})} = 0, \quad (4)$$

где $(\tilde{a}_{ij} + \tilde{a}_{il})$ – сложение $\tilde{a}_{ij}$ и $\tilde{a}_{il}$ по mod(2), $j \neq l$.

Формирование множества характеристик $\tilde{H}$ (определение имен, областей допустимых значений, рангов и весов характеристик) и создание на их основе множества классов потоков заданий $\tilde{C}$ (определение для каждого класса характеристик из множества $\tilde{H}$ с указанием их областей значений, допустимых для конкретного класса) выполняется администратором интегрированной кластерной системы.

Поток заданий со всеми своими характеристиками представляется булевым вектором $\tilde{x}$ размерности k. Между индексами элементов вектора $\tilde{x}$ и индексами характеристик из $\tilde{H}$ установлено взаимно однозначное соответствие. Значение i-го элемента вектора $\tilde{x}$ определяется следующим образом:

$$\tilde{x}_i = \begin{cases} 0, \textit{если } \tilde{D}'_i \equiv \{\theta\}, \\ 1, \textit{если } \tilde{D}'_i \subseteq \tilde{D}_i \setminus \{\theta\}, \end{cases}$$

где $\tilde{D}'_i$ – это область значений характеристики $\tilde{h}_i$. Вектор $\tilde{x}$ должен удовлетворять условию $\bigwedge_{i=1}^{k} \overline{\tilde{x}_i} = 0$.

Соответствие требуемых областей допустимых значений характеристик потока заданий областям допустимых значений характеристик j-го класса потоков заданий из $\tilde{C}$ устанавливается с помощью характеристической функции

$$\tilde{\chi}_j(\tilde{x}) = \begin{cases} 0, \textit{если } \exists i : (\tilde{a}_{ij} = 1) \wedge (\tilde{D}'_i \not\subseteq \tilde{D}^*_{ij}), \\ 1, \textit{в противном случае}, \end{cases}$$

где $i \in \{1,2,\ldots,k\}$, $j \in \{1,2,\ldots,m\}$.

Исходя из условий (3) и (4) в результате применения функции $\tilde{\chi}_j$ к каждому j-му классу поток заданий будет либо взаимно однозначно соотнесен с одним из классов потоков заданий из $\tilde{C}$, либо не определен.

Алгоритм классификации потока заданий

Рассмотрим алгоритм классификации потока заданий. На вход алгоритму поступает вектор $\tilde{x}$, представляющий поток заданий. На выходе получаем вектор $\tilde{y}$, содержащий информацию о классах, к одному из которых относится данный поток заданий. Алгоритм включает следующие этапы работы:

I. Инициализация элементов вектора $\tilde{y}$: $\tilde{y}_j = 0$, $\forall j \in \{1,2,\ldots,m\}$.

II. Выполнение классификации потока заданий – проверка принадлежности потока задания каждому из m классов: $\tilde{y}_j = \tilde{\chi}_j(\tilde{x})$, $\forall j \in \{1,2,\ldots,m\}$.

III. Если $\tilde{y}_j = 0$, $\forall j \in \{1,2,\ldots,m\}$, то завершение работы алгоритма (поток заданий не может быть классифицирован). Иначе – $\exists \tilde{y}_j = 1$, $j \in \{1,2,\ldots,m\}$, завершение работы алгоритма (задание классифицировано).

Вычислительные эксперименты

На основе приведенных выше теоретических исследований автором диссертационной работы выполнена программная реализация имитационного прототипа классификатора заданий с использованием языков программирования GPSS и PLUS [34]. Имитационное моделирование работы классификатора заданий было проведено на потоке заданий, соответствующем потоку реальных заданий, выполненных на кластере Blackford с СУПЗ Cleo (таблица 2). В таблице 2 для каждого кластера указано общее число его узлов n_n и ядер n_c, а также усредненный коэффициент увеличения времени выполнения заданий k_{run}.

Таблица 2 – Вычислительные ресурсы РВС

Кластер	n_n	n_c	k_{run}
	единиц		
Blackford	20	160	1
МВС-1000/16	16	32	3
Кластер невыделенных рабочих машин, организованный на базе ПЭВМ научных лабораторий	12	12	2

Всего было обработано 62249 заданий. Список использованных классов задач приведен в таблице 3. Для каждого класса заданий приведены диапазоны допустимых значений ряда его базовых характеристик: число вариантов данных n_v, число требуемых ядер n_c, запрашиваемое время для выполнения задания t.

Таблица 3 – Классы заданий в интегрированной кластерной системе

Класс заданий	Характеристика		
	n_v	n_c	t, мин
	единиц		
univariant sequential small	1	1	1 – 5
univariant sequential medium	1	1	5 – 60
univariant sequential large	1	1	> 60
univariant parallel small	1	≥ 2	1 – 5
univariant parallel medium	1	≥ 2	5 – 60
univariant parallel large	1	≥ 2	> 60

Перед классификацией каждому кластеру (таблица 2) были назначены наиболее подходящие ему классы заданий. Все задания были классифицированы. Для части заданий выявлена потенциальная возможность их перемещения с кластера Blackford на другие кластера с целью оптимизации загрузки вычислительных ресурсов. Перемещение задания возможно только на кластер, удовлетворяющий характеристикам класса задания, и при условии, что время выполнения перемещенного задания останется в диапазоне времени выполнения заданий данного класса. Для заданий классов *univariant sequential large* и *univariant parallel large* учитывалось дополнительное ограничение: перемещенное задание не должно выполняться дольше суток. Число классифицированных заданий n_{job} и число перемещенных заданий n_r для каждого класса приведены в таблице 4.

Полученные результаты показали, что дополнительное применение классификатора заданий в процессе распределения заданий совместно с метапланировщиком заданий интегрированной кластерной системы позволило бы повысить коэффициент полезного использования узлов системы на 18%.

Таблица 4 – Результаты классификации заданий

Класс заданий	n_{job}	n_r
univariant sequential small	10221	4675
univariant sequential medium	7927	5861
univariant sequential large	1211	972
univariant parallel small	33453	492
univariant parallel medium	6125	50
univariant parallel large	3312	95

Реализован действующий прототип классификатора заданий для интегрированной кластерной системы ИДСТУ СО РАН. Схема взаимодействия компонентов классификатора заданий представлена на рис. 1.

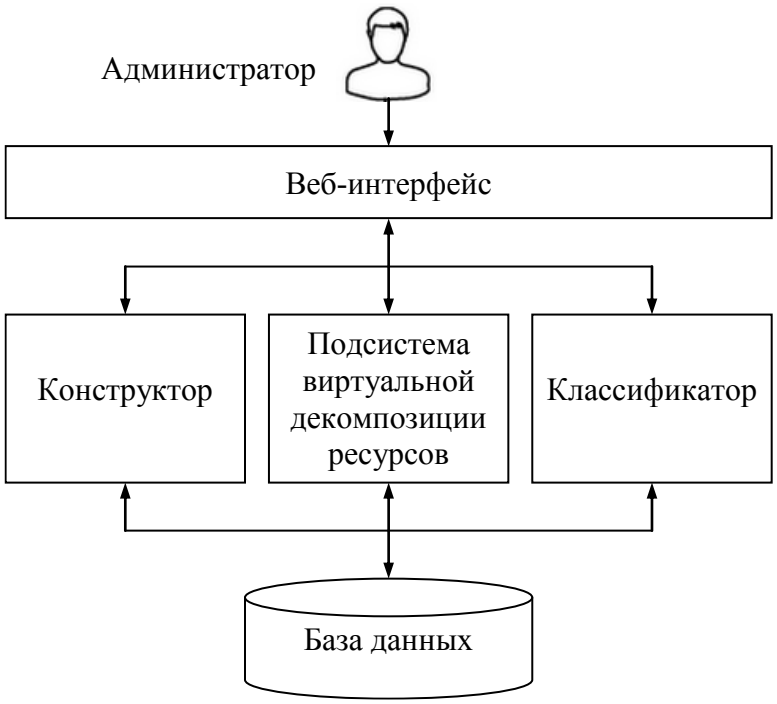

Рис. 1. Схема взаимодействия компонентов классификатора заданий

Подсистема виртуальной декомпозиции ресурсов используется администратором РВС для выделения определенных классам заданий наиболее подходящих для их выполнения вычислительных узлов. На основе такого распределения все задания, относящиеся к конкретному классу, могут быть выполнены только на специально выделенных ресурсах интегрированной кластерной системы. Конструктор обеспечивает возможность создания элементов классификации (характеристик, классов и т.д.) и их занесения в базу данных классификатора заданий. Классификатор обеспечивает проведение классификации задания и определение его принадлежности потоку заданий конкретного приложения. Классы потоков заданий заранее задаются администратором интегрированной кластерной системы. Взаимодействие пользователя с классификатором заданий осуществляется с помощью веб-

интерфейса. Все данные (характеристики заданий, классы и т.д.), необходимые в процессе классификации заданий, хранятся в базе данных классификатора заданий.

Классификатор заданий разработан в дополнение к инструментальным средствам [35] имитационного моделирования процессов функционирования разнородных РВС, в том числе для исследования и оптимизации мультиагентных систем управления распределенными вычислениями [36].

Дополнительно разработаны средства конкретизации заданий для более детальной настройки требований к РВС, содержащихся в заданиях, с целью обеспечения более эффективного планирования и распределения ресурсов планировщиками СУПЗ при обработке этих заданий. Схема взаимодействия данных средств между собой и с СУПЗ представлена на рис. 2.

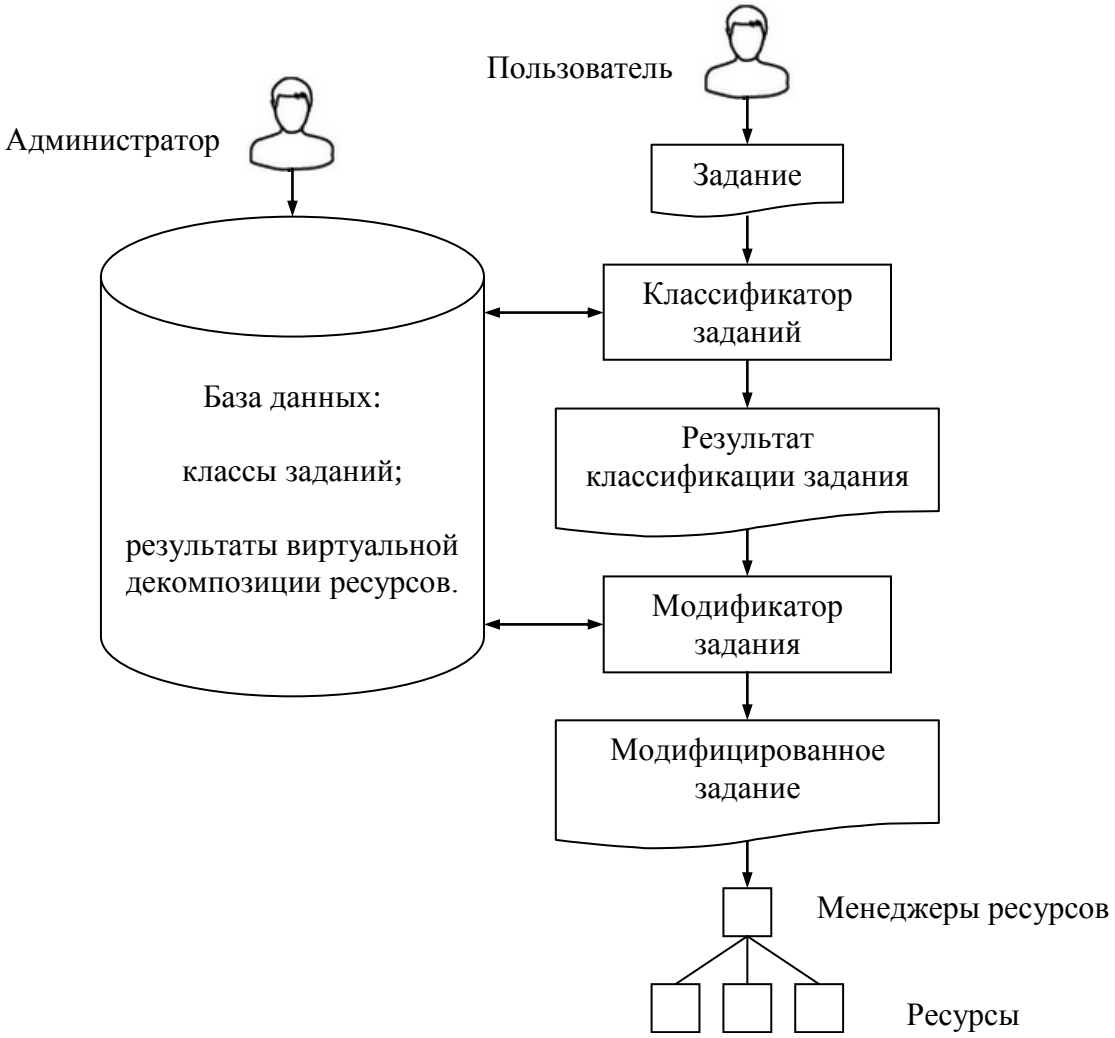

Рис. 2. Схема конкретизации задания

Конкретизатор заданий осуществляет перехват заданий, поступающих в РВС. Конкретизатор передает перехваченное задание классификатору, который выполняет унификацию задания и определение дополнительных его характеристик, извлекаемых из переменных окружения вычислительной среды, конфигурационных параметров СУПЗ или статистических файлов. Затем

классификатор определяет класс задания и передает исходное задание и результат его классификации модификатору задания.

Модификатор задания добавляет в задание дополнительные параметры, определяющие множество ресурсов, соответствующих классу задания, и передает модифицированное задание в СУПЗ. В дальнейшем планировщик СУПЗ будет выбирать ресурсы для выполнения задания только из множества ресурсов, заданного дополнительными параметрами модифицированного задания. Таким образом, конкретизатор заданий может быть использован для управления распределением ресурсов на основе классификации и конкретизации потоков заданий масштабируемых приложений.

С целью более полного исследования рассмотренных выше методов и инструментальных средств классификации и конкретизации потоков заданий масштабируемых приложений проведено имитационное моделирование функционирования РВС с помощью системы GPSS World [34]. Моделируемая система включала 10 кластеров с числом ядер от 6000 до 14000 единиц и 300 пользователей. Общее число ядер составляло 100000 единиц. Кластеры включали гибридные узлы, поддерживающие различные технологии параллельного программирования. При имитации времени выполнения задания на кластерах применялись коэффициенты ускорения счета, значения которых для разных кластеров варьировались от 1 до 1,5 в зависимости от вычислительных характеристик этих кластеров. Моделируемый период времени работы системы – 30 суток. За этот период было обработано 12990 потоков заданий. Эти потоки включали от 1000 до 10000 процессов для параллельных программ или заданий для многовариантных расчетов. В качестве системы управления вычислениями в РВС использовались метапланировщик GridWay. Дисциплина обслуживания очередей заданий – FCFS (First Come, First Served) с приоритетами. В качестве основных наблюдаемых переменных имитационной модели были выбраны следующие показатели: среднее число n_{avg} заданий в очереди кластера, среднее время t_{avg} пребывания задания в очереди кластера и средний коэффициент k_{avg} полезного использования узлов кластеров, среднеквадратическое отклонение σ коэффициент полезного использования узлов кластеров, среднее число рестартов программ n_{rest} и среднее число сбойных задач n_{err}. Приведенные в таблице 5 результаты моделирования показывают, что применение конкретизатора может существенно улучшить все выбранные показатели функционирования РВС.

Результаты модельных экспериментов подтверждены результатами практического использования конкретизатора заданий при решении в РВС ряда крупномасштабных задач булева моделирования [37-40] и складской логистики [41, 42].

Таблица 5 – Результаты использования конкретизатора заданий

Распределение ресурсов	Показатель эффективности системы управления вычислениями					
	n_{avg}, единиц	t_{avg}, с	k_{avg}, %	σ	n_{rest}, единиц	n_{err}, единиц
Без применения конкретизатора заданий	572.398	771.820	0.710	0.006	135	34
С применением конкретизатора заданий	196.142	372.283	0.756	0.004	97	2

Выводы

Представленный подход к классификации заданий позволяет расширить функциональные возможности метапланировщика РВС и осуществлять оптимизацию распределения вычислительных ресурсов среды не только на уровне отдельно взятого задания, но также на уровне потока заданий. Следует отметить, что аналоги реализованным средствам классификации заданий в используемых на практике СУПЗ для вычислительных кластеров авторам неизвестны.

Автор выражают благодарность за помощь в программной реализации классификатора заданий кандидату технических наук Корсукову А.С.

Исследование выполнено при финансовой поддержке РФФИ, проект № 15-29-07955-офи_м.

Литература

1. Коваленко В. Н., Корягин Д. А. Грид: истоки, принципы и перспективы развития // Информационные технологии и вычислительные системы. 2008. № 4. С. 38-50.

2. Advancements in Distributed Computing and Internet Technologies: Trends and Issues / Eds. A.-S.K. Pathan, M. Pathan, H.Y. Lee. IGI Global, 2011. 430 p.

3. Бухановский А. В. Информационно-аналитический обзор по критической технологии «Технологии и программное обеспечение высокопроизводительных распределенных вычислительных систем: технологические тренды, приоритетные направления, перспективы развития, основные организации, оценка рынков, сопоставление российских и мировых результатов». СПб.: НИУ ИТМО. 2013. 31 с.

4. Шамакина А. В. Обзор технологий распределенных вычислений // Вестник Южно-Уральского государственного университета. Серия: Вычислительная математика и информатика. 2014. Т. 3. № 3. С. 51-85.

5. Бетелин В. Б., Кушниренко А. Г., Райко Г. О. Проблемы обеспечения роста производительности отечественных суперЭВМ в период до 2020 года // Информационные технологии и вычислительные системы. 2010. № 9. С. 15-18.

6. Эксафлопные технологии. Концепция по развитию технологии

высокопроизводительных вычислений на базе суперЭВМ эксафлопсного класса (2012-2020 гг.). М.: Росатом, 2011. 112 с.

7. Dongarra J., Bosilca G., Herault T., Rezmerita A. On Scalability for MPI Runtime System // CLUSTER '11. Washington, 2011. Pp. 187-195.

8. Menasce D. A., Casalicchio E. QoS in grid computing // IEEE Internet Computing. 2004. Vol. 8. No. 4. Pp. 85-87.

9. Chunlin L., Layuan L. A distributed multiple dimensional QoS constrained resource scheduling optimization policy in computational grid // Journal of Computer and System Sciences. 2006. Vol. 72. No. 4. Pp. 706-726.

10. Perez-Gonzalez P., Framiñan J. A Common Framework and Taxonomy for Multicriteria Scheduling Problems with Interfering and Competing Jobs: Multi-agent Scheduling Problems // European Journal of Operational Research. 2014. Vol. 235. No. 1. Pp. 1-16.

11. Muthuvelu N., Chai I., Chikkannan E., Buyya R. QoS-based Task Group Deployment on Grid by Learning the Performance Data // Journal of Grid Computing. 2014. Vol. 12. No. 3. Pp. 465-483.

12. Иванов В. В. Методы вычислений на ЭВМ: Справочное пособие. Киев: Наукова думка, 1986. 584 с.

13. Лорьер Ж.-Л. Системы искусственного интеллекта. М.: Мир. 1991. 568 с.

14. Воеводин В. В., Воеводин Вл. В. Параллельные вычисления. СПб.: БХВ-Петербург, 2002. 608 с.

15. Herrera J., Huedo E., Montero R., Llorente I. Porting of Scientific Applications to Grid Computing on GridWay // Scientific Programming. 2005. Vol. 13. No. 4. Pp. 317–331.

16. Henderson R. Job scheduling under the portable batch system // Job scheduling strategies for parallel processing. Springer. 1995. Pp. 279-294.

17. Litzkow M., Livny M., Mutka M. Condor – A Hunter of Idle Workstations // In 8th International Conference of Distributed Computing Systems (ICDCS). IEEE CS Press, Los Alamitos, CA, USA, 1988. Pp. 104-111.

18. Бахвалов Н. С. Численные методы. М.: Наука, 1973. Т. 1. 632 с.

19. Марчук Г. И. Методы вычислительной математики – М.: Наука, 1980. 536 с.

20. Человек и вычислительная техника / Под ред. В.М. Глушкова. Киев: Наукова думка, 1971. 294 с.

21. Козлов Н. И. Организация вычислительных работ. М.: Наука, 1981. 240 с.

22. Ершов А. П., Ильин В. П. Пакеты программ как методология решения прикладных задач // Пакеты прикладных программ: Проблемы и перспективы. 1982. С. 4-18.

23. Тыугу Э. Х. Концептуальное программирование. М.: Наука, 1984. 256 с.

24. Опарин Г. А. Автоматизация разработки и применения пакетов программ для исследования динамики сложных управляемых систем. Автореф. дис. докт. техн. наук: 05.13.11. Иркутск: Изд-во ИДСТУ СО РАН, 1998. 40 с.

25. Voevodin V. V. The solution of large problems in distributed computational media // Automation and Remote Control. 2007. Vol. 68. No. 5. Pp. 773-786.

26. Bandman O. L. Fine-graned parallelism in computational mathematics // Programming and computer software. 2001. No. 1. Pp. 170-182.

27. Бычков И. В., Корсуков А. С., Опарин Г. А., Феоктистов А. Г. Инструментальный комплекс для организации гетерогенных распределенных вычислительных сред // Информационные технологии и вычислительные системы. 2010. № 1. С. 45-54.

28. Топорков В. В. Модели распределенных вычислений. М.: Физматлит, 2004. 320 с.

29. Berman F., Wolski R., Casanova H., Cirne W., Dail H., Faerman M., Figueira S., Hayes J., Obertelli G., Schopf J., Shao G., Smallen S., Spring N., Su A., Zagorodnov D. Adaptive Computing on the Grid Using AppLeS // IEEE Transactions on Parallel and Distributed Systems. 2003. Vol. 14. No. 4. Pp. 369–382.

30. Коваленко В. Н. Комплексное программное обеспечение грида вычислительного типа // Препринты ИПМ им. М.В. Келдыша РАН. 2007. № 10. 39 с.

31. European Middleware Initiative [Электронный ресурс]. URL: http://www.eu-emi.eu/ (дата обращения: 12.10.2015).

32. Enabling Grids for E-sciencE [Электронный ресурс]. URL: http://eu-egee-org.web.cern.ch/eu-egee-org/index.html (дата обращения: 12.10.2015).

33. Worldwide LHC Computing Grid [Электронный ресурс]. URL: http://wlcg.web.cern.ch/ (дата обращения: 12.10.2015).

34. Боев В. Д. Моделирование систем. Инструментальные средства GPSS World. СПб.: БХВ-Петербург, 2004. 368 с.

35. Дядькин Ю.А. Инструментальное средство моделирования разнородной распределенной вычислительной среды // Современные проблемы науки и образования. 2015. № 2. URL: http://www.science-education.ru/122-21388 (дата обращения: 21.08.2015).

36. Костромин Р. О. Модели, методы и средства управления вычислениями в интегрированной кластерной системе // Фундаментальные исследования. 2015. № 6-1. С. 35-38.

37. Опарин Г. А., Богданова В. Г. РЕБУС – интеллектуальный решатель комбинаторных задач в булевых ограничениях // Вестник Новосибирского государственного университета. Серия: Информационные технологии. 2008. Т. 6. Вып. 1. С. 61-69.

38. Опарин Г. А., Богданова В. Г., Макеева Н. Г. Инструментальная среда параллельного решения систем булевых уравнений // Современные технологии. Системный анализ. Моделирование. 2009. № 3. С. 62-68.

39. Богданова В. Г., Горский С. А. Технология параллельного решения систем булевых уравнений на вычислительном кластере // Современные технологии. Системный анализ. Моделирование. 2013. № 1. С. 54-60.

40. Богданова В. Г., Горский С. А., Пашинин А. А. Сервис-ориентированные инструментальные средства для решения задач булевой

выполнимости // Фундаментальные исследования. 2015. № 2-6. С. 1151-1156.

41. Горский С. А., Башарина О. Ю. Моделирование складской логистики: разработка и комплексирование в Orlando Tools // Программные продукты и системы. 2012. № 1. С. 89-91.

42. Башарина О. Ю., Носков С. И. Решение задач складской логистики на основе применения методологии системного анализа // Современные технологии. Системный анализ. Моделирование. 2014. № 1. С. 70-75.

References

1. Kovalenko V. N., Koryagin D.A. Д.А. Grid: istoki, printsipy I perspektivy razvitiya [Grid: Sources, Principles and Development Prospects]. *Informatsionnye tekhnologii i vychislitelnye sistemy*, 2008, no. 4, pp. 38-50 (in Russian).

2. Pathan A.-S.K., Pathan M., Lee H.Y. *Advancements in Distributed Computing and Internet Technologies: Trends and Issues*. IGI Global, 2011. 430 p.

3. Bukhanovskij A. V. *Informatsionno-analiticheskij obzor po kriticheskoj tekhnologii "Tekhnologii I programnoe obespechenie vysokoproizvoditel'nykh raspredelennykh vychislitel'nykh sistem: tekhnologicheskie trendy, prioritetnye napravleniya, perspektivy razvitiya, osnovnye organizatsii, otsenka rynkov, sopostavlenie rossijskikh I mirovykh rezultatov"* [Information-Analytical Review of the Critical Technology "Technologies and Software High-Performance Distributed Computing Systems: Technological Trends, Priorities, Prospects, Key Organizations, Markets Evaluation, Comparison of Russian and International Results"]. Saint-Petersburg, ITMO National Research University Publ., 2013, 31 p. (in Russian).

4. Shamakina A. V. Survay on Distributed Computing Tecnologies. *Bulletin of the South Ural State University, Series "Computational Mathematics and Software Engineering"*, 2014, vol. 3, no. 3, pp. 51-85 (in Russian).

5. Betelin V. B., Kushnirenko A. G., Rajko G .O. Problemy obespecheniya rosta proizvoditel'nosti otechestvennykh superEVM v period do 2020 goda [Problems of Productivity Growth for Domestic Supercomputers in the Period up to 2020]. *Informatsionnye tekhnologii i vychislitelnye sistemy*, 2010, no. 9, pp. 15-18 (in Russian).

6. *Eksaflopsnye tekhnologii. Kontseptsiya po razvitiyu tekhnologii vysokoproizvoditel'nykh vychislenij na baze superEVM eksaflopcnogo klassa (2012-2020 gg.)* [Exascale Technology. The Concept for the Development of High-Performance Computing Technology Based on Exascale Supercomputers (2012-2020)]. Moscow, Rosatom Publ., 2011. 112 p. (in Russian).

7. Dongarra J., Bosilca G., Herault T., Rezmerita A. On Scalability for MPI Runtime System. *Proc. of the IEEE International Conference on Cluster Computing*, Austin, 2011, pp. 187-195.

8. Menasce D. A., Casalicchio E. QoS in Grid Computing. *IEEE Internet Computing*, 2004, vol. 8, no. 4, pp. 85-87.

9. Chunlin L., Layuan L. A Distributed Multiple Dimensional QoS Constrained Resource Scheduling Optimization Policy in Computational Grid. *Journal of Computer and System Sciences*, 2006, vol. 72, no. 4, pp. 706-726.

10. Perez-Gonzalez P., Framiñan J. A Common Framework and Taxonomy for

Multicriteria Scheduling Problems with Interfering and Competing Jobs: Multi-agent Scheduling Problems. *European Journal of Operational Research*, 2014, vol. 235, no. 1, pp. 1-16.

11. Muthuvelu N., Chai I., Chikkannan E., Buyya R. QoS-based Task Group Deployment on Grid by Learning the Performance Data. *Journal of Grid Computing*, 2014, vol. 12, no. 3, pp. 465-483.

12. Ivanov V. V. *Metody vychislenij na EVM: spavochnoe posobie* [Methods of Computer Calculations: A Reference Guide]. Kiev, Naukova dumka Publ., 1986, 584 p. (in Russian).

13. Lauriere J.-L. *Intelligence Artificielle. Resolusion de problemes par l'Homme et la machine*. Paris, Eyrolles, 1987, 485 p. (in Franch).

14. Voevodin V. V., Voevodin Vl.V. *Parallel'nye vychisleniya* [Parallel Computing]. Saint-Petersburg, BHV-Petersburg Publ., 2002, 608 p. (in Russian).

15. Herrera J., Huedo E., Montero R., Llorente I. Porting of Scientific Applications to Grid Computing on GridWay. *Scientific Programming*, 2005, vol. 13, no. 4, pp. 317–331.

16. Henderson R. Job Scheduling under the Portable Batch System. *Proc. of the IPPS Workshop Job Scheduling Strategies for Parallel Processing*. Springer, Santa Barbara, 1995, pp. 279-294.

17. Litzkow M., Livny M., Mutka M. Condor – A Hunter of Idle Workstations. *In Proc. of 8th International Conference of Distributed Computing Systems (ICDCS)*, IEEE CS Press, Los Alamitos, 1988, pp. 104-111.

18. Bakhvalov N. S. *Chislennye metody* [Computational Methods]. Moscow, Nauka Publ., 1973, vol. 1, 632 p. (in Russian).

19. Marchuk G. I. *Metody vychislitel'noj matematiki* [Methods of Computational Mathematics]. Moscow, Nauka Publ., 1980, 536 p. (in Russian).

20. *Chelovek I vychislitel'naya tekhnika* [Human and Computer Engineering]. Ed. V.M. Glushkov. Kiev, Naukova dumka Publ., 1971, 294 p. (in Russian).

21. Kozlov N. I. *Organizatsiya vychislitel'nykh rabot* [Computing Works Organization]. Moscow, Nauka Publ., 1981, 240 p. (in Russian).

22. Ershov A. P., Ilyin V. P. *Pakety program kak metodologiya resheniya prikladnykh zadach* [Software packages as methodology for solving applied problems]. *Pakety prikladnykh program: Problemy I perspektivy* [Applied Software Packages: Problems and Prospects], 1982, pp. 4-18 (in Russian).

23. Tyugu E. *Kontseptual'noe programmirovanie* [Conceptual Programming]. Moscow, Nauka Publ., 1984, 256 p. (in Russian).

24. Oparin G. A. *Avtomatizatsiya razrabotki I primeneniya paketov programm dlya issledovaniya dinamiki slozhnykh upravlyaemykh system*. Dis. dokt. tehn. nauk [Automating the Development and Application of Software Packages for the Study of Dynamics of Complex Control Systems. Extended Abstract of Dr. habil. Thesis]. Irkutsk, ISDCT SB RAS Publ., 1998, 40 p. (in Russian).

25. Voevodin V. V. The Solution of Large Problems in Distributed Computational Media. *Automation and Remote Control*, 2007, vol. 68, no. 5, pp. 773-786.

26. Bandman O. L. Fine-Graned Parallelism in Computational Mathematics.

Programming and Computer Software, 2001, no. 1, pp. 170-182.

27. Bychkov I. V., Korsukov A. S., Oparin G. A., Feoktistov A. G. Instrumental'nyj kompleks dlya organizatsii geterogennykh raspredelennykh vychislitel'nykh sred Инструментальный комплекс для организации гетерогенных распределенных вычислительных сред [The Toolkit for Development of Heterogeneous Distributed Computing Environments]. *Informatsionnye tekhnologii i vychislitrl'nye sistemy*, 2010, no. 1, pp. 45-54 (in Russian).

28. Toporkov V. V. *Modeli raspredelennykh vychislenij* [Models of Distributed Computing]. Moscow, Fizmatlit Publ., 2004, 320 p. (in Russian).

29. Berman F., Wolski R., Casanova H., Cirne W., Dail H., Faerman M., Figueira S., Hayes J., Obertelli G., Schopf J., Shao G., Smallen S., Spring N., Su A., Zagorodnov D. Adaptive Computing on the Grid Using AppLeS. *IEEE Transactions on Parallel and Distributed Systems*, 2003, vol. 14, no. 4, pp. 369-382.

30. Kovalenko V. N. *Kompleksnoe programmnoe obespechenie grida vychislitel'nogo tipa* [Complex Software for Grid Computing]. Moscow, Preprints of the Keldysh Institute of Applied Mathematics RAS, 2007, no. 10, 39 p. (in Russian).

31. *European Middleware Initiative.* Available at: http://www.eu-emi.eu/ (accessed 12 October 2015).

32. *Enabling Grids for E-sciencE.* Available at: http://eu-egee-org.web.cern.ch/eu-egee-org/index.html (accessed 12 October 2015).

33. *Worldwide LHC Computing Grid.* Available at: http://wlcg.web.cern.ch/ (accessed 12 October 2015).

34. Boev V. D. *Modelirovanie sistem. Instrumentalnyie sredstva GPSS World* [Modeling Systems. Tools of the GPSS World]. Saint-Petersburg, BHV-Petersburg Publ., 2004, 368 p. (in Russian).

35. Dyadkin Y. A. The Toolkit for Modeling of Heterogeneous Distributed Computing Environments. *Modern Problems of Science and Education*, 2015, no. 2. Available at: URL: http://www.science-education.ru/122-21388 (accessed 11 October 2015) (in Russian).

36. Kostromin R.O. Models, Methods and Means Calculations Control in Integrated Cluster System. *Fundamental Research*, 2015, no. 6-1, pp. 35-38 (in Russian).

37. Oparin G. A., Bogdanova V. G. REBUS – Intellctual Solver for Combinatorial Problems in Boolean Constraints. *Vestnik Novosibirsk State University, Series "Information Technologies"*, 2008, vol. 6, no. 1, pp. 61-69 (in Russian).

38. Oparin G. A., Bogdanova V. G., Makeeva N. G. Toolkit for the Parallel Solving of Systems Boolean Equations. *Modern Technologies. System Analysis. Modeling*, 2009, no. 3, pp. 62-68 (in Russian).

39. Bogdanova V. G., Gorsky S. A. The Technology of Parallel Solution of Nonlinear Systems of Boolean Equations on a Computer Cluster. *Modern Technologies. System Analysis. Modeling*, 2013, no. 1, pp. 54-60 (in Russian).

40. Bogdanova V. G., Gorsky S. A., Pashinin A. A. Service-Oriented Tools for Solving of Boolean Satisfiability Problem. *Fundamental Research*, 2015, no. 2-6,

pp. 1151-1156 (in Russian).

41. Gorsky S. A., Basharina O. Y. Modeling of Warehousing: Development and Integration in Orlando Tools. *Programmnye produkty i sistemy*, 2012, no. 1, pp. 89-91 (in Russian).

42. Basharina O. Y., Noskov S. I. Solving of Warehouse Logistic Tasks on Basis Methodology System Analysis. *Modern Technologies. System Analysis. Modeling*, 2014, no. 1, pp. 70-75 (in Russian).

Статья поступила 27 октября 2015 г.

Информация об авторе

Феоктистов Александр Геннадьевич – кандидат технических наук, доцент. Старший научный сотрудник лаборатории параллельных и распределенных вычислительных систем. Институт динамики систем и теории управления им. В.М. Матросова СО РАН. Область научных интересов: организация проблемно-ориентированных распределенных вычислений; имитационное моделирование; мультиагентные технологии. E-mail: agf65@yandex.ru

Адрес: Россия, 664033, г. Иркутск, ул. Лермонтова, д. 134.

The Methodology of Conceptualizing and Classifying Job Flows of the Scalable Applications in a Heterogeneous Distributed Computing Environment

A. G. Feoktistov

Purpose. *Modern distributed computing environments are characterized by the high competition of users for shared resources of these environments during performing of the users' jobs. The problem of the optimal allocation of resources for jobs actualizes the solving tasks of identifying, describing, classifying and using of the information about characteristics of resources and jobs. Currently, the traditional resource management systems for distributed computing environments are unable to effectively solving the problems listed above. The purpose of the present paper is the development of the methodology of conceptualizing and classifying job flows of scalable applications in distributed computing environments. This methodology, firstly, includes the specialized models and methods of representing and using of knowledge about resources and jobs, and, secondly, provides the technology to the practical usage of these models and methods.* ***Methods.*** *Attributive description of classified objects using numerical and non-numerical characteristics are used in a process of classifying of jobs and their flows. In the case of classifying jobs, computational characteristics of these jobs are used as attributes, in the job flows classification – structural and behavioral characteristics of these flows. A recognition of properties for jobs and their flows is performed by means a set of specialized characteristic functions. The detailed setting requirements contained in the classified jobs is carried out on the basis of program specialization methods.* ***Novelty.*** *As is known to the author, analogs of the means for classifying jobs and their flows suggested in the paper are not used by resource management systems for distributed computing environments in practice.* ***Results.*** *The usage of the means for classifying jobs and their flows suggested in the paper as superstructure to a resource management system allows significantly improve the results of the allocation of these resources.* ***Practical relevance.*** *The simulation results show that the usage of the developed systems for classifying jobs and their flows allows to optimize resources allocation and provides a significant increase of some important parameters of the efficiency for functioning distributed computing environment. The practical usage of these systems confirms the received model results.*

Key words: *distributed computing environment, scalable applications, job flows, conceptualizing and classifying.*

Information about Author

Aleksandr Gennadyevich Feoktistov – Ph.D. of Engineering Sciences, Associate Professor. Senior Research Officer of the Laboratory of Parallel and Distributed Computing Systems. Matrosov Institute for System Dynamics and Control Theory of SB RAS. Field of research: organization of problem-oriented distributed computing, simulation modeling; multiagent technologies. E-mail: agf65@yandex.ru

Address: Russia, 664033, Irkutsk, Lermontov St., 134.