Математическое обеспечение эффективного разделения ресурсов для решения задач на распределенной вычислительной среде

В. А. Васенин, Д. А. Сериков

Московский государственный университет им. М. В. Ломоносова, 119992, Москва, Россия

Рассматриваются подходы к математическому моделированию процесса разделения ресурсов между задачами различных пользователей, которые поступают на обслуживание в распределенную вычислительную среду, построенную на основе методологии Grid. Анализируются модели системы диспетчеризации, которые используются в инструментальном комплексе GridWay.

Ключевые слова: математическое моделировние, распределенные вычислительные системы.

The paper discusses approaches to mathematical modeling of the division of resources between jobs of the various users which arrive to services in a distributed computing environment, based on the methodology GRID. Analyzed a model of the system of dispatching, which are used in the instrumental complex GridWay.

Keywords: mathematical modeling, distributed computing, GRID.

Введение. Несмотря на быстрый рост производительности отдельных вычислительных установок [1], решить с их помощью целый ряд сложных научно-технических и практически значимых задач в настоящее время не представляется возможным. Одной из главных причин такого положения дел является недостаток необходимых для этого объемов вычислительных ресурсов (ресурс − средство вычислительной машины (или нескольких), которым она может пользоваться в процессе работы (процессорное время, оперативная и дисковая память и т. п.)), которыми даже такие сверхвысокопроизводительные установки не располагают. Вместе с тем высокие темпы развития информационно-вычислительных и коммуникационных технологий, сетевой инфраструктуры на основе пакетных коммуникаций создают технические предпосылки для консолидации таких ресурсов. Методология построения подобных распределенных (в том числе − географически) информационно-вычислительных комплексов, консолидирующих ресурсы различных организаций для решения сложных задач, начала активно развиваться приблизительно 10-15 лет назад, когда методология Grid применялась для решения многих научных задач, связанных, например, с расшифровкой генома человека [2] или поиска внеземных цивилизаций [3]. В 1997 г. появился первый крупный проект distributed.net [4], посвященный использованию ресурсов компьютеров обычных пользователей с помощью сети Интернет для решения исследовательских задач, требующих больших вычислительных ресурсов.

Согласно трем критериям классического определения Grid-системы, предложенным одним из идеологов данной методологии Я. Фостером [5], Grid − открытая стандартизованная компьютерная среда, которая обеспечивает децентрализованное разделение ресурсов и высококачественное обслуживание в рамках виртуальной организации. При этом под виртуальной организацией понимается группа пользователей, как отдельных лиц, так и структурных подразделений различных форм собственности, совместно использующих общие ресурсы. Открытость и стандартизация означают, что система должна строиться на основе стандартных, открытых протоколов и интерфейсов, позволяющих решать такие традиционные задачи, как аутентификация, авторизация, обнаружение ресурсов и управление доступом к ним. Кроме того, система должна координировать использование ресурсов при отсутствии централизованного управления ими. Следует отметить некоторое отличие от Grid-системы слабосвязанного кластера − группы компьютеров, объединенных высокоскоростными каналами связи, с точки зрения пользователя представляющей собой единый аппаратный ресурс. В случае такого кластера речь идет о компьютерах, изначально управляемых из единого центра, в случае Grid − об одноранговой сети с независимыми узлами, которые объединяются в виртуальные организации для решения общих задач. В этом смысле слабосвязанные кластеры не являются Grid-системами. Использование ресурсов в Grid должно осуществляться таким образом, чтобы обеспечивались должная функциональность, высокое качество обслуживания потенциальных клиентов и защищенность. Обслуживание характеризуется доступностью ресурсов, надежностью работы системы в целом, временем отклика на запрос пользователя, пропускной способностью сетевых каналов и другими подобными атрибутами.

В силу многопользовательского характера Grid-инфраструктур необходимым механизмом, обеспечивающим высококачественное обслуживание, является планирование ресурсов (в рамках процесса диспетчеризации − автоматического распределения ресурсов при обслуживании запросов пользователей). Процесс планирования координирует разделения ресурсов между задачами пользователей. При правильной организации процесс диспетчеризации (и планирования) должен требовать от пользователя минимального участия, а именно пользователь должен лишь запускать задачу и получать результат. Задачу, где и как будет исполняться приложение, диспетчер должен решить самостоятельно, без дополнительного вмешательства со стороны пользователя. Диспетчер является одним из основных компонентов любого Grid-комплекса, и его функциональность является фактором, определяющим производительность распределенных вычислений. Поэтому разработка математической модели эффективного диспетчера для Grid-комплексов является актуальной задачей.

В настоящей работе рассмотрены возможные подходы к моделированию процесса диспетчеризации задач, к которым относятся аналитический и дискретно-событийный.

1. Ключевые понятия. В целях унификации используемых далее терминов и определений введем ключевые понятия.

Grid-среда − открытая, стандартизованная (т. е. строящаяся на основе стандартных протоколов и интерфейсов), распределенная вычислительная среда, объединяющая совокупность распределенных ресурсов, которая поддерживает их децентрализованное разделение для решения широкого класса задач с высоким качеством обслуживания в рамках виртуальной организации.

Ресурс − логический или физический элемент распределенной вычислительной среды Grid, который может быть выделен пользователю. Рассматриваются только вычислительные ресурсы, которые используются для исполнения задач. Примерами вычислительных ресурсов являются процессорное время, оперативная и дисковая память.

Задача − программная единица для обработки в распределенной вычислительной среде Grid. Задача включает исполняемый файл, паспорт (файл, содержащий описание задачи), а также файлы входных и выходных данных. Рассматриваются не все возможные задачи, а только некоторый подкласс.

Виртуальная организация − владельцы (поставщики) и пользователи (потребители) ресурсов, действующие на основе некоторых правил предоставления (потребления) ресурсов.

Диспетчеризация задач − автоматическая обработка набора задач, включающая механизмы планирования ресурсов, доставку необходимых входных файлов на ресурс, запуск, управление и мониторинг выполнения задач, доставку выходных файлов.

Планирование ресурсов − распределение задач по доступным ресурсам.

В инфраструктурах Grid для диспетчеризации задач могут быть использованы различные подходы, наиболее распространенным из них является централизованная диспетчеризация задач. При использовании этого подхода все задачи распределяются на ресурсы одной специальной службой, именуемой диспетчером. Однако такая централизация в определенной степени противоречит методологии Grid, которая основывается на децентрализованном разделении ресурсов. Поэтому диспетчеризация задач в Grid-среде должна осуществляться с помощью нескольких независимых диспетчеров. Как следствие децентрализованный подход к диспетчеризации задач может строиться на базе механизмов централизованного подхода. Централизованная диспетчеризация применяется в системах [6-8]. В данной работе в качестве объекта для исследования рассматривается программный комплекс GridWay [7] как наиболее распространенный в настоящее время диспетчер для Grid-систем. В качестве инфраструктурной основы для исследований функциональных возможностей и эффективности механизмов планирования ресурсов для решения вычислительных задач на Grid-среде использовался экспериментальный Grid-полигон [6]. Данный полигон развернут на базе высокопроизводительных вычислительных систем Научно-исследовательского Института механики МГУ (Москва), Института проблем информационной безопасности МГУ (Москва), Научно-образовательного Центра компьютерного моделирования и безопасных технологий МГУ (Москва) и Института вычислительной математики и математической геофизики СО РАН (Новосибирск), объединенных с помощью сетей передачи данных МГУ − РАН.

2. Аналитическая модель системы диспетчеризации задач, используемой в GridWay. Исследование аналитической модели системы диспетчеризации задач, используемой в программном комплексе GridWay, представляет большой интерес, так как она позволяет описывать потенциально возможные механизмы, которые могут применяться для моделирования подобных систем. Содержательно данная модель отражает количественную зависимость эффективности процесса диспетчеризации задач от параметров Grid-полигона. К их числу относятся интервалы времени, необходимые для поиска новых узлов в Grid-полигоне, для обновления сведений по ресурсам, для опроса состояния задач и т. п. Аналитический характер модели означает, что она представляет собой функциональную зависимость выходных параметров (эффективность процесса диспетчеризации задач) от входных (параметры Grid-полигона). Для формального определения процесса диспетчеризации задач вводятся понятия вычислительного ресурса и функции утилизации вычислительного ресурса.

Определение 2.1. Вычислительный ресурс − логический или физический элемент распределенной вычислительной среды Grid, который соответствует одной вычислительной машине или кластеру. Вычислительный ресурс может включать один или несколько процессоров (или слотов).

Определение 2.2. Функция утилизации процессора p_j на временном интервале (t_1, t_2) определяется как отношение количества тактов процессора, потраченных на пользовательский режим на временном интервале (t_1, t_2), к общему числу тактов процессора на этом интервале:

$$f_j(t_1,t_2) = \frac{U_j(t_2) - U_j(t_1)}{T_j(t_2) - T_j(t_1)}.$$

(Режим user mode - непривилегированный режим работы процессора, предназначенный для исполнения прикладных программ. В отличие от привилегированного режима (system mode), доступ к системным ресурсам программам предоставляется только посредством системных вызовов.) Содержательно данное понятие означает, что, чем меньше разность $1 - f_j(t_1,t_2)$, тем эффективнее используется процессор p_j. Без ограничения общности можно считать, что моделирование проводится на процессорах с постоянной тактовой частотой $T(t) = kt$ (k − тактовая частота процессора). Для таких процессоров функция утилизации принимает вид

$$f_j(t_1,t_2) = \frac{U_j(t_2) - U_j(t_1)}{k_j(t_2 - t_1)}.$$

Определение 2.3. Функция утилизации вычислительного ресурса h_i на временном интервале (t_1, t_2) представляет собой среднее арифметическое функций утилизации процессоров p_j на соответствующем временном интервале для всех процессоров p_j, входящих в ресурс h_i:

$$F_i(t_1,t_2) = \frac{1}{n_i}\sum_{p_j \in h_i} f_j(t_1,t_2) = \frac{1}{n(t_2 - t_1)}\sum_{p_j \in h_i} \frac{U_j(t_2) - U_j(t_1)}{k_j},$$

где n_i − число процессоров узла h_i.

В операционной системе GNU/Linux величину $U(t)$ в любой момент времени можно вычислить с помощью файла */proc/stat*. Однако массовое вычисление этой функции (например, на многопроцессорных кластерах) потребовало бы разработки собственной системы мониторинга. Существующие системы мониторинга могут вычислять несколько иную величину.

Определение 2.4. Загрузка процессора p_j в момент времени t определяется значением функции утилизации процессора p_j на временном интервале $(t - \Delta t, t)$:

$$c_j(t) = c_j(t, \Delta t) = \frac{U_j(t) - U_j(t - \Delta t)}{k_j \Delta t},$$

где Δt − фиксированный отрезок времени.

Предположим, что $t_2 - t_1 = n\Delta t$ (n − целое). Тогда $U_j(t_2) - U_j(t_1) = k_j \Delta t(c_j(t_2) + c_j(t_2 - \Delta t) + \ldots + c_j(t_2 - (n-1)\Delta t))$.

Таким образом,

$$f_j(t_1, t_2) = \frac{\Delta t}{t_2 - t_1} \sum_{l=0}^{n-1} c_j(t_2 - l\Delta t).$$

Функция утилизации для вычислительного ресурса h_i на временном интервале (t_1, t_2) рассчитывается по формуле

$$F_i(t_1, t_2) = \frac{\Delta t}{n_i(t_2 - t_1)} \sum_{p_j \in h_i} \sum_{l=0}^{n-1} c_j(t_2 - l\Delta t).$$

С целью получения количественной характеристики эффективности системы диспетчеризации задач, используемой в GridWay, проведено несколько последовательных запусков наборов задач на Grid-полигоне [9].

Для измерения загрузки процессора в зависимости от компьютерной системы можно использовать пакеты collectd [10] и Ganglia [11]. Пакет collectd − небольшой демон, который периодически собирает информацию о системной активности (нагрузка процессора и памяти, сетевой трафик, системная температура и подобная им) через lm-сенсоры. Пакет Ganglia − масштабируемая распределенная система мониторинга для высокопроизводительных компьютерных систем, таких как вычислительные кластеры и комплексы Grid.

Отличительной особенностью указанных пакетов является тот факт, что собранная ими информация помещается в RRD-файлы (Round Robin Database [12]). С помощью набора утилит для работы с RRD RRDtool информация о системной активности может быть извлечена и проанализирована. Сбор информации с удаленных хостов можно осуществить с помощью SSH.

В основу выбора методики тестирования положены следующие соображения. Можно послать на выполнение массив из нескольких задач (100), а после их выполнения вычислить функцию утилизации для каждого ресурса. Следует отметить, что при таком подходе не совсем ясно, как выбирать параметр t_2. С одной стороны, он может быть для всех ресурсов одинаковым (время завершения выполнения последней задачи). Однако тогда факт "простоя" ресурсов, на которых выполнение задач завершилось ранее момента времени t_2, изменит значение функции утилизации на них. С другой стороны, время t_2 может быть равно времени завершения выполнения последней задачи на данном ресурсе. В этом случае не ясно, как приводить результаты для каждого ресурса "к общему знаменателю", потому что они получены при разных параметрах измерений. Кроме того, если ресурс был занят (например, локальной очередью задач), то при таком выборе параметра t_2 функция утилизации на нем вырождается ($t_1 = t_2$). В силу сказанного выше применялась методика тестирования, согласно которой на выполнение посылался большой пул задач (1000), а функция утилизации для каждого ресурса вычислялась периодически через каждые 15 мин.

2.1. *Выявление основных параметров Grid-полигона, оказывающих наибольшее влияние на эффективность диспетчеризации задач.* Планировщик GridWay имеет несколько параметров (все они задаются в конфигурационных файлах *gwd.conf* и *sched.conf*), комбинация которых, принятая для исполнения, оказывает влияние на эффективность процесса диспетчеризации. Основными параметрами являются:

− *SCHEDULING_INTERVAL* (*SI*) − период между двумя итерациями планирования;

− *DISCOVERY_INTERVAL* (*DI*) − период поиска новых хостов в Grid-полигоне;

− *MONITORING_INTERVAL* (*MI*) − период обновления сведений по каждому ресурсу;

− *POLL_INTERVAL* (*PI*) − период опроса о состоянии задач;

− *DISPATCH_CHUNK* (*DC*) − максимальное число задач, которые будут обработаны планировщиком за один шаг планирования.

В таблице приведены результаты вычисления утилизации четырех ресурсов Grid-полигона, полученные при указанных значениях параметров.

Результаты вычисления функции утилизации

SI	*DI*	*MI*	*PI*	*DC*	tp(2)	gf(2)	imec(16)	sscc(81)
30	900	300	180	15	62,58 %	75,76 %	66,31 %	51,36 %
15	900	300	180	15	65,82 %	81,00 %	75,99 %	53,38 %
60	900	300	180	15	53,98 %	69,88 %	63,75 %	51,31 %
30	450	300	180	15	66,70 %	80,36 %	61,18 %	45,51 %
30	1800	300	180	15	62,50 %	70,80 %	74,22 %	53,46 %
30	900	150	180	15	65,66 %	72,93 %	62,87 %	53,17 %
30	900	600	180	15	53,75 %	81,90 %	68,07 %	49,98 %
30	900	300	90	15	63,61 %	82,75 %	67,00 %	61,34 %
30	900	300	360	15	56,91 %	75,42 %	60,10 %	42,15 %
30	900	300	180	5	41,53 %	48,39 %	45,23 %	35,49 %
30	900	300	180	30	69,33 %	79,34 %	68,36 %	53,24 %

В качестве задачи использовались задача доказательства ненадежности криптоалгоритма RC5 путем взлома зашифрованного им сообщения; задача поиска новых, более оптимальных "линеек Голомба" [12]. Обе задачи представлены в материалах проекта распределенных вычислений distributed net [4].

Результаты тестирования показали, что из пяти перечисленных параметров наибольшее влияние на эффективность процесса диспетчеризации задач оказывают параметры *SI*, *PI* и *DC*. При этом с уменьшением периода между двумя итерациями планирования (*SI*), а также с уменьшением периода опроса ресурсов о состоянии выполняемых на них задач (*PI*) функция утилизации увеличивается. Малые значения параметра *DC* существенно снижают эффективность процесса диспетчеризации, что закономерно: за одну итерацию планирования обрабатывается меньшее количество задач, чем доступно ресурсов, поэтому ресурсы начинают "простаивать". Тестирование не позволило определить зависимости функции утилизации от периода поиска новых хостов в составе Grid-полигона (*DI*) и периода обновления сведений по каждому ресурсу (*MI*). При последовательном уменьшении (увеличении) этих параметров значение функции утилизации могло как уменьшаться, так и увеличиваться.

3. Имитационное моделирование. Представленная ранее аналитическая модель основана на анализе эффективности процесса диспетчеризации задач в различных условиях, например при разных значениях входных параметров, в том числе характеризующих состояние полигона и системы диспетчеризации. Такая модель позволяет получить адекватный результат, поскольку в ней учитываются практические данные. Вместе с тем подобный подход требует очень больших вычислительных затрат на его реализацию. Поэтому целесообразно исследовать систему с помощью методов имитационного моделирования [13]. Такой способ моделирования позволяет проводить многократные испытания модели с необходимыми входными данными, для того чтобы определить их влияние на выходные критерии оценки работы системы.

Одним из видов имитационного моделирования является дискретно-событийное моделирование, которое используется для построения модели, отражающей изменение состояния системы во времени, когда оно меняется мгновенно в конкретные моменты времени. Смену состояния системы называют событием. Одна из

наиболее распространенных моделей дискретно-событийного моделирования − система массового обслуживания − объект, деятельность которого связана с многократной реализацией исполнения некоторых однотипных задач и операций. Система массового обслуживания состоит из обслуживаемой и обслуживающей подсистем. Обслуживаемая подсистема включает совокупность источников требований и входящего потока требований. (В теории массового обслуживания термин требование является устоявшимся аналогом английского термина customers.) Обслуживающая система состоит из накопителя и механизма обслуживания.

Учитывая специфику исследуемой системы диспетчеризации задач, которая, по сути, является системой, обслуживающей входящий поток задач, можно предположить, что эффективным для нее будет применение модификации стандартной дискретно-событийной модели системы массового обслуживания.

3.1. *Моделирование систем массового обслуживания.* Система массового обслуживания состоит из одного или нескольких устройств обслуживания, предоставляющих какие-либо услуги входящему потоку требований. Требования, поступающие, как правило, тогда, когда все устройства заняты, образуют одну или несколько очередей к устройствам обслуживания. Система массового обслуживания включает три компонента: процесс поступления, механизм обслуживания и дисциплину обслуживания.

Процесс поступления состоит из описания механизма появления требований в системе массового обслуживания. Допустим, что A_i − время между поступлениями требования $(i\text{-}1)$ и требования i. Если $A_1, A_2, \ldots$ являются независимыми одинаково распределенными величинами, то среднее время (или математическое ожидание) между поступлениями можно обозначить как $E(A)$, а $\lambda = 1/E(A)$ является интенсивностью поступления требований.

Механизм обслуживания в системе массового обслуживания определяется следующими факторами: числом устройств обслуживания *N*, наличием для каждого устройства своей очереди или существованием одной очереди для всех устройств, распределением вероятностей времени обслуживания требований. Пусть S_i − время обслуживания поступившего требования i. Если $S_1, S_2, \ldots$ − независимые и одинаково распределенные величины, то среднее время обслуживания требования выразится как $E(S)$, и скорость обслуживания требований составит $\omega = 1/E(S)$.

Дисциплина обслуживания определяется правилом, которое устройство обслуживания использует для выбора из очереди следующего требования (если таковые имеются) по завершении обслуживания требования текущего. Как правило, используются следующие дисциплины очереди:

− требования FIFO (First-In, First-Out) обслуживаются по принципу "первым поступило − первым обработано";

− требования LIFO (Last-In, First-Out) обслуживаются по принципу "последним поступило − первым обработано";

− требования с приоритетом обслуживаются в порядке их значимости или в соответствии с другими требованиями к обслуживанию.

Рассмотрим систему массового обслуживания, имеющую следующие характеристики:

1) *N* устройств массового обслуживания и одна очередь FIFO для всех этих устройств;

2) $A_1, A_2, \ldots$ − независимые одинаково распределенные величины;

3) $S_1, S_2, \ldots$ − независимые одинаково распределенные величины;

4) величины A_i и S_i не зависят друг от друга.

Такая система массового обслуживания обозначается как *GI/G/N*. При этом *GI* (General Independent − произвольное независимое) относится к распределению величин A_i, а *G* (General − произвольное) − к распределению величин S_i.

Для любой системы массового обслуживания *GI/G/N* величина $\rho = \lambda / N\omega$ называется коэффициентом использования системы массового обслуживания ($N\omega$ − скорость обслуживания в системе, когда заняты все устройства обслуживания). Коэффициент использования является показателем того, насколько эффективно задействованы ресурсы системы.

3.2. *Дискретно-событийная модель системы диспетчеризации задач в GridWay*. Рассмотрим систему массового обслуживания *GI/G/N* с *N* устройствами обслуживания. В этой терминологии система массового обслуживания соответствует диспетчеру GridWay, а каждое из устройств обслуживания − вычислительному ресурсу. В данной системе интервалы времени между поступлениями требований (задач) $A_1, A_2, \ldots$ являются независимыми одинаково распределенными случайными величинами. Когда задача поступает, и при этом хотя бы один из ресурсов свободен, обслуживание задачи начинается немедленно. Время обслуживания $S_1, S_2, \ldots$ следующих задач представлено также независимыми одинаково распределенными случайными величинами. Если при поступлении задачи все ресурсы заняты, то задача ставится в очередь. При освобождении какого-либо ресурса (т. е. по завершении обслуживания одной из задач) диспетчер выбирает задачу из очереди (если такая имеется) по принципу FIFO.

Моделирование начинается с состояния, когда в системе отсутствуют задачи и все ресурсы свободны. С момента времени, равного нулю (начального), начинаем ожидать поступление первой задачи. С большой вероятностью это произойдет по истечении некоторого интервала времени A_1, а не в начальный момент времени. Моделирование завершается по прошествии фиксированного периода времени *T*. Следовательно, время завершения моделирования является случайной величиной, которая зависит от наблюдаемых значений случайных переменных, обозначающих интервалы времени между поступлениями задач и интервалы времени, затраченные на их обслуживание.

Рассмотрим видоизмененную систему массового обслуживания *GI/G/N*, которая обладает следующими свойствами.

1. Если хотя бы один из ресурсов свободен, то выборка задач из очереди и их обслуживание начинаются не немедленно, а в моменты времени, кратные параметру *SI*, где *SI* > 0 − параметр системы. Применительно к диспетчеру GridWay это означает, что планирование ресурсов для задачи происходит не в момент ее поступления, а периодически, с интервалом в *SI* секунд, что соответствует параметру *SCHEDULING_INTERVAL*.

2. $N_0, N_1 N_2 \ldots$ − независимые одинаково распределенные величины, представляющие общее количество ресурсов системы в моменты времени 0, *DI*, 2*DI*, ..., где *DI* > 0 − параметр системы. Параметр *DI* соответствует периоду времени, с которым диспетчер GridWay проверяет Grid-полигон на наличие новых ресурсов (параметр *DISCOVERY_INTERVAL* диспетчера).

3. $M_0, M_1, M_2 \ldots$ − независимые одинаково распределенные величины, представляющие количество ресурсов системы, занятых локальными очередями соответственно в моменты времени 0, *MI*, 2*MI*, ..., где *MI* > 0 − параметр системы. Параметр *MI* соответствует периоду обновления сведений по каждому ресурсу (параметр *MONITORING_INTERVAL* диспетчера).

4. Если обслуживание задачи i началось в момент времени *kSI* (см. свойство 1), то соответствующий ресурс освободится в момент времени

$$PI\left\lceil \frac{kSI + S_i}{PI} \right\rceil,$$

где *PI* > 0 − параметр системы. Диспетчер GridWay опрашивает ресурсы о состоянии задач не постоянно, а периодически с интервалом *PI* (соответствующий параметр *POLL_INTERVAL*). Поэтому после выполнения задачи ресурс сможет освободиться только в момент времени, кратный *PI*.

5. За один шаг планирования, соответствующий выборке задач из очереди, из нее извлекается не более *DC* задач, где *DC* > 0 − параметр системы, соответствующий максимальному числу задач, которые будут обработаны модулем планирования GridWay за одну итерацию (параметр *DISPATCH_CHUNK* диспетчера).

Определение 3.1. Система, обладающая свойствами 1-5, называется системой массового обслуживания *GI/G/D/M*(*SI*,*DI*,*MI*,*PI*,*DC*). При этом *GI* относится к распределению величин A_i, G − к распределению величин S_i, D и M − к распределению величин N_i и M_i соответственно.

Заметим, что система массового обслуживания *GI/G/D/M*(*SI,DI,MI,PI,DC*) является дискретно-событийной моделью системы диспетчеризации задач, используемой в GridWay. Кроме того, при $N_0 = N_1 = N_2 = ... = N, M_0 = M_1 = M_2 = ... = 0$ система массового обслуживания *GI/G/N* является предельным случаем системы *GI/G/D/M*(*SI,DI,MI,PI,DC*) при $SI, PI \to 0, DC \to \infty$. Для данной системы массового обслуживания *GI/G/D/M*(*SI,DI,MI,PI,DC*) величину $\rho = \lambda / s\omega$ назовем коэффициентом использования системы массового обслуживания (s=*E*(*N*)-*E*(*M*) − среднее количество ресурсов в системе).

3.3. *Критерии оценки работы системы диспетчеризации задач в GridWay.* Существует достаточно большое количество критериев оценки работы систем массового обслуживания. Рассмотрим показатели эффективности их работы, которые обычно используются при математическом исследовании систем массового обслуживания. Примем следующие обозначения: D_i − время задержки в очереди задачи i; $W_i = D_i + S_i$ − время ожидания в системе задачи i; $Q(t)$ − число задач в очереди в момент времени t; $L(t)$ − число задач в системе в момент времени t ($Q(t)$ плюс число задач, которые находятся на обслуживании в момент времени t); $B(t)$ − число занятых ресурсов.

Тогда показатели

$$d = \lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} D_i$$

с вероятностью 1 (выражение "с вероятностью 1" дано для соблюдения математической корректности и не имеет практического значения, поэтому далее опускается) и

$$w = \lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} W_i$$

(если они существуют) называются соответственно установившейся средней задержкой и установившимся средним временем ожидания. Показатели

$$Q = \lim_{T\to\infty} \frac{1}{T}\int_0^T Q(t)dt$$

и

$$W = \lim_{T\to\infty} \frac{1}{T}\int_0^T L(t)dt$$

(если они существуют) называются соответственно установившимся средним по времени числом задач в очереди и установившимся средним по времени числом задач в системе.

Показатель

$$u = \lim_{T\to\infty} \frac{1}{T}\int_0^T B(t)dt$$

(если он существует) называется установившимся коэффициентом использования ресурсов (часть времени, когда ресурсы находятся в состоянии "занято").

3.4. *Компьютерное моделирование дискретно-событийной модели системы диспетчеризации задач, используемой в GridWay.* Определим компоненты, необходимые для компьютерного моделирования системы массового обслуживания *GI/G/D/M*(*SI,DI,MI,PI,DC*) на языке С.

Временные интервалы между поступлениями и время обслуживания моделируются по экспоненциальным распределениям как независимые случайные величины. Экспоненциальное распределение со средним значением $\lambda > 0$ является непрерывным, с плотностью распределения вероятностей [12]

$$f(x) = \frac{1}{\lambda}\mathrm{e}^{-x/\lambda}.$$

Экспоненциальное распределение выбрано потому, что на компьютере несложно сгенерировать экспоненциально распределенные случайные величины. В действительности допущение экспоненциально распределенного времени между поступлениями, как правило, хорошо соответствует реалиям. Принятие экспоненциально распределенного времени обслуживания при этом является менее правдоподобным [13].

Определение 3.2. Систему массового обслуживания *GI/G/D/M*(*SI,DI,MI,PI,DC*) с экспоненциально распределенными временными интервалами между поступлением задач, временем их обслуживания, общим количеством ресурсов и количеством ресурсов, занятых локальными очередями, будем называть системой *M/M/M/M*(*SI,DI,MI,PI,DC*) (*M* обозначает экспоненциальное распределение).

Для генерирования случайных чисел будем использовать метод Марсе − Робертса [14].

Программа состоит из нескольких подпрограмм. Кроме основной программы программа моделирования включает подпрограммы для инициализации, синхронизации, обработки событий, генерирования отчетов и экспоненциально распределенных случайных величин. Наиболее важные действия производятся в подпрограммах событий и имеют следующую нумерацию: 1) поступление задачи в систему; 2) уход задачи из системы по окончании обслуживания; 3) планирование задач; 4) поиск новых ресурсов; 5) обновление информации о ресурсах; 6) генерация отчета. Моделирование завершается тогда, когда число задач, задержанных в очереди, достигает 1000.

Рассмотрим логику работы подпрограмм обработки событий.

1. При поступлении задачи в систему в первую очередь генерируется время следующего поступления и помещается в список событий. Затем задача помещается в очередь и проверяется, заполнена ли память, выделенная для хранения очереди. Если очередь уже заполнена, генерируется сообщение об ошибке, и моделирование прекращается. Если в очереди еще имеется место, время поступления задачи помещается в конец очереди.

2. После извлечения задачи из очереди число занятых ресурсов уменьшается на единицу и вычисляется время следующего ухода задачи.

3. Согласно модели подпрограмма обработки события планирования задач вызывается каждые *SI* мин. Если очередь задач не пуста и имеются свободные ресурсы, то первая задача в очереди оставляет очередь и переходит на обслуживание. При этом продолжительность задержки этой задачи вычисляется и регистрируется соответствующим статистическим счетчиком. Число задержек в очереди увеличивается на единицу, и планируется время ухода задачи, перешедшей на обслуживание. Оставшиеся после этого в очереди задачи (если таковые имеются) передвигаются на одно место вперед. Цикл планирования продолжается до тех пор, пока либо в очереди не останется задач, либо не останется свободных ресурсов, либо не будет обслужено *DC* задач. Затем вычисляется время следующего ухода задачи, а также следующее время планирования задач.

4. Подпрограмма поиска новых ресурсов вызывается каждые *DI* мин и обновляет общее количество ресурсов.

5. Подпрограмма обновления информации о ресурсах вызывается каждые *MI* мин и обновляет количество ресурсов, занятых локальными очередями.

6. При генерации отчета вычисляются показатели эффективности модели (средняя задержка в очереди, среднее число задач в очереди, коэффициент занятости ресурсов, число задач, получивших обслуживание) и выводятся в файл отчета.

Код программы можно найти на сайте [15].

3.5. *Анализ выходных данных моделирования.* Проанализируем некоторые результаты компьютерного моделирования системы массового обслуживания *M/M/M/M*(*SI,DI,MI,PI,DC*), полученные в ходе экспериментов. Моделирование проводилось при следующих параметрах конфигурации системы:

− среднее время между поступлениями − 0,1 мин;

− среднее время обслуживания − 10 мин;

− среднее число ресурсов − 100 единиц;

− среднее число локально занятых ресурсов − 20 единиц;

– *SI* = 0,1 мин;

– *DI* = 15 мин;

– *MI* = 5 мин;

– *PI* = 0,1 мин;

– *DC* = 15 задач.

В процессе моделирования менялся один из параметров при фиксированных остальных.

При увеличении параметра *SI* коэффициент использования ресурсов и общее число обслуженных задач уменьшается (рис. 1). Отсюда следует, что оптимальным значением параметра *SI* является его наименьшее допустимое значение.

Как следует из рис. 2, 3, параметры *DI* и *MI* практически не оказывают влияния на эффективность модели.

Анализ рис. 4 показывает, что увеличение параметра *PI* ведет к уменьшению коэффициента использования ресурсов и общего числа задач, получивших обслуживание. Поэтому оптимальным значением параметра *PI* является его наименьшее допустимое значение.

На рис. 5 видно, что малые значения параметра *DC* уменьшают эффективность модели. Следовательно, значение этого параметра должно быть достаточно большим.

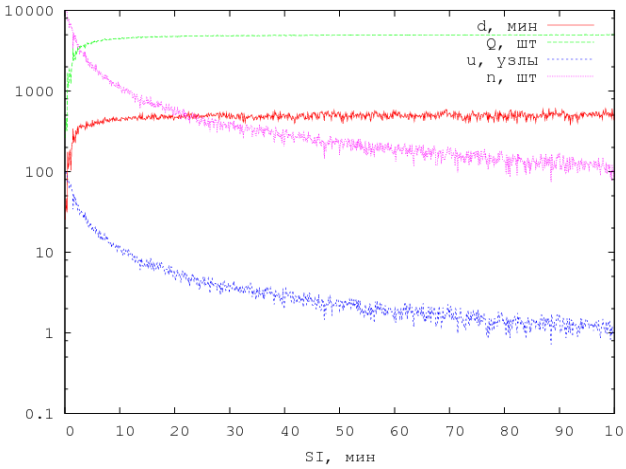

Рис.1. Зависимость показателей эффективности от параметра *SI*

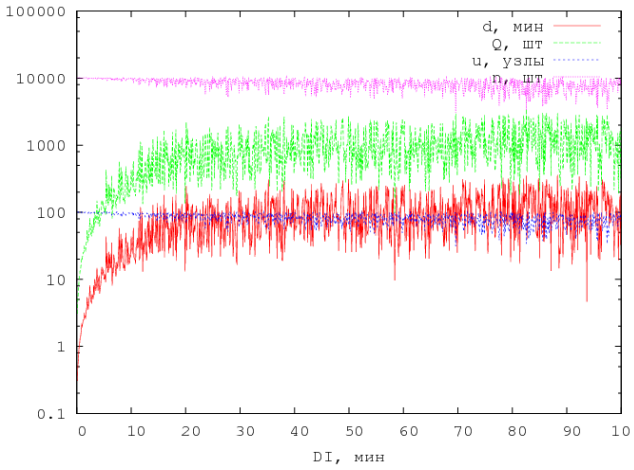

Рис. 2. Зависимость показателей эффективности от параметра *DI*

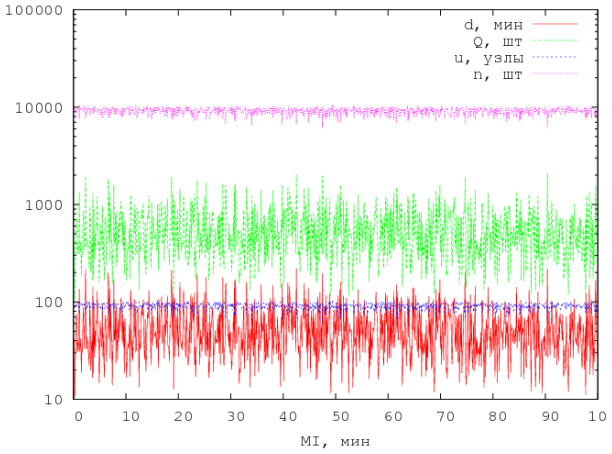

Рис. 3. Зависимость показателей эффективности от параметра *MI*

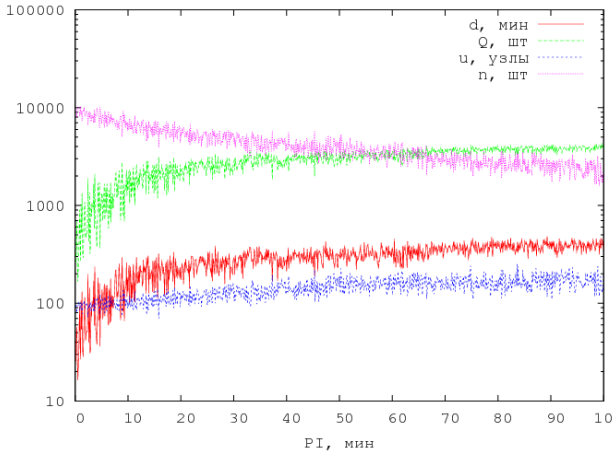

Рис. 4: Зависимость показателей эффективности от параметра *PI*

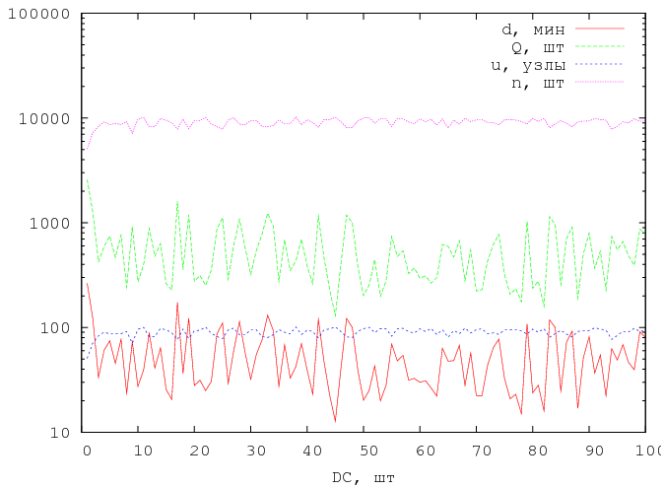

Рис. 5. Зависимость показателей эффективности от параметра *DC*

Таким образом, результаты моделирования показывают, что из пяти рассмотренных параметров наибольшее влияние на показатели эффективности, а значит, и на эффективность имитационной модели оказывают параметры *SI*, *PI* и *DC*. Коэффициент использования ресурсов растет с уменьшением периода между двумя итерациями планирования (*SI*), а также с сокращением периода опроса ресурсов о состоянии выполняемых на них задач (*PI*). Малые значения параметра *DC* значительно снижают эффективность модели, так как за одну итерацию планирования обрабатывается меньшее количество задач, чем доступно ресурсов.

В заключение следует отметить, что результаты, полученные при имитационном моделировании, совпадают с результатами, полученными при проведении эксперимента на базе рассмотренной выше аналитической модели. Это свидетельствует о том, что построенная дискретно-событийная модель соответствует реальной физической модели, а значит, модель и ее результаты являются адекватными.

Заключение. Работа посвящена подходам к математическому моделированию процесса диспетчеризации задач в Grid-среде. Рассмотрены два из возможных методов моделирования − аналитический и дискретно-событийный (имитационное моделирование). Приведены результаты компьютерного моделирования каждого из представленных подходов.

В отличие от аналитической модели, дискретно-событийный подход не требует больших вычислительных затрат на свою реализацию. Так, примерно 6000 запусков модели (в каждом запуске имитируется обслуживание 1000 задач) выполнялось всего за 2 минуты (CPU Pentium 4 3.00 ГГц), а сбор входных данных для аналитической модели занял несколько суток. В то же время аналитический подход более приближен к реальности, чем дискретно-событийный, поскольку опирается на практические данные. Поэтому для моделирования процесса диспетчеризации задач целесообразно использовать оба подхода: дискретно-событийный как теоретическую модель, аналитический подход − как практическую проверку этой модели.

Список литературы

1. TOP500 Supercomputing Sites [Electron. resource] / http://www.top500.org.
2. Human Genome Project [Electron. resource] / http://www.ornl.gov/sci/techresources/Human_Genome/home.shtml.
3. SETI@home [Electron. resource] / http://setiathome.berkeley.edu/.
4. distributed.net: Node Zero [Electron. resource] / http://http://distributed.net.
5. FOSTER IAN. What is the Grid? A three point checklist. Argonne National Laboratory and University of Chicago. July 20, 2002.
6. EGEE JRA1 Workload Management - WMS Architecture overview [Electron. resource] / http://egee-jra1-wm mi.infn.it/egee-jra1-wm/wms.shtml.
7. GridWay Metascheduler: Metascheduling technologies for the Grid [Electron. resource] / http://gridway.org
8. VENUGOPAL S., BUYYA R., WINTON L. A Grid service broker for scheduling distributed data-oriented applications on global grids [Electron. resource] / http://www.gridbus.org/papers/gridbusbroker.pdf.
9. GRID МГУ [Electron. resource] / http://grid.pp ru.
10. Collectd − the system statistics collection daemon [Electron. resource] / http://collectd.org.
11. Ganglia Monitoring System [Electron. resource] / http://ganglia.sourceforge.net.
12. Википедия [Электрон. ресурс] / http://ru.wikipedia.org.

13. LAW A., KELTON W. Simulation modeling and analysis. S. n.: McGraw-Hill, 2000.
14. MARSE K., ROBERTS S. D. Implementing a portable FORTRAN uniform (0, 1) generator. Simulation 1983;41:135-9.
15. Several-resource queueing system, fixed run length [Electron. resource] / http://grid.pp ru/git/?p=serd/chap1_c/.git;a=summary.
16. ГОРДИЕНКО А. А. Метапланирование ресурсов в Grid-системах.

Валерий Александрович Васенин − д-р физ.-мат. наук, проф. механико-математического факультета Московского гос. университета им. М. В. Ломоносова; тел.: (916) 503-46-34; e-mail: vasenin@msu.ru

Денис Александрович Сериков − аспирант механико-математического факультета Московского гос. университета имени М. В. Ломоносова; тел.: (916) 647-39-70; e-mail: serd@mexmat.net