УДК 681.324

М.А. Волк, М.А. Филимончук, Т.В. Филимончук

*Харьковский национальный университет радиоэлектроники, Харьков*

# МОДУЛЬ РАСПРЕДЕЛЕНИЯ ЗАДАНИЙ В GRID-СИСТЕМАХ

*В работе представлена модель модуля распределения заданий на вычислительные ресурсы GRID-систем. Разрабатываемый модуль поддерживает возможность проведения серии экспериментов, состоящей из последовательных запусков исследуемой модели с изменением некоторых параметров при каждом последующем запуске. Это позволяет в рамках одного эксперимента просмотреть динамику изменения эффективности системы и определить узкие места.*

***Ключевые слова:*** *методы распределения, имитационная модель, многокритериальная оптимизация, поток заданий, множество ресурсов, приоритет заданий.*

## Введение

GRID – это согласованная, открытая и стандартизованная система, которая обеспечивает гибкое, безопасное, скоординированное разделение ресурсов в рамках виртуальной организации [1]. В настоящее время глобальные вычислительные сети (GRID) позволяют решать вычислительные задачи большого объема для различных областей, таких как наука, бизнес, техника, медицина и др. Такие сети строятся на основе разнородных вычислительных ресурсов различных организаций, и для решения поставленной задачи позволяют одновременно использовать большое число вычислительных ресурсов. Однако недостаточно собрать ресурсы в одну сеть, необходимо эффективно использовать эти ресурсы по назначению. Т.к. ресурсы принадлежат различным организациям, то каждый поставщик устанавливает перечень правил управления ими (т.е. имеются ограничения использования их по времени, определена стоимость использования конкретного ресурса для различных категорий пользователей). Таким образом, возникает необходимость разработки системы управления ресурсами, которая нацелена на оптимизацию отношений между поставщиками заданий и владельцами ресурсов в соответствии с выбранными ими стратегиями.

Важное место в системе GRID занимают планировщики заданий, которые создают расписание использования ресурсов, учитывающее интересы участников GRID на основе определенного набора правил, ограничений и соглашений. В настоящее время существует множество систем управления и распределения заданий (Portable Batch System, Condor, MAUI Scheduler и др.) [2]. Однако планировщики, реализованные в данных системах, не предоставляют эффективного способа распределения заданий, с их помощью пользователь может лишь воспользоваться одним простым алгоритмом, который запускается при условии, что все задания и ресурсы системы приведены к заданному заранее общему описанию. Задания, поступающие на вход планировщика, являются разнородными, что создает дополнительные сложности при их распределении, т.к. на данный момент отсутствует универсальный по эффективности метод, который мог бы распределять любые классы задач, учитывая при этом все особенности имеющихся доступных ресурсов [3]. Наличие значительных трудностей и специфических особенностей решения задач распределения породило большое количество методов и алгоритмов, обзоры которых содержатся в ряде статей и монографий [4 – 6]. В связи с этим разработка модуля, распределяющего задания, которые поступают в систему GRID, на ресурсы является актуальной проблемой. Данный модуль позволит оценивать поведение GRID системы при изменяющихся условиях, а в дальнейшем и оптимизировать стратегию управления потоками заданий.

Однако использование отдельных компонентов GRID-системы – очень затратное, поэтому, вначале следует построить модель и исследовать ее свойства. Для этих целей подходит имитационное моделирование, которое является мощным инструментом для исследования поведения реальных систем. Оно основывается на том, что математическая модель воспроизводит процесс функционирования во времени элементарных событий, которые протекают в системе с сохранением логики их взаимодействия. Очень часто реальные эксперименты невозможно осуществить из-за финансовых или физических препятствий, тогда единственным решением является построение компьютерной модели [7]. Такая модель позволяет проводить вычислительные эксперименты, целью которых является сбор, анализ и интерпретация результатов моделирования, а также сопоставление полученных данных с реальным по-

ведением изучаемого объекта.

**Цель статьи** – разработка структуры программного обеспечения, имитирующего работу модуля распределения заданий на множество ресурсов. Данный модуль должен позволять находить наиболее подходящие для каждого задания ресурсы на основе сгенерированных требований, предоставлять возможность проведения экспериментов для различных методов распределения с возможностью сохранения исходных данных и результатов распределения для дальнейшего анализа.

## Среда моделирования GRASS

В Харьковском национальном университете радиоэлектроники ведутся работы по созданию среды моделирования GRASS (GRid Advanced Simulation System) [8], которая позволяет воспроизводить все процессы, происходящие в реальной GRID-системе. Она позволяет проводить эксперименты, с помощью которых можно проверять идеи для понимания того, как реальная система будет вести себя в различных ситуациях. Среда GRASS реализована на основе динамически подключаемых плагинов (plug-in), которые представляют собой независимые программные модули [9]. Каждый модуль выполняет свою узкоспециализированную задачу, обращаясь при необходимости к другим модулям системы. GRASS является макетом для отладки разрабатываемого модуля распределения заданий, который в дальнейшем может быть включен в реальную систему. Структуру среды моделирования можно разделить на 3 составные части [9]: программное обеспечение (ПО) пользователя, ПО виртуальной организации и ПО ресурса.

ПО пользователя позволяет производить мониторинг состояния системы, а также управление ее работой (добавлять и удалять задания из очереди, балансировать нагрузку между имеющимися ресурсами, выбирать алгоритм распределения и резервирования ресурсов и т.д.).

ПО виртуальной организации является центральным компонентом среды, обеспечивая контроль над подключенными ресурсами системы и очередью поступающих заданий, сбор и обработку статистики по ее работе. На рис. 1 показана схема взаимодействия подсистем среды моделирования GRASS с разрабатываемым модулем распределения заданий. Модуль распределения заданий (Algoritm Loader) является частью ПО виртуальной организации, главной задачей которого является загрузка алгоритмов распределения при возникновении в системе событий, требующих перераспределения заданий по ресурсам (например, добавление нового задания или ресурса, отключение существующего ресурса и т.д.).

ПО ресурса обеспечивает управление ресурсом, а также собирает и предоставляет другим компонентам информацию о нем (например, количество свободной памяти, среднюю загруженность ресурса и т.д.).

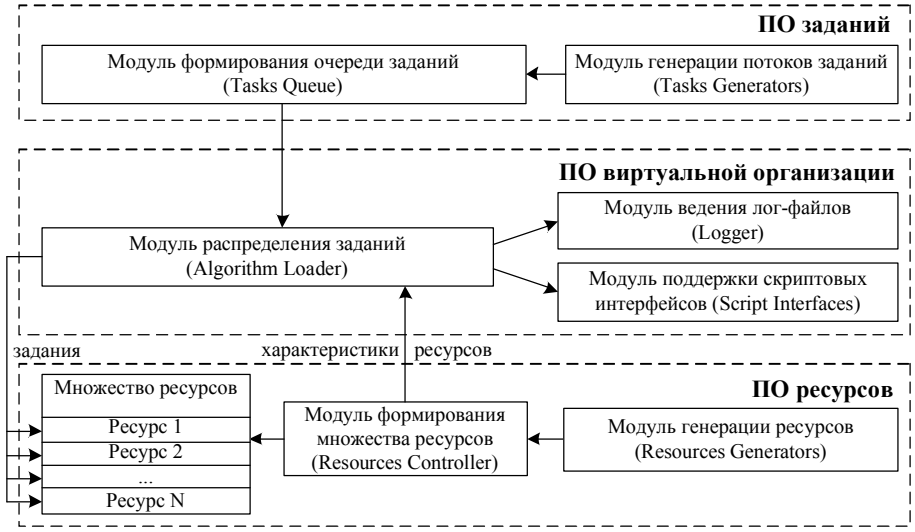


Рис. 1. Схема взаимодействия модулей среды моделирования GRASS

Рассмотрим кратко основные цели модулей среды моделирования GRASS.

Модуль генерации потоков заданий (Tasks Generators) выполняет моделирование входного потока заданий, поступающего в систему (может либо считываться из заранее записанного файла, либо генерироваться в соответствие с заданным законом распределения).

Модуль формирования очереди заданий (Tasks Queue) моделирует очередь заданий в GRASS, обеспечивает хранение и предоставление доступа к ней другим модулям системы, а также ведет статистику нахождения заданий в очереди.

Модуль генерации ресурсов (Resources Generators) выполняет моделирование множества имеющихся в наличии ресурсов: подключение новых и отключение существующих (моделирование неисправностей), а также изменение их характери-

стик во времени (например, количество свободной памяти или среднюю загрузку процессора).

Модуль формирования множества ресурсов (Resources Controller) обеспечивает доступ к имеющимся в наличии ресурсам моделируемой GRID-системы. Модуль ведения лог-файлов (Logger) предоставляет возможность гибкого и централизованного ведения логов для всех плагинов среды моделирования, а модуль поддержки скриптовых интерфейсов (Script Interfaces) позволяет создавать объекты обработки скриптов с настроенным окружением для взаимодействия с внутренними компонентами и модулями GRASS.

## Разработка концептуальной модели

Разрабатываемая модель строится на основе трех множеств: множестве ресурсов R, множестве заданий Z и множестве методов Q, т.е. $G = \{R, Z, Q\}$. Множества ресурсов и заданий моделируются вне модуля, однако в ходе эксперимента создается имитация динамики поведения GRID сети: происходит добавление вычислительных ресурсов, формируется поток заданий, поступающих в очередь, с которой в дальнейшем будет работать модуль. Множество методов распределения [4], которыми оперирует система моделирования GRASS, описывается соотношением $Q_k = \{mn_k, lp_k\}$, $\forall k = 1..k$, где параметр mn определяет алгоритм распределения заявок по ресурсам, а lp перечень параметров, учитываемых при распределении. Каждый из представленных в среде методов использует для распределения определенное заранее количество параметров.

В ходе разработки модуля были приняты следующие утверждения. Задание – это пакет задач, объединенных определенной тематикой, каждая задача представляет собой исполняемую программу. Задания, поступающие в GRID-систему, образуют поток $\{Z_i, i = 1, 2, ..., M\}$, где M – количество заданий, каждое из которых содержит ряд параметров, необходимых для их запуска в данной системе: $Z_i = \left\{ar_i^z, os_i^z, pc_i^z, ps_i^z, ms_i^z, dc_i^z, pr_i, ca_i, bw_i, rt_i\right\}$, $\forall i = 1..M$. Процессор в каждый момент времени может выполнять только одну задачу. Следовательно, если пакет задания требует для выполнения 32 процессора, то подразумевается, что в пакете 32 задачи, которые могут быть либо выполнены на одном ресурсе, который содержит 32 процессора (или больше), либо на нескольких ресурсах, удовлетворяющих требованиям всего пакета задания. Все задачи, которые входят в задание, запускаются одновременно, и если задание разбивается на несколько ресурсов, то запуск осуществится одновременно для всех задач только в том случае, когда для всех задач будут подобраны требуемые ресурсы.

Ресурсы GRID-системы также образуют поток $\{R_j, j = 1, 2, ..., N\}$, где N – число ресурсов в GRID-системе. Любой ресурс, поступающий в GRID систему, описывается рядом характеристик:

$$Rj = \left\{ar_j^r, os_j^r, pc_j^r, ps_j^r, ms_j^r, dc_j^r, bw_j\right\}, \ \forall j = 1..N .$$

В ходе эксперимента множество ресурсов $R_j$ может динамически изменяться, вследствие удаления/добавления ресурса в систему либо в ходе изменения характеристик ресурса. В роли ресурсов могут выступать как кластеры, так и отдельные рабочие станции, которые предоставляют пользователю доступ к оперативной и виртуальной памяти, дисковому пространству, а также процессорам вычислительного ресурса. Не последнее место отведено пропускной способности канала связи, так как часто для решения конкретной задачи могут потребоваться данные, которые рассчитываются другой задачей на удаленном ресурсе, и если пропускная способность канала будет низкой, то соответственно увеличится время выполнения задания. Это повлечет за собой сдвиг запуска всех последующих заданий из очереди. Таким образом, любая вычислительная система может быть рассмотрена как потенциальный вычислительный ресурс GRID-системы, при условии наличия специализированного ПО, которое позволяет сделать ресурс доступным для GRID.

Поясним заданные параметры ресурсов и потока заданий. Так как формирование множества ресурсов, на котором может быть запущено задание, происходит последовательно в несколько шагов, что позволяет сократить время моделирования, было принято ввести разбиение параметров на две категории: качественные и количественные. Вначале на первом шаге с помощью качественных параметров происходит формирование множества ресурсов, на которых возможно запустить выбранное задание, и отсекают ресурсы не пригодные для запуска с помощью качественных параметров, на втором шаге сформированное ранее множество подвергается проверке, которая находит свободные ресурсы для распределения.

К качественным параметрам следует отнести ar (architecture) – архитектуру процессора и os (operating system) – операционную систему. Все остальные заявленные в системе параметры являются количественными: pc (processor count) – количество процессоров, ps (processor speed) – быстродействие процессоров, ms (memory size) – объем оперативной памяти, dc (disk capacity) – доступный объем винчестера, ca (coefficient of association) – коэффициент связности задач в задании, pr (priority) – приоритет задания, bw (bandwidth) – пропускная способность канала связи для организации обмена между задачами, rt (run time) – время выполнения задания.

## Задачи, решаемые модулем распределения заданий

### *Задача распределения заданий на ресурсы*

Задача распределения заданий на ресурсы – основная для данного модуля и заключается в созда-

нии оптимального плана распределения заданного перечня заданий на определенное ранее множество ресурсов. Задача распределения в общем случае может быть представлена в виде полного ориентированного двудольного графа (рис. 2).

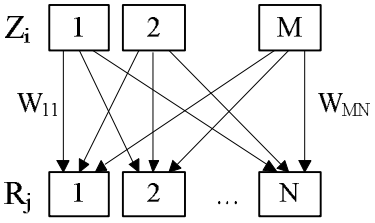


Рис. 2. Граф задачи распределения заданий

Весами дуг графа $W_{ij}$ (ограничениями на выполнение) здесь выступают характеристики, заданные пользователем, а также параметры вычислительных ресурсов: количество процессоров, объем памяти, скорость процессора, объем жесткого диска, пропускная способность канала связи, а также тип операционной системы и архитектура процессоров. Данный перечень параметров может быть дополнен по требованию пользователя. Все это приводит к многообразию вариантов решения поставленной задачи распределения, а, следовательно, к сложности определения оптимального решения.

Решение задачи распределения потока заданий модулем распределения сводится к последовательности действий:

– поток заданий $Z_i$ должен быть распределен на множество ресурсов $R_j$, используя один из методов распределения множества $Q_k$, учитывая все требования, выставляемые поставщиками заданий;

– если задание не может быть распределено (нет пригодных ресурсов на выполнение в системе), происходит отказ в его обслуживании;

– после распределения задание $Z_i$ получает ресурс $R_j$ для своего выполнения, и происходит определение параметров $\{t_b, t_e\}$, где $t_b$ – время запуска задания $Z_i$ на ресурсе $R_j$, а $t_e$ – время окончания задания. Данные параметры необходимы для дальнейшего процесса распределения, т.к. позволяют осуществлять предварительное планирование в наиболее известных алгоритмах распределения.

***Задача разбиения задания***

Очень часто может возникнуть ситуация, когда задание не может быть выполнено на одном ресурсе, и поэтому оно должно быть распределено на ряд ресурсов, которые подходят по заданному пользователем перечню параметров. Однако следует учитывать характер пакета задания, а именно, как связаны между собой задачи. Когда коэффициент связности равен нулю, это означает, что задачи в пакете не связаны между собой, и нет никаких проблем при их распределении, т.е. данные задачи могут быть распределены на разные вычислительные ресурсы. Если же коэффициент связности находится в пределах $0 < ca \le 1$ – задачи связаны между собой, и чем выше значение $ca$, тем выше интенсивность обмена данными между отдельными задачами пакета задания. Следовательно, в этом случае при распределении следует учитывать их расположение, т.е. распределение должно быть направлено на то, чтобы задачи пакета задания были распределены на как можно меньшее число ресурсов. Такое распределение является одним из главных шагов этапа оптимизации.

## Описание функционирования модуля распределения заданий

При использовании GRID-систем для распределения ставится задача эффективного использования элементов сети, что подразумевает либо увеличение пропускной способности сети, либо суммарное уменьшение времени выполнения пакета заданий, либо уменьшение стоимости вычислений. Как было сказано выше, каждое задание характеризуется рядом параметров, которые используются в зависимости от выбранного метода распределения. В один и тот же момент времени системой может быть обслужено только одно задание, т.е. пока оно не поступит на выполнение – переход к следующему не осуществляется. В настоящее время для моделирования пользователь может выбрать только один метод для каждого эксперимента. Однако в модуле предусмотрена возможность сохранения последовательности заданий и перечня ресурсов с заданными характеристиками в файлы, что позволяет повторить эксперимент для другого метода распределения с теми же данными, т.е. сохраняется чистота эксперимента.

Рассмотрим взаимодействие отдельных плагинов среды моделирования GRASS, а также ввод параметров для проведения экспериментов.

Все параметры настройки среды содержатся в конфигурационном файле plugins.xml, который расположен в каталоге «config». Файл plugins.xml описывает взаимосвязи между именами модулей в системе и названиями файлов и библиотек [5], а также пути к конфигурационным файлам.

На рис. 3 представлена структурная схема модуля распределения потока заданий на вычислительные ресурсы, которая содержит 4 отдельных блока и для корректной работы использует потоки заданий и ресурсов, которые формируются вне модуля, по заданным заранее правилам.

Блок методов распределения содержит ряд модулей, имитирующих работу различных алгоритмов распределения: First-Come First-Served (FCFS), Last In First Out (LIFO), Highest Priority First (HPF), метод обратного заполнения (Backfill), метод линейного программирования (Simplex), а также метод


Рис. 3. Структура модуля распределения потока заданий на вычислительные ресурсы

распределения по освободившемуся ресурсу (Smart). Каждый из представленных методов использует свой набор параметров для распределения.

Методы FCFS и LIFO не используют дополнительные параметры, т.к. это простейшие политики, которые служат для наглядного сравнения с другими более рациональными методами.

После того как были запущены модули Tasks Generators и Resources Generators, начинает свою работу блок приема и обработки сигналов о событиях, происходящих в очереди и с ресурсами, который отслеживает события добавления нового задания в очередь и удаления его из нее, а также добавления и удаление ресурсов. Если произошло одно из этих событий, то происходит анализ состояния очереди.

Для корректной работы среды GRASS был принят ряд правил, одним из которых является то, что задания по мере продвижения в системе могут изменять свое состояние. Так, задания в очереди характеризуются одним из 8 значений: Failed (задание завершено неуспешно), Finished (задание успешно завершено), Invalid (некорректное задание или ошибка работы), Running (задание запущено на ресурсе), Timeout (задание снято с выполнения по таймауту), Waiting (задание ожидает распределения), Cancelled (выполнение задания отменено из-за отсутствия в системе заявленных ресурсов), Planned (задание запланировано для запуска при выборе алгоритма Backfill). По ходу поступления заданий в систему происходит проверка состояний заданий в очереди, и если количество заявок в состоянии Waiting равно или превышает параметр tasks_count, то приходит сообщение о том, что эти задания могут быть обслужены данной системой и согласно выбранному методу планирования произойдет распределения заявок на ресурсы.

Далее блок распределения заявок по ресурсам формирует расписание распределения в несколько этапов. На первом шаге осуществляется проверка двух основных условий. Первое, удовлетворяет ли ресурс минимальным системным требованиям, и второе – доступны ли такие ресурсы для выполнения задачи. При моделировании любое поступившее в систему задание будет распределено, исключение составит случай, если в системе нет ресурсов, пригодных к запуску. Если на момент запуска задания ресурсы заняты, то задание будет ожидать в очереди освобождение необходимых ресурсов, после чего займет ресурсы под свои требования.

Важным параметром эксперимента является переменная strict , которая позволяет проведение эксперимента в двух направлениях: с учетом распараллеливания задания и без него. Если strict = 1 , то происходит вызов метода checkQueueStrict(), который заключается в прохождении по всем заданиям, которые пребывают в очереди в состоянии Waiting. Для каждого задания сравнивается перечень требований и множество ресурсов. Если в системе есть ресурсы, удовлетворяющие требованиям задания, то оно остается в очереди и получает значение состояния Waiting. Если же таких ресурсов нет, независимо от их занятости в данный момент времени, то задание переходит в состояние Cancelled, что означает невозможность его распределения и выполнения в заданной конфигурации. Если strict = 0 , то вызывается метод checkQueueBase(), в ходе которого осуществляется проверка, аналогичная методу checkQueueStrict(), только при проверке количества требуемых процессоров учитывается возможность распределения задания по нескольким ресурсам. Т.е. если необходимое количество процессоров может быть обеспечено несколькими ресурсами, то заявка остается в состоянии Waiting.

На втором шаге происходит оценка плотности заполнения очереди $P_з = К_и / К_т$ , где $К_и$ – количество заданий в очереди на данный момент времени, $К_т$ – требуемое количество заданий для начала распределений (данный параметр задается перед началом эксперимента в файле plugins.xml). Если

$P_з \geq 1$, то происходит переход на выбранный ранее метод распределения и в нем запускается метод run().

И на третьем шаге, в зависимости от выбранного метода распределения, формируется список заданий и ресурсов, на которые они будут распределяться. Эти списки передаются в плагин Distributor в метод runTask(), в котором выполняется планирование таймаутов, исходя из времени, которое требуется на выполнение задания, и транспортной задержки для ресурса. Заявки, которые были распределены, автоматически удалятся из очереди, а ресурсы принимаю состояние «занят».

Блок транспортировки задания на ресурс осуществляет доставку задания на требуемый ресурс или множество ресурсов, присваивает заданию состояние Running и резервирует ресурс или ресурсы на время выполнения задания. По истечении времени, запланированного на выполнение задачи, в плагине Distributor запускается метод terminateTask(), который отвечает за удаление задания из очереди и возвращение ресурсов для дальнейшего использования в системе. В ходе выполнения эксперимента осуществляется мониторинг (сбор статистики в лог файлах) и в дальнейшем полученная информация может быть использована для отладки среды моделирования GRID.

## Заключение

В настоящее время перед GRID-системой ставится задача эффективного распределения заданий по имеющимся ресурсам. Она заключается в нахождении такого набора ресурсов, при котором критерии оптимальности примут экстремальное значение, т.е. система будет функционировать эффективно.

С помощью разработанного модуля имеется возможность оценить эффективность методов распределения и выбрать наилучший из них для дальнейшей работы, т.е. имеется возможность на заданном множестве ресурсов и заданий рассмотреть всевозможные варианты. В дальнейшем собранная статистическая информация может быть использована для обучения системы, что позволит при поступлении определенного задания производить распределение без дополнительных вычислений.

## Список литературы

*1. I. Foster and C. Kesselman (eds), "The Grid 2: Blueprint for a New Computing Infrastructure", Morgan Kaufmann Publishers, 2004.*

*2. Волк М.А. Анализ архитектур и характеристик современных планировщиков заданий в GRID-системах / М.А. Волк, М.А. Филимончук, К.В. Дема // Збірник наукових праць ХУПС. – Х.: ХУПС, 2011. – Вип. №2(28). – С. 123-128.*

*3. Волк М.А. Методы распределения ресурсов для GRID-систем / М.А. Волк, М.А. Филимончук, Р.Н. Гридель // Збірник наукових праць ХУПС. – Х.: ХУПС, 2009. – Вип. (19). – С. 100-104.*

*4. Листровой С.В. Об использовании гарантированных прогнозов в методах решения задач булевого программирования на основе рангового подхода / С.В. Листровой, О.Н. Симашкевич // Электрон. моделирование. – 2003. – Т. 25, №4. – С. 89-103.*

*5. Пономаренко В.С. Методы и модели планирования ресурсов в GRID-системах: Монография / В.С. Пономаренко, С.В. Листровой, С.В. Минухин, С.В. Знахур. – Х.: ВД „ІНЖЕК", 2008. – 408 с.*

*6. Юрич М.Ю. Математическая модель системы диспетчеризации задач / М.Ю. Юрич // Науковий вісник Чернівецького університету. – Чернівці, 2009. – Випуск 479. – С. 134-138.*

*7. Грушин Д.А. Система моделирования Grid: реализация и возможности применения / Д.А. Грушин, А.И. Поспелов // Труды Института системного программирования РАН. – 2010. – Т. 18. – С. 243-260.*

*8. Волк М.А. Архитектура имитационной модели GRID-системы, основанная на подключаемых модулях / М.А. Волк, Р.Н. Гридель, А.С. Горенков // Системи обробки інформації. – Х.: ХУПС, 2010. – Вип. 1(82). – С. 17-20.*

*9. Волк М.А. Структура программного комплекса имитационного моделирования элементов GRID-систем для научных исследований / М.А. Волк // Системи обробки інформації. – Х.: ХУПС, 2009. – Вип. 3(77). – С. 125-128.*

*Поступила в редколлегию 11.01.2012*

**Рецензент:** д-р техн. наук, проф. С.Г. Удовенко, Харьковский национальный университет радиоэлектроники, Харьков.

### МОДУЛЬ РОЗПОДІЛУ ЗАВДАНЬ У GRID-СИСТЕМАХ

М.О. Волк, М.А. Філімончук, Т.В. Філімончук

*У роботі представлена модель модуля розподілу завдань на обчислювальні ресурси GRID-систем. Модуль, що буде розроблений, підтримує можливість проведення серії експериментів, які складаються з послідовних запусків досліджуваної моделі зі зміною деяких параметрів при кожному наступному запуску. Це дозволяє в рамках одного експерименту переглянути динаміку зміни ефективності системи і визначити вузькі місця.*

***Ключові слова:*** *методи розподілу, імітаційна модель, багатокритерійна оптимізація, потік завдань, множина ресурсів, пріоритет завдань.*

### METHODS OF ALLOCATION OF RESOURCES FOR THE GRID-SYSTEMS

M.A. Volk, M.A. Filimonchuk, T.V. Filimonchuk

*In work the model of tasks distribution module computing resources of GRID-systems is presented. The developed module supports carrying opportunity out of experiments series. A series consists of consecutive starts of researched model with change of some parameters at each following start. It allows to see a system effectiveness change dynamics within the limits of one experiment and to determine bottlenecks.*

***Keywords:*** *distributing methods, simulation model, multicriterion optimization, stream of jobs, great number of resources, priority of tasks.*