УДК 681.3.06+519.68

И. В. Бычков, Г. А. Опарин, А. Г. Феоктистов, А. С. Корсуков

Институт динамики систем и теории управления СО РАН
ул. Лермонтова, 134, Иркутск, 664033, Россия
E-mail: alexask@icc.ru

ДЕЦЕНТРАЛИЗОВАННОЕ УПРАВЛЕНИЕ ПОТОКАМИ ЗАДАНИЙ В ИНТЕГРИРОВАННОЙ КЛАСТЕРНОЙ СИСТЕМЕ *

В работе рассматривается подход к управлению потоками заданий в интегрированной кластерной системе, базирующийся на применении методологии концептуального программирования, технологии мультиагентных систем и теории очередей.

Ключевые слова: распределенные вычисления, интегрированные кластерные системы, мультиагентные системы, управления потоками заданий.

Введение

Современное развитие информационно-вычислительных и сетевых технологий, программного обеспечения и аппаратных средств позволяет организовывать сложные географически распределенные вычислительные системы для поддержки проведения массовых ресурсоемких фундаментальных и прикладных исследований. В частности, в научном сообществе ведутся активные работы по созданию и использованию Grid-систем различного назначения [1], среди которых важное место отводится вычислительным Grid-системам. Высокая интенсивность потоков вычислений в подобных Grid-системах обуславливает необходимость эффективного управления этими потоками. Известно много различных моделей, методов и алгоритмов, используемых для решения данной проблемы [2; 3]. С этой целью разрабатываются и применяются агентно-ориентированные подходы к управлению вычислениями [4], а также подходы, основанные на экономических методах распределения вычислительных ресурсов [5]. Существование большого числа стратегий управления распределенными вычислениями, применяемых в разнородных Grid-системах и ориентированных на решение разнотипных задач, зачастую не позволяет провести для этих стратегий достаточно обоснованное сравнение по причине отсутствия как универсальных критериев сравнительного анализа, так и однородных средств получения качественных или количественных оценок этих критериев [2].

Как правило, Grid-системы обладают рядом свойств, существенно усложняющих унификацию процесса управления вычислениями. К свойствам такого рода, например, относятся: организационно-функциональная разнородность, динамичность и неполнота описания интегрируемых в Grid-системах ресурсов; разнообразие спектра задач, решаемых с помощью этих ресурсов; наличие различных категорий пользователей, преследующих свои цели и задачи эксплуатации вычислительной системы. В этом случае достаточно сложно сформировать единый объективный критерий управления системой, который отвечал бы различным субъективным целям и задачам использования ее ресурсов. Возникает необходимость определения таких критериев функционирования вычислительной системы, которые бы позволили отразить изменяющиеся во времени интересы различных субъектов системы и осуществить выбор удовлетворительного уровня обслуживания поступающих в систему заданий.

* Работа выполнена при частичной поддержке программы Президиума РАН № 13 (проект СО РАН № 3).

С этой точки зрения наиболее целесообразным видится применение модели с различными уровнями обслуживания заданий [6], выбор которых регулируется соотношением спроса и предложения по ресурсам. Использование такой модели сводит задачу управления Grid-системой к определению приемлемых интервалов изменения значений для характеристик назначенного уровня обслуживания задания. Однако такой подход требует, во-первых, интенсивного и сложно реализуемого на практике взаимодействия администраторов ресурсов с пользователями, решающими свои задачи с помощью этих ресурсов, и, во-вторых, детального учета специфики решаемых задач и вычислительных характеристик используемых ресурсов. Как следствие, возникает необходимость разработки программных средств, которые бы позволили всесторонне описать аспекты прохождения потоков заданий в Grid-системе и обеспечили управление потоками заданий на основе прав и обязанностей, делегированных этим средствам администраторами и пользователями Grid-системы. Для реализации таких программных средств наиболее предпочтительным является агентно-ориентированный подход [7].

В данной работе рассматривается подход к управлению распределенными вычислениями, базирующийся на применении методологии концептуального программирования, технологии мультиагентных систем и теории очередей.

Интегрированная кластерная система

Одной из разновидностей Grid-систем является интегрированная кластерная система – распределенная вычислительная среда, предназначенная для решения фундаментальных и прикладных вычислительных задач и характеризующаяся следующими особенностями:

- в качестве узлов системы выступают вычислительные кластеры;
- кластеры организуются на базе как выделенных, так и невыделенных вычислительных машин и, следовательно, существенно различаются по степени надежности своих вычислительных ресурсов;
- на разных уровнях интеграции системы существуют различные категории пользователей, в том числе пользователи, нуждающиеся в высокоуровневых средствах организации вычислительного процесса решения задачи;
- вычислительные кластеры используются пользователями системы совместно с владельцами этих кластеров;
- задание пользователя представляет собой спецификацию процесса решения задач, содержащую информацию о требуемых вычислительных ресурсах, исполняемых прикладных программах, входных / выходных данных, а также другие необходимые сведения;
- множество заданий пользователей рассматривается с точки зрения теории очередей и представляется в виде совокупности потоков заданий с приоритетами;
- поток заданий характеризуется следующими свойствами: динамичностью, стохастичностью, неоднородностью, отсутствием обратной связи, неординарностью, стационарностью;
- свободных ресурсов системы недостаточно для одновременного обслуживания всех заданий, находящихся в очередях;
- в рамках системы функционируют распределенные проблемно-ориентированные программные комплексы, размещенные в ее узлах;
- в общем случае в системе имеется программно-аппаратная вычислительная избыточность (программа может быть размещена и выполнена в разных узлах системы, а одни и те же вычисления могут быть произведены с помощью различных программ);
- в системе нет единой политики администрирования вычислительных кластеров, на кластерах применяются различные принципы и механизмы обработки потоков заданий различных типов.

Далее рассматриваются модели и методы представления знаний о системах такого рода, разработанные в рамках предложенного подхода к управлению распределенными вычислениями. Для поддержки общности рассуждений за элементарную вычислительную единицу системы принято вычислительное устройство (ВУ), которое символизирует вычислительный модуль кластера выделенных машин или отдельный персональный компьютер кластера не-

выделенных машин. Вычислительный узел системы (кластер) включает множество ВУ. Один из ВУ системы будем считать эталонным и обозначать $ВУ_e$. В общем случае узел системы может быть представлен отдельной однопроцессорной или многопроцессорной машиной. В свою очередь процессор этой машины может быть как одноядерным, так и многоядерным.

Агрегированная концептуальная модель системы

Авторами разработана оригинальная объектная модель (рис. 1), которая обеспечивает взаимосвязанное представление проблемно-ориентированного, программно-аппаратного, имитационного и управляющего слоев знаний об интегрированной кластерной системе, а также всестороннее исследование необходимых свойств (эффективность, надежность и др.) проектируемых для этой системы прикладных программных комплексов различного назначения. Данная модель является основой для двухуровневой системы интеллектуальных агентов, используемой для управления распределенными вычислениями.

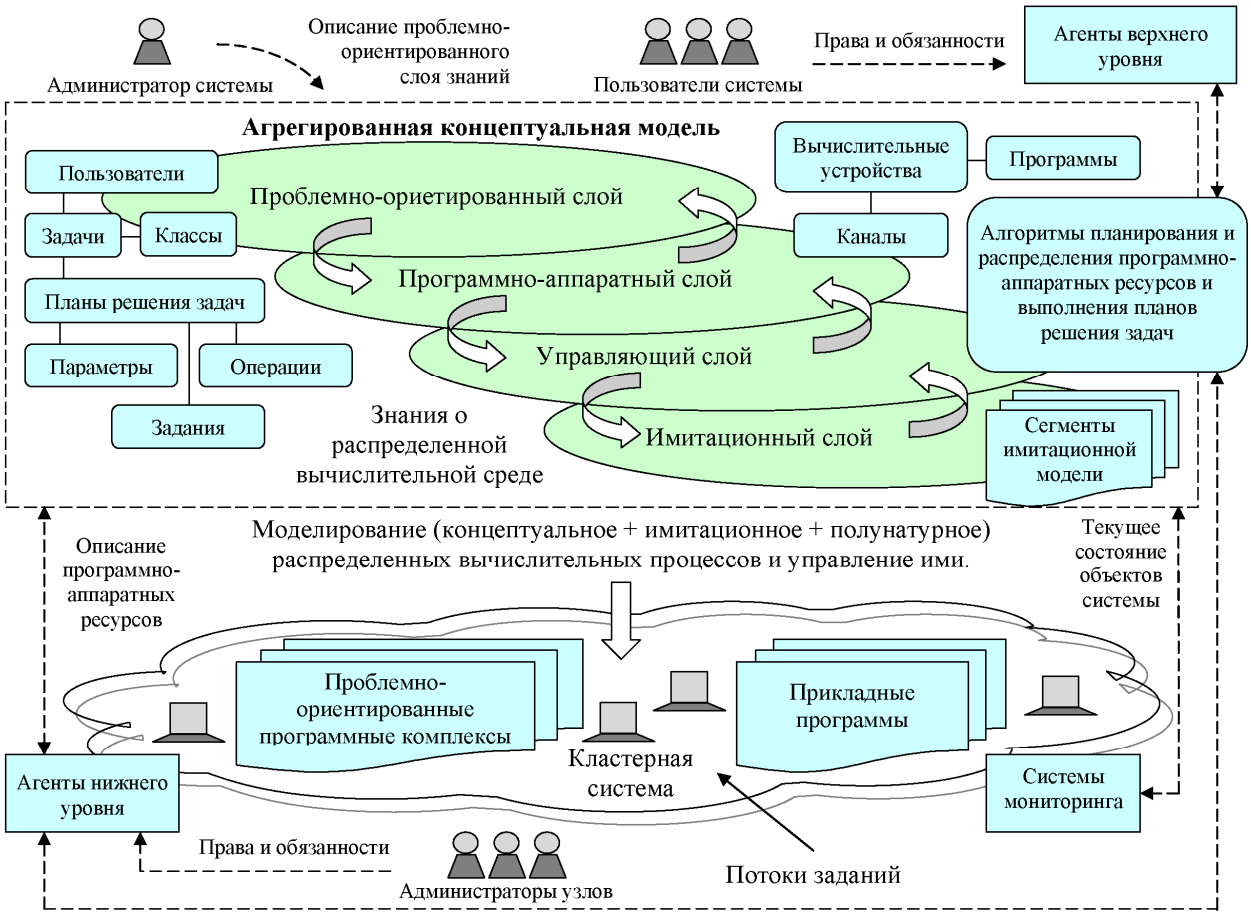


Рис. 1. Агрегированная концептуальная модель кластерной системы

Формально данную модель можно представить в виде структуры $M =< O, Con, R >$, где O – множество классов объектов; Con – множество ограничений целостности, накладываемых на объекты из O; R – множество отношений (связей) между классами объектов из O вида $r_k(o_i(n_i^{\min}, n_i^{\max}) / con_l : o_j(n_j^{\min}, n_j^{\max}) / con_m)$, где $o_i, o_j \in O$ – классы объектов, связанные бинарным отношением r_k; $n_i^{\min}, n_i^{\max}$ – числа, которые означают соответственно минимальное и максимальное число элементов o_i, связанных с элементом o_j; $n_j^{\min}, n_j^{\max}$ – числа, кото-

рые означают соответственно минимальное и максимальное число элементов o_j, связанных с элементом o_i; $con_l, con_m \in Con$ – ограничения целостности, накладываемые на элементы o_i и o_j, участвующие в отношении r_k.

В качестве объектов модели выступают пользователи, задачи и задания пользователей, параметры и операции, вычислительные узлы и устройства, коммутирующие устройства, каналы сети передачи данных, алгоритмы, программы, вычислительные процессы и другие сущности предметной области. Разные слои концептуальной модели содержат свои классы объектов. Для каждого класса объекта определены его атрибуты (характеристики). Набор базовых операций с объектами включает: операции создания и удаления типов данных, допустимых в модели, классов объектов, характеристик объектов, экземпляров объектов; операции установления и разрыва связей между объектами; операции проверки полноты (завершенности) и целостности связей между объектами; операции поиска и выбора информации об объектах.

Важной особенностью рассматриваемой модели, во многом обеспечивающей в дальнейшем эффективность управления распределенными вычислениями, является обеспечение возможности выявления и учета специфики решаемых в системе задач. В числе важных характеристик задач можно выделить: 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 подзадач; вид параллелизма алгоритма решения задачи (мелкозернистый, крупноблочный и смешанный параллелизм), степень ресурсоемкости вычислений (малые, средние и большие задачи, как вычислительного характера, так и обработки данных); необходимость управления процессом вычислений в пакетном или интерактивном режимах; срочность вычислений и другие особенности процесса решения задач. Выбор перечисленных характеристик основан на исследованиях, затрагивающих вопросы классификации вычислительных задач и способов их постановки [8–10]. Задания классифицируются в соответствии с характеристиками решаемых задач. Для каждого уже определенного класса заданий могут быть образованы подклассы.

Функцию построения модели M исполняет администратор системы с помощью инструментальных средств [11], предназначенных для описания и имитационного моделирования распределенных вычислительных сред. В процессе работы системы занесение знаний в M осуществляется несколькими путями (см. рис. 1). Сведения о пользователях, задачах, заданиях, программах и данных поступают через административную подсистему интегрированной кластерной системы. Детальное описание программно-аппаратных ресурсов выполняется интеллектуальными агентами, представляющими узлы системы. Текущее состояние объектов системы отслеживается с помощью различных средств мониторинга вычислительной среды [12].

Характеристики вычислительных устройств

В качестве основных компонентов ВУ выделим процессор, оперативную и дисковую память, видеокарту и коммуникационные составляющие ВУ. Производительность компонентов $ВУ_i$ определяется множеством из n характеристик $C_i^* = \{c_{i1}, c_{i2}, ..., c_{in}\}$, $1 \le i \le nf$, nf – число ВУ системы. В число таких характеристик входят количественные показатели элементной базы (число процессоров, ядер, дисков и других комплектующих элементов), показатели выполнения процессором операций с целыми и вещественными числами, латентность и пропускная способность кэш-памяти процессора, объем и пропускная способность оперативной памяти, время произвольного доступа, скорость чтения и записи данных при работе с дисковой памятью, а также ряд других показателей. Набор характеристик является общим для всех ВУ. Производительность работы каждого *j*-го компонента $ВУ_i$ определяется подмножеством характеристик $C_{ij} \subset C_i^*$, удовлетворяющим условиям

$$1 \le \left|C_{ij}\right| < n,\ C_{ik} \cap C_{il} = \varnothing,\ \forall k,l \in \{1,2,...,m\},$$

где $\left|C_{ij}\right|$ – число характеристик j-го компонента ВУ$_i$; m – число компонентов ВУ$_i$.

Характеристики ВУ, так же как и характеристики других аппаратных средств, определяются при описании программно-аппаратного слоя знаний о системе.

Отдельно взятые вычислительные кластеры выделенных рабочих станций являются, как правило, высоконадежными вычислительными комплексами. Однако в рассматриваемую систему могут быть также интегрированы кластеры других видов. Например, кластеры, сформированные на базе невыделенных рабочих станций компьютерных классов различных вузов или разнородных персональных компьютеров научных организаций. В этом случае особенно важной становится проблема обеспечения надежности вычислительной системы, узлы которой обладают различной степенью доступности и отказоустойчивости.

Для решения этой проблемы необходимо определение показателей надежности ВУ. Исходя из теории надежности вычислительных систем [13] выбирается, как правило, небольшое число таких показателей. Эти показатели должны отражать специфику функционирования ВУ, обеспечивать возможности нетрудоемкого вычисления и проверки их значений.

Пусть в системе существуют простейшие потоки отказов различных видов. Отказы каждого вида возникают с некоторой постоянной интенсивностью $\lambda = \text{const}$. Будем считать, что ВУ$_i$ представляет собой восстанавливаемое устройство без резервирования, а время его работы до отказа имеет экспоненциальное распределение

$$F_i(t) = P_i(T_i < t) = 1 - e^{-\lambda_i t},\ \lambda_i > 0,$$

где t – время возникновения отказа; T_i – случайная величина наработки до отказа ВУ$_i$. Функция надежности ВУ$_i$ в этом случае имеет вид

$$G_i(t) = 1 - F_i(t) = e^{-\lambda_i t}.$$

Стационарный коэффициент готовности [13], характеризующий ВУ$_i$ в произвольный момент времени t_0 по статистическим результатам его работы, определяется соотношением

$$K_i^{\text{stat}} = \frac{t_i^w}{t_i^w + t_i^r},\ t_i^w > 0,\ t_i^r > 0,$$

где t_i^w – суммарное время работы ВУ$_i$; t_i^r – суммарное время восстановления ВУ$_i$.

Тогда коэффициент интервальной готовности, показывающий вероятность того, что ВУ$_i$ окажется работоспособным в некоторый момент времени t_0 и проработает безотказно в течение интервала времени длительностью t будет определен как

$$K_i^{\text{in}}(t_0, t) = K_i^{\text{stat}} G_i(t).$$

Дополнительно введем показатель доступности ВУ$_i$

$$K_i^{\text{avail}} = \begin{cases} 1, \text{если ВУ}_i \text{ доступно}, \\ 0, \text{если ВУ}_i \text{ недоступно}. \end{cases}$$

Оценка производительности вычислительных устройств

Каждое задание связано с выполнением одной или нескольких программ из множества $Prog = \{prog_1, prog_2, ...\}$, $Prog \in O$. Информация о программах накапливается в системе по мере поступления заданий. В нескольких заданиях может быть указана одна и та же программа, выполняемая над произвольными наборами данных.

Время, запрашиваемое в задании для выполнения программы $prog_k$, будем считать временем выполнения этой программы в ВУ$_e$ и обозначать t_e^k. Каждая программа обуславливает различную сложность вычислений, в том числе степень использования компонентов ВУ. Чтобы достаточно точно определить и запросить время, необходимое для выполнения программы, пользователь системы может воспользоваться средствами проведения в ВУ$_e$ предварительного динамического анализа вычислительной специфики своей программы. Такое ис-

следование позволяет выявить влияние характеристик компонентов ВУ на время выполнения программ и определить в нем долю использования каждого компонента. Динамический анализ программ осуществляется с помощью системы Intel VTune Performance Analyzer [14]. При отсутствии возможности проведения динамического анализа такая информация в значительной степени огрубленная, может быть извлечена эвристическими методами из описания задания или получена на основе экспертных оценок.

Вычислительную специфику программы $prog_k$ отразим через весовые коэффициенты $\alpha_1^k, \alpha_2^k, ..., \alpha_m^k$, представляющие долю времени использования *j*-го компонента ВУ$_e$ в общем времени выполнения программы $prog_k$ в ВУ$_e$ и удовлетворяющие условиям

$$0 \le \alpha_j^k \le 1, \ \sum_{j=1}^{m} \alpha_j^k = 1.$$

Определим t_i^k – прогнозное время выполнения программы $prog_k$ в ВУ$_i$ – соотношением

$$t_i^k = (\sum_{j=1}^{m} \alpha_j^k \sum_{\forall z: c_{iz} \in C_{ij}} \frac{\beta_{ez} c_{ez}}{c_{iz}}) t_e^k + \varepsilon, 1 \le n_j < n, ,$$

где β_{ez} – булев индикатор, показывающий, учитывать ($\beta_{ez} = 1$) или не учитывать ($\beta_{ez} = 0$) при прогнозировании характеристику c_{ez}, ε – погрешность прогнозирования, имеющая вид

$$\varepsilon = \frac{1}{v} \sum_{l \in I} ((t_l^k)^* - t_l^k),$$

где v – число выполнений программы $prog_k$, t_l^k и $(t_l^k)^*$ – соответственно прогнозное и реальное время выполнения программы в ВУ$_l$, $I = \{i_1, i_2, ..., i_v\}$ – множество индексов ВУ, в которых выполнялась программа $prog_k$.

Общее прогнозное время выполнения задания T^{job} складывается из следующих показателей:

$T^{job} = t^{sched} + t^{in\_send} + t^{prog\_send} + t_i^{ini\_wait} + t_i^k + t_i^{rest} + t_i^{exe\_wait} + t^{out\_send}$, где t^{sched}, t^{in_send}, t^{prog_send}, $t_i^{ini\_wait}$, t_i^{rest}, $t_i^{exe\_wait}$, t^{out_send} – это соответственно прогнозное время планирования и назначения задания, передачи входных данных, передачи программы, ожидания запуска, время на перезапуск, в случае сбоя, ожидании после завершения выполнения и передачи результатов счета. Показатели t^{sched}, $t_i^{ini\_wait}$, t_i^{rest} и $t_i^{exe\_wait}$ определяются на основе значений конфигурационных параметров системных планировщиков (значение t_i^{rest} находится с учетом показателей надежности ВУ). Показатели t^{in_send}, t^{prog_send} и t^{out_send} рассчитываются на основе знаний о сети передачи данных, заложенных в программно-аппаратном слое агрегированной концептуальной модели системы *M*.

Предложенный метод оценки времени выполнения заданий позволяет в динамике учитывать характеристики ВУ, вычислительную специфику исполняемых программ, а также конфигурации системных планировщиков и сети передачи данных. Построение соответствующих прогнозов осуществляется с помощью относительно простых математических процедур, не требующих больших накладных расходов по времени.

Уровни обслуживания заданий

Задание пользователя характеризуется множеством требований к ВУ и критериев их отбора (например, минимизировать время выполнения задания или стоимость выполнения задания) $S_i^u = \{q_{i1}^u, q_{i2}^u, ..., q_{il}^u\}$. Уровень обслуживания задания должен удовлетворять заданной совокупности качественных и количественных критериев к процессу его выполнения. Все множество характеристик уровня обслуживания задания $S_j^a = \{q_{j1}^a, q_{j2}^a, ..., q_{jk}^a\}$ разделяется на локальные и глобальные характеристики.

К локальным характеристикам относятся характеристики ВУ: оценка веса задания для ВУ; границы интервала изменения стоимости услуг ВУ; множество показателей эффективности функционирования ВУ, необходимых для определения административных утилитарных целей и задач использования ВУ. К числу таких показателей относятся коэффициент полезного использования ВУ, среднее время ожидания заданий в очереди к ВУ, среднее число заданий в очереди к ВУ и ряд других характеристик очередей заданий. Локальные характеристики уровня обслуживания задания задаются администраторами узлов системы.

Глобальными характеристиками являются приоритетность задания, правила постановки нового задания в очередь, распределения ВУ, использования при отборе ВУ показателей их надежности, обработки прерванного задания и др. В число глобальных характеристик входит также булева маска ВУ. Каждый элемент маски взаимно-однозначно соответствует одному из ВУ и определяет возможность использования данного ВУ для выполнения задания. Значения глобальных характеристик задаются администратором системы.

Уровень обслуживания задания назначается системой в процессе планирования и распределения вычислительных ресурсов. Назначенный уровень обслуживания задания определяет два важнейших с точки зрения пользователя показателя – время и стоимость вычислительных услуг.

Схема управления потоками заданий

Децентрализованное управление потоками заданий в системе реализуется двухуровневой системой интеллектуальных агентов (СИА/2), определяющих уровень обслуживания поступающих в систему заданий (рис. 2).

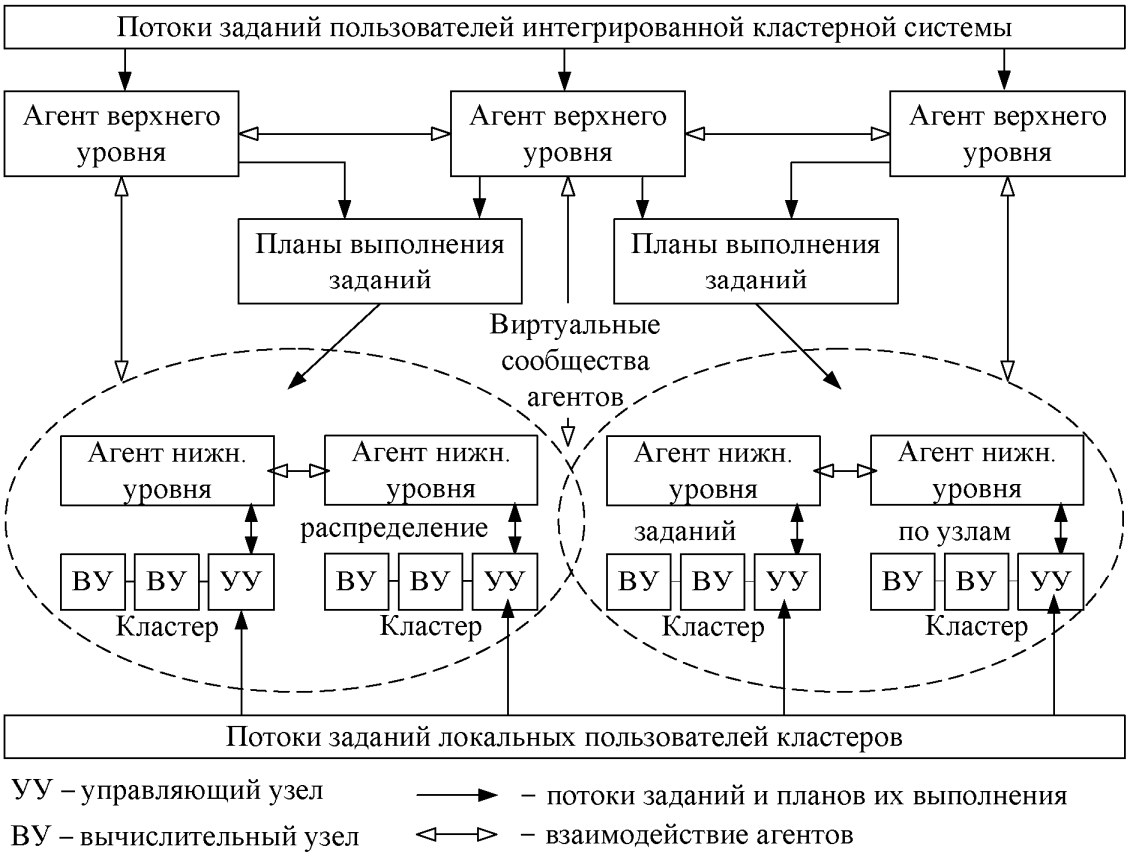


Рис. 2. Схема управления потоками заданий

В соответствии с полученным заданием агент верхнего уровня, используя знания о модели интегрированной кластерной системы, строит множество планов решения задачи [1] и формирует виртуальное сообщество из агентов нижнего уровня, обладающих потенциальными ресурсами для выполнения этих планов. Далее функции выбора конкретного плана решения задачи и его выполнение передаются сформированному виртуальному сообществу агентов нижнего уровня. В данном контексте план решения задачи представляет собой модель крупноблочной программы и отражает информационно-логическую структуру вычислений – связи между блоками (программными модулями) в терминах понятий предметной области решаемой задачи. Для заданий разных классов реализованы специализированные агенты верхнего уровня.

Агенты нижнего уровня, исходя из целей и задач, определенных политикой администрирования управляемых ими узлов, осуществляют выбор конкретного плана решения задачи и его выполнение на основе координированного сотрудничества [15].

Архитектура и программная реализация агентов

Агенты верхнего и нижнего уровня имеют унифицированную архитектуру, включающую системное ядро, подсистему логического вывода и коммуникационный интерфейс. Системное ядро обеспечивает управление вычислительными процессами и ресурсами, а также поддерживает взаимодействие между всеми компонентами агента. Подсистема логического вывода предназначена для реализации прав и полномочий, делегированных агентам верхнего и нижнего уровня соответственно пользователями и администраторами узлов системы с целью максимально возможного достижения поставленных им целей и задач по управлению распределенными вычислениями. Основой базы знаний для агентов верхнего уровня является модель *M*. База знаний агента нижнего уровня содержит информацию о программно-аппаратных ресурсах подчиненного ему узла, параметры политики администрирования узла, журнал статистики работы узла, сведения о других агентах системы, а также набор продукционных правил, необходимых для принятия решения по участию в процессе решения той или иной задачи. Знания агентов верхнего и нижнего уровня частично пересекаются. Коммуникационный интерфейс используется для общения агентов между собой и связи с внешней средой.

Алгоритмы функционирования агентов верхнего и нижнего уровней разработаны на основе конечно-автоматной модели в соответствии со спецификой действий, выполняемых этими агентами в системе. Данная модель имеет вид $M_a =< S, S_0, A, Mes, Log >$, где S – множество состояний агента; $S_0 \in S$ – начальное состояние агента; A – множество действий, совершаемых агентом, которое включает подмножество локальных действий $A_{locl} \subset A$, а также подмножества коммуникационных взаимодействий с другими агентами: отправок сообщений $A_{send} \subset A$ и приемов сообщений $A_{recv} \subset A$; Mes – множество входных и выходных сообщений агента; $Log : A \times S \to S$ – логическая функция переходов, которая ставит в соответствие текущему состоянию $s_i \in S$ и действиям из A, выполнимых в данном текущем состоянии, новое состояние $s_{i+1} \in S$.

Программная реализация архитектурных компонентов каждого агента выполнена с учетом специфики его функционирования на базе инструментального комплекса Java Agent Development Framework (JADE) [16]. Данный инструментарий реализован на языке Java и функционирует в любой среде, имеющей в своем программном окружении виртуальную машину Java Runtime Environment. Обмен сообщений строится на основе протокола Java RMI и языка сообщений Agent Communication Language [17], интегрированных с протоколами SMTP и HTTP. С помощью этих средств реализован ряд алгоритмов обмена сообщениями, ориентированных на различные конфигурации сетевой топологии системы и обеспечиваю-

[1] В общем случае, множество планов решения одной задачи возникает вследствие вычислительной избыточности, заложенной в модели интегрированной кластерной системы.

щих быстроту, синхронизацию и достаточную надежность процессов передачи и получения сообщений [18].

Использование инструментального комплекса JADE обеспечивает переносимость, открытость и эффективность разработанной мультиагентной системы, а также возможность ее дальнейшего, относительно «безболезненного» структурного и функционального развития.

Вычислительный эксперимент

Рассмотрим вычислительный эксперимент по моделированию работы СИА/2 и оценке ее эффективности по сравнению с другими аналогичными системами с точки зрения учета показателей надежности ВУ при распределении ресурсов. В качестве среды моделирования использовалась имитационная модель распределенной вычислительной среды, реализованная на языке GPSS. Для сравнения эффективности работы СИА/2 были выбраны: планировщик GridWay [19], являющийся де-факто стандартизированным средством распределения ресурсов и применяемый во многих практических Grid, а также планировщик, базирующийся на применении мобильных агентов [20].

Вычислительный эксперимент проводился в соответствии с методикой, предложенной в работе [21]. В качестве наблюдаемых переменных были выбраны число рестартов программ и число сбойных заданий (заданий, снятых с решения вследствие неоднократных рестартов или продолжительного времени неактивности программ, включенных в эти задания), так как именно эти переменные зависят от показателей надежности ВУ. Характеристики входных переменных – ненадежных ВУ (компьютеров, входящих в состав кластеров невыделенных ресурсов), видов неисправностей, а также показателей их возникновения и устранения (табл. 1), определялись на основе усредненной статистической информации о корпоративных вычислительных сетях, собранной в ряде учебных и научных организаций.

Имитационное моделирование представляет собой статистический эксперимент, поэтому при его проведении необходимо получить достоверный результат с заданной точностью. В общем случае количество прогонов модели, необходимое для получения оценок наблюдаемой переменной с заданной точностью, зависит от ряда факторов: коррелированности элементов выборки между собой, вида распределения наблюдаемой переменной и длительности переходного периода.

В нашем случае значения наблюдаемой переменной не коррелированы между собой и их распределение не меняется от прогона к прогону. Следовательно, выборочное среднее можно считать нормально распределенным [21], а количество необходимых прогонов модели N определять по формуле

$$N = \frac{\sigma_\alpha^2}{\varepsilon^2} t_\alpha^2,$$

где σ_α – среднеквадратическое отклонение наблюдаемой переменной, полученное на основе N_0 пробных прогонов модели, α – коэффициент достоверности выборки отклонений наблюдаемой переменной, ε – точность оценки наблюдаемой переменной, t_α – аргумент функции Лапласа.

Длительность переходного периода соотнесем с максимальной интенсивностью возникновения неисправностей ненадежных ВУ (табл. 2). Таким образом, длительность переходного периода будет равна 10 суткам модельного времени.

При $N_0 = 20$, $\alpha = 0{,}1$ и $\varepsilon = 0{,}1$ количество прогонов модели для первой наблюдаемой переменной (числа рестартов программ) равно 55 (из них 1 прогон для перехода в стационарный режим), а для второй переменной (числа сбойных заданий) количество прогонов модели равно 35 (из них 1 прогон для перехода в стационарный режим). Пробные прогоны модели вошли в общее число прогонов. Всего было проведено два вычислительных эксперимента с разной интенсивностью потоков заданий.

Моделируемая система включала 50 кластеров с числом ВУ от 8 до 1 280 единиц. Общее число ВУ – 10 000 единиц, из них 312 «ненадежных» (3,12 % от общего числа ВУ). В качестве СУПЗ кластеров имитировались системы SGE, PBS и Condor, конфигурационные пара-

метры которых были оптимизированы под инфраструктуру и нагрузку моделируемой интегрированной кластерной системы. Для разных планировщиков использовались одинаковые значения конфигурационных параметров СУПЗ.

Таблица 1

Список неисправностей

№	Вид неисправности	Интенсивность возникновения неисправности в одном из ненадежных ВУ (в секундах)	Время устранения неисправности (в секундах)
1.	Сбой процессора	863 200	16 950
2.	Сбой оперативной памяти	787 881	44 300
3.	Сбой дисковой памяти	469 800	6 300
4.	Сбой коммуникационной сети	158 630	1 122
5.	Отключение электропитания	570 240	3 750

Таблица 2

Состав потоков заданий

Системы управления потоками заданий	Потоки заданий					Эксперименты
	Стандартные задания	Задания по выполнению программ, размещенных в узлах системы	Многовариантные задания	Взаимосвязанные задания	Задания локальных пользователей	
GridWay	255	252	254	255	499	Эксперимент 1
Планировщик на основе мобильных агентов	254	253	255	254	500	
СИА/2	253	253	254	253	501	
GridWay	2 543	2 542	2 541	2 540	500	Эксперимент 2
Планировщик на основе мобильных агентов	2 540	2 539	2 541	2 541	499	
СИА/2	2 541	2 540	2 543	2 542	500	

В процессе моделирования число доступных ВУ изменялось от 9 844 до 9 996 единиц. Состав суточного потока заданий, управление которыми моделировалось в разных режимах, приведен в табл. 2. Количество подзаданий для взаимосвязанного задания составляло от 10 до 15 заданий. Многовариантные задания включали от 100 до 1 000 вариантов. Количество процессоров, требуемых для выполнения параллельных программ, изменялось от 200 до 1 000 единиц. Требуемые объемы оперативной и дисковой памяти составляли соответственно от 500 до 2 000 Мбайт и от 1 до 1 000 Мбайт. Моделируемый период времени работы системы – 30 суток.

Таблица 3

Результаты моделирования

Системы управления потоками заданий	Среднее число рестартов программ	Среднее число сбойных заданий	Эксперименты
GridWay	89	23	Эксперимент 1
Планировщик на основе мобильных агентов	86	21	
СИА/2	37	0	
GridWay	123	31	Эксперимент 2
Планировщик на основе мобильных агентов	125	28	
СИА/2	57	0	

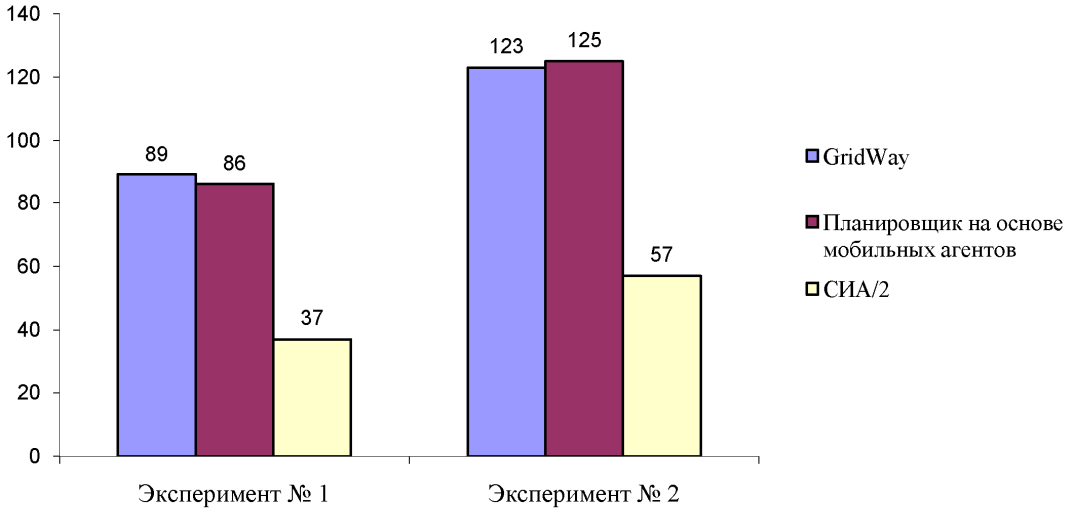
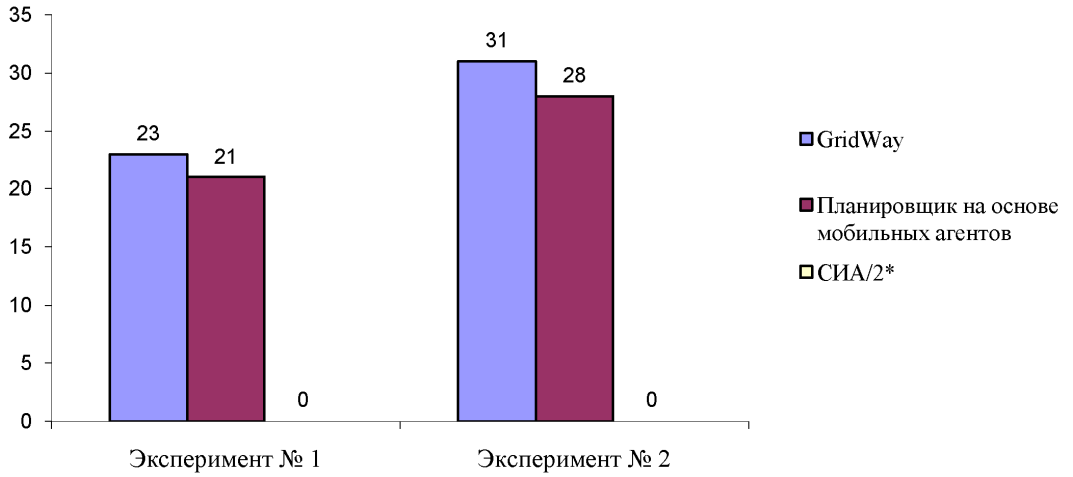


Рис. 3. Среднее число рестартов программ в сутки



*СИА/2: среднее число сбойных заданий равно 0.

Рис. 4. Среднее число сбойных заданий в сутки

Результаты моделирования (табл. 3) показывают, что применение СИА/2 может существенно улучшить показатели надежности процесса решения задач (среднее число рестартов программ и среднее число сбойных задач) по сравнению с другими известными системами (рис. 3 и 4). Снижение числа рестартов обусловлено тем, что СИА/2 учитывают при выборе ВУ вероятность его безотказной работы на время выполнения программы. Если же рестарт все-таки происходит, то программа успевает выполниться в течение следующего периода безотказной работы узла. Поэтому в экспериментах 1 и 2 (см. табл. 3) при использовании СИА/2 среднее число сбойных заданий равно 0.

Результаты имитационного моделирования были подтверждены в процессе полунатурных испытаний экспериментальной кластерной Grid Института динамики систем и теории управления СО РАН [22], а также в процессе решения ряда практических задач имитационного моделирования складской логистики [23].

Заключение

Представленный в данной работе децентрализованный подход к управлению распределенными вычислениями позволяет учитывать различные особенности интегрированных кластерных систем и тем самым повышать отдельные показатели эффективности функционирования системы. Дальнейшие исследования в этом направлении связаны с развитием алгоритмов взаимодействия агентов между собой на основе экономических методов оценки и учета характеристик заданий и ресурсов.

Список литературы

1. *Baker M., Buyya R., Laforenza D.* Grids and Grid Technologies for Wide-Area Distributed Computing // Software: Practice and Experience. 2002. Vol. 32, № 15. P. 1437–1466.

2. *Casavant T. L., Kuhl J. G.* A Taxonomy of Scheduling in General-Purpose Distributed Computing Systems // IEEE Trans. On Software Engineering. 1988. Vol. 14, № 2. P. 141–154.

3. *Vidyarthi D. P., Sarker B. K., Tripathi A. K., Yang L. T.* Scheduling in Distributed Computing Systems: Analysis, Design and Models. Springer Science+Business Media, 2009.

4. *Durfee E. H.* Distributed Problem Solving and Planning // Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence / Ed. by G. Weiss. MIT Press, 1999. P. 121–164.

5. Market-Oriented Grid and Utility Computing / Eds. R. Buyya, K. Bubendorfer. Wiley & Sons, 2010.

6. *Таха Х. А.* Введение в исследование операций. М.: Вильмс, 2005.

7. *Foster I., Jennings N. R., Kesselman C.* Brain Meets Brawn: Why Grid and Agents Need Each Other // Proc. of 3rd International Conference on Autonomous Agents and Multi-Agent Systems. N. Y., 2004. P. 8–15.

8. Человек и вычислительная техника / Под ред. В. М. Глушкова. Киев: Наук. дум., 1971.

9. *Тыугу Э. Х.* Концептуальное программирование. М.: Наука, 1984.

10. *Воеводин В. В., Воеводин Вл. В.* Параллельные вычисления. СПб.: БХВ-Петербург, 2002.

11. *Опарин Г. А., Феоктистов А. Г., Александров А. А.* Графическая инструментальная среда для описания модели распределенной вычислительной системы // Вестн. Иркут. гос. техн. ун-та. 2006. № 2 (26). Т. 3. С. 35–40.

12. *Zanikolas S., Sakellariou R.* A Taxonomy of Grid Monitoring Services // Future Generation Computer Systems. 2005. Vol. 21, № 1. P. 163–188.

13. *Ушаков И. А.* Вероятностные модели надежности информационно-вычислительных систем. М.: Радио и связь, 1991.

14. *Reinders J.* VTune Performance Analyzer Essentials: Measurement and Tuning Techniques for Software Developers. Intel Press, 2003.

15. *Wooldridge M. J., Jennings N. R.* Towards a Theory of Cooperative Problem Solving // Proc. of 6th European Workshop on Modelling Autonomous Agens in a Multi-Agent World. Odense, Denmark, 1994. P. 15–26.

16. *Bellifemine F., Bergenti F., Caire G., Poggi A.* Jade – A Java Agent Development Framework // Multiagent Systems, Artificial Societies, And Simulated Organizations: Multi-Agent Programming / Eds. R. Bordini, M. Dastani, J. Dix, A. El Fallax Seghrouchni. Springer, 2006. Vol. 15. P. 125–147.

17. *Labrou Y., Finin T.* Semantics and Conversations for an Agent Communication Language // Readings in Agents / Eds. M. Huhns, M. Singh. San Mateo, 1998. P. 235–242.

18. *Тель Ж.* Введение в распределенные алгоритмы. М.: МЦНМО, 2009.

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

20. *Altameem T., Amoon M.* An Agent-based Approach for Dynamic Adjustment of Scheduled Jobs in Computational Grids // Изв. РАН. Теория и системы управления. 2010. № 5. С. 87–94.

21. *Боев В. Д.* Моделирование систем. Инструментальные средства GPSS World. СПб.: БХВ-Петербург, 2004.

22. *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.

23. *Дмитриев В. И., Башарина О. Ю., Феоктистов А. Г., Ларина А. В.* Моделирование современных логистических складских комплексов с использованием вычислительной техники // Экономика и управление. 2010. № 6. С. 88–91.

Материал поступил в редколлегию 29.04.2011

I. V. Bychkov, G. A. Oparin, A. G. Feoktistov, A. S. Korsukov

DECENTRALIZED JOB FLOW CONTROL IN THE INTEGRATED CLUSTER SYSTEM

In this paper the approach to job flow control in integrated cluster system is reviewed. This approach is based on applying the methodology of conceptual programming, technology of multi-agent systems, and queuing theory.

Keywords: distributed computing, integrated cluster systems, job flow control, multi-agent systems.