УДК 681.51

С.В. Минухин, А.В. Коровин

Харьковский национальный экономический университет, Харьков

МОДЕЛИРОВАНИЕ ПЛАНИРОВАНИЯ РЕСУРСОВ GRID СРЕДСТВАМИ ПАКЕТА GRIDSIM

Рассмотрены принципы моделирования распределения и планирования ресурсов для решения задач GRID в пакете GridSim. Предложен эффективный метод планирования ресурсов, использующий подход на основе задачи о минимальном вершинном покрытии. Разработаны алгоритм и программное обеспечение на языке Java, реализующее этот метод. Проведена интеграция программы в среду пакета GridSim. Проведены экспериментальные исследования для различных значений параметров моделируемой GRID-среды в условиях применения существующих и предложенного метода планирования ресурсов GRID. Приведен сравнительный анализ полученных результатов, обосновывающий эффективность предложенного в работе метода.

Ключевые слова: *архитектура, кластер, минимальное вершинное покрытие, моделирование, планирование, ресурсы, рефакторинг, Alea 2, GRID-среда, GridSim.*

Введение

Современные технологии создания вычислительной техники подошли к рубежу, когда дальнейшее наращивание скорости производительности работы индивидуальных устройств становится практически невозможным. В связи с этим развитие современных технологий идет по пути, основанному на дублировании вычислительных устройств, которые параллельно могут работать над общей задачей, включая и распределенные вычисления. Сегодня разработчики программных систем используют параллелизм на всех уровнях, начиная от нескольких конвейеров суперскалярных процессоров, и заканчивая параллельно работающими вычислительными узлами в GRID-системе.

Последние работы в области GRID [1] позволяют приложениям использовать вычислительные ресурсы, принадлежащие различным организациям, распределенным географически. Одним из видов ресурсов GRID являются однородные многопроцессорные системы (кластеры), которые могут состоять из сотен или даже тысяч процессоров. Параллельное приложение для кластерной системы представляет собой несколько процессов, которые взаимодействуют друг с другом по сети. Таким образом, если пользователь сумеет эффективно распределить свою задачу между несколькими процессорами на узлах кластера, то он может получить выигрыш в скорости работы, пропорциональное числу процессоров.

Однако при массовом запуске параллельных приложений на кластерах нужно умело распоряжаться узлами кластера, стараясь распределить нагрузку максимально равномерно. Необходимо находить «зазоры» в расписании между «тяжелыми» задачами и стараться запускать между ними более «легкие», которые должны решаться на оставшихся свободных (или просто наименее загруженных) узлах. Решение задач, связанных с составлением расписания запусков пользовательских приложений, ложатся на систему управления кластером. Помимо составления расписания система должна обеспечивать надежность функционирования кластера. В идеале система должна «на лету» обнаруживать и обрабатывать изменение состава вычислительных узлов.

Однако, несмотря на то, что уже сейчас предлагаются средства создания GRID-инфраструктур, ставшие «де-факто» стандартными, существует ряд важных научных задач, в том числе и теоретических, без решения которых полномасштабное использования возможностей GRID-технологий невозможно. Одной из актуальных задач в настоящее время является эффективное управление планированием и распределением вычислительными ресурсами в распределенной среде.

С ростом числа ресурсных центров, входящих в распределенную инфраструктуру (ЦОД, кластеров), отсутствие хорошего планировщика, обеспечивающего управление потоком задач, значительно снижает эффективность использования всей GRID-инфраструктуры. При этом следует отметить, что для таких распределенных систем характерным является динамичное развитие, что делает невозможным решение задачи эффективного планирования “в статике” – один раз и навсегда.

С другой стороны, оптимизация алгоритмов управления распределенной средой на непосредственно уже существующей GRID-инфраструктуре затруднено и связано со значительными издержками и простоями ресурсных центров, а часто, в силу масштабности распределенной среды – невозможно. В связи с этим актуальной задачей является использо-

вание системы моделирования GRID-инфраструктуры, которая позволит адекватно оценивать ее поведение при изменяющихся условиях и на основе этого оптимизировать стратегии управления потоками задач, непосредственно распределения и планирования ресурсов..

Система моделирования может быть использована для оценки эффективности распределенной вычислительной среды в различных ситуациях, например: при изменении нагрузки: количества поступающих задач, их размерности, приоритета, периода поступления и т.д.; при отключении части вычислительных ресурсов или добавлении новых ресурсов; при увеличении количества передаваемых данных; при выходе из строя части коммуникационных каналов.

В настоящей работе рассматриваются проблемы моделирования загрузки вычислительного кластера для различных режимов обслуживания глобальной очереди GRID с использованием системы GridSim.

Основные принципы моделирования в GridSim

Проект GridSim [3] разрабатывается группой исследователей в лаборатории по изучению облачных и распределенных вычислений отдела информатики и компьютерных вычислений в университете Мельбурна, Австралия.

Эта платформа позволяет пользователям моделировать работу GRID-системы с возможностью симулирования характеристик ресурсов и вычислительных сетей при различных конфигурациях. Она предоставляет возможность моделировать поведение пользователей GRID-системы, вычислительных ресурсов и брокеров ресурсов (планировщиков). GridSim может быть использован исследователями, которые разрабатывают и повышают эффективность существующих алгоритмов планирования задач на вычислительных кластерах. С помощью GridSim можно проводить воспроизводимые эксперименты, которые сложно реализовать в настоящем окружении динамических GRID-систем.

Основные возможности GridSim заключаются в: моделировании различных характеристик ресурсов GRID-среды; моделировании различных политик планирования задач на узлах вычислительных кластеров – как уже реализованных (FCFS, Easy Backfill, Conservative Backfill), так и разработанных пользователями алгоритмов; использование данных о загрузке реальных кластеровдля проведения экспериментов; поддержка механизма аукциона для планирования задач; моделирование различных конфигураций вычислительной сети GRID-системы для различных топологий; моделирование региональных компонентов GRID информационных сервисов.

GridSim проектировался как многоуровневая система моделирования [3], возможности которой могут быть легко и значительно расширены (рис. 1). Добавление новых компонентов или уровней может быть осуществлено довольно просто. В частности, исследователи Национального университета Сингапура добавили в GridSim сетевой модуль, который основан на QoS (Quality of Service), сотрудники университета Любляны работают над модулем DataGrid. К тому же, такая многоуровневая архитектура включает модель вычислительного окружения GRID.

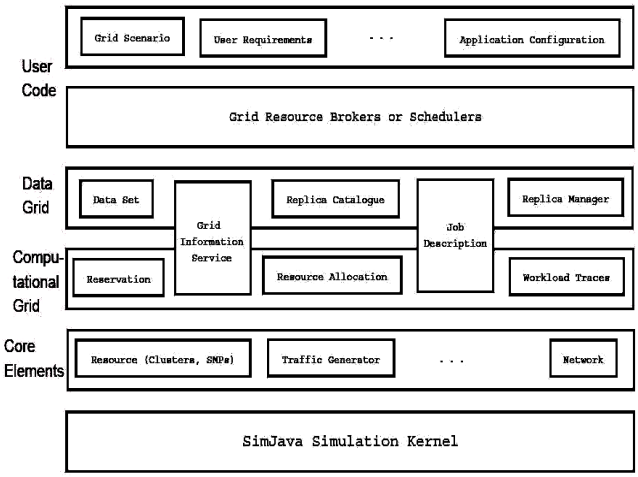

Рис. 1. Архитектура GridSim

GridSim основан на SimJava [6] – пакете для симулирования дискретно-событийных процессов, реализованном на языке Java. Поэтому на рис. 1 показано, что нижнем слоем управления занимается SimJava – обработкой внутренних событий и взаимодействия компонентов GridSim. Все компоненты GridSim взаимодействуют между собой через отправку сообщений, определенных в SimJava.

Второй слой моделирует основные компоненты распределенной инфраструктуры, а именно, ресурсы GRID, например кластеры, хранилища данных и сетевые соединения. Эти компоненты существенны при создании моделей с помощью GridSim.

Третий и четвертый слой концентрируются на моделировании и симуляции служб, специфичных для вычислений и технологии DataGrid. Некоторые из служб предоставляют функциональность, общую для всех видов GRID-систем, например, информацию о доступности ресурсов системы и управление назначением (планированием) задач. Для случая DataGrid управление задачами также включает в себя управление передачей информации между узлами хранения данных и вычислительными узлами. Службы управления файлами и данными также реализованы особым образом для DataGrid.

Пятый уровень содержит компоненты, которые помогают пользователям в реализации собственных

брокеров ресурсов и планировщиков заданий таким образом, что они могут проверить их собственные стратегии и алгоритмы.

Самый верхний уровень позволяет пользователям реализовывать их собственные сценарии работы и конфигурации системы для тестирования их собственных политик и алгоритмов.

Визуализация процесса моделирования и результатов

Отдельным и полезным является вопрос визуализации процесса моделирования. Хорошее средство для визуализации моделирования может сэкономить время, затраченное на анализ результатов моделирования. Многие выводы можно сделать лишь взглянув на графики или диаграммы (конечно, при условии что человек, который анализирует их понимает их природу и физический смысл). Конечно, для полноценной проверки результатов необходимо проводить различного вида статистический анализ, но для получения быстрого ответа на вопрос, например «Эффективно ли работает алгоритм планирования?», хорошее средство визуализации результатов будет незаменимым помощником.

Начиная с версии 5.0 в GridSim появился пакет parallel.gui, в котором содержится интерфейсы Visualizer и AbstractVisualizer, которые предназначены для визуализации процесса моделирования. Единственная реализация этих интерфейсов, которая идет в GridSim – это ParallelVisualizer, который позволяет из графического интерфейса пользователя запускать, останавливать эксперимент и переходить в режим пошагового выполнения эксперимента.

Согласно официальной документации GridSim [11], эта реализация должна использоваться только в отладочных целях, например для пошаговой проверки новых политик или алгоритмов планирования. Реальные эксперименты не предусматривают никаких визуальных инструментов. А интерфейсы Visualizer и AbstractVisualizer были изначально разработаны для другого симулятора под названием PajFit.

Работа симулятора Alea 2 [12] сконцентрирована на моделировании планирования загрузки вычислительного кластера в GRID-окружении. Симулятор Alea2 требует для работы GridSim версии не ниже 5.0. Возможности визуализации у Alea2 намного превосходят аналогичные у GridSim и PajFit. Он позволяет наблюдать изменение состояния экспериментальной GRID-инфраструктуры во время проведения эксперимента. Результаты выводятся в реальном масштабе времени в виде гистограмм, графиков и спектральных диаграмм. Анализировать ход выполнения эксперимента можно по таким критериям:

количество процессорных элементов: запрошенных, используемых и доступных;

загруженность кластеров в процентном соотношении по каждому часу и дню;

количество задач, которые ждут выполнения и выполняющихся в данный момент времени;

среднее использование процессорных элементов кластеров за каждый час;

процент отказов ресурсов GRID-системы.

Все результаты экспериментов, кроме отображения на графиках, сохраняются в текстовые файлы в формате CSV. После проведения экспериментов эти данные можно загружать в программные продукты, которые позволяют проводить последующий статистический анализ. Например, данные, сохраненные в формате CSV, могут быть легко преобразованы в другие форматы – XLS (Microsoft Excel), XML, STA (Statistica).

Полезной может оказаться возможность проведения экспериментов с использованием записей загруженности реальных GRID-инфраструктур в следующих форматах: Grid Workloads Format (GWF), Satndard Workloads Format (SWF) и MetaCentrum workload

В Alea2 реализован централизованный планировщик, который использует алгоритмы планирования, основанные на очередях, такие как FCFS, EasyBackfill, .

Планировщик способен обрабатывать ситуацию, когда задачи поступают на выполнение динамически во время проведения симуляции. В этом случае генерируемые графики будут менять свой вид, отображая текущее состояние экспериментальной инфраструктуры GRID. Все графики можно сохранять в файлы различных графических форматов (*.jpeg, *.bmp, *.png) для последующего использования. На рис. 2 приведена архитектура центрального планировщика Alea2.

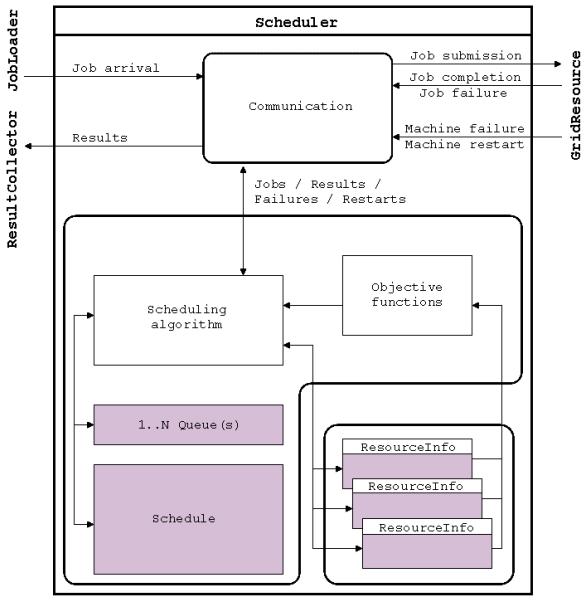

Рис. 2. Схема работы центрального планировщика Alea 2

Архитектура Alea 2 позволяет проводить подряд эксперименты на одних и тех же исходных данных (данных о поступающих задачах и доступных ресурсах) с использованием различных алгоритмов планирования. Таким образом, можно проводить сравнительный анализ работы различных алгоритмов планирования, их поведения в тех или иных ситуациях и разрабатывать сценарии оптимальной загрузки имеющихся ресурсов.

Моделирование различных политик для загрузки вычислительного кластера GRID

Выбор политики и алгоритма планирования заданий для вычислительного кластера оказывает большое влияние на загрузку ресурсов кластера, время решения задач и, как результат, – на эффективность использования этих ресурсов.

Целей планирования последовательности назначений работ по процессорам может быть несколько. Обычно стремятся минимизировать общее времени исполнения всего множества работ на многопроцессорной системе (кластере), однако возможны вариации, когда стратегия построена таким образом, что минимизация времени исполнения происходит косвенно и не гарантируется стратегией.

Для алгоритма планирования, в случае если планирование производится во время запуска программы или постоянно планировщиком заданий, всегда можно выделить время планирования процессов на многопроцессорной системе (кластере). Это время между получением планировщиком информации об освобождении процессора и непосредственным назначением работы на исполнение. При этом необходимо минимизировать это время. Обычно получение расписания, близкого к оптимальному, требует большого времени планирования и, наоборот, легко получить расписание, которое не будет оптимальным. Время планирования обычно зависит от количества работ. Важной характеристикой алгоритма планирования является характер данной зависимости. Алгоритм может строить хорошее расписание, но быть плохо масштабируемым к количеству планируемых работ на устройстве.

Существует ряд достаточно простых алгоритмом планирования, которые позволяют выбрать работу для назначения на освободившийся процессор из списка имеющихся работ – выполняется (обслуживается) первая по списку работа (First Come First Served, FCFS) или работа с максимальным приоритетом (Higest Priority First, HPF). Существуют политики, связанные с директивными сроками выполнения работ (EDF) или работы, которыее выполняются за кратчайшее время (JSF).

Такого рода алгоритмы используются в системах реального времени. Планирование процессов в операционной системе Linux в ядре, начиная с версии 2.6.8.1, осуществляется на основе модификации RR (Round Robin) алгоритма, где, по мнению создателей алгоритма, время планирования не зависит от числа процессов за счёт специальной системы подсчёта приоритетов.

Алгоритм FB (feedback) очередей с обратной связью использует *n* очередей, каждая из которых обслуживается в порядке поступления. Новая работа поступает в очередь с номером 0, затем, после получения кванта времени, она переходит в очередь со следующим номером и так далее после очередного кванта времени. Время процессора планируется таким образом, что он обслуживает непустую очередь с наименьшим номером. В методе FB каждая, вновь поступающая на обслуживание работа получает высокий приоритет и выполняется подряд в течение такого количества квантов времени, пока не появится новая работа. Если приход новой работы задерживается, то текущая работа не может проработать большее количество квантов времени, чем предыдущая работа. Данный алгоритм наиболее эффективно работает с большим числом коротких по времени работ. Основное преимущество очередей FB и RR по сравнению с алгоритмом EDF и алгоритмом кратчайшей по времени работы в том, что FB и RR не требуют предварительной информации о времени выполнения работ.

Для сравнения показателей эффективности работы алгоритмов планирования при различных условиях и режимах работы удобно использовать системы моделирования GRID-инфраструктуры.

Моделирование процесса формирования очереди на основе решения задачи о наименьшем вершинном покрытии

Рассмотрим задачу планирования работ при следующих условиях. Предполагается наличие многопроцессорной системы с n процессорными элементами, на которой необходимо решить m задач. Каждая задача описывается набором требований к ресурсам, а ресурсы, в свою очередь, обладают качественными характеристиками, которые могут удовлетворять требованиям заданий.

В данной работе предлагается алгоритм распределения заданий на кластеры GRID, основанный на решении задачи о наименьшем вершинном покрытии [1].

Существенное отличие исследуемого алгоритма от представленных в Alea 2, заключается в том, что он использует промежуточный пул фиксированного размера для распределения задач на ресурсы. Основная цель использования пула заключается в снижении затрат на сам процесс планирования. Алгоритм планирования запускается при выполнении одного из двух условий: размер пула достиг уста-

новленного размера или прошло заданное время с момента последнего распределения.

Задачи, находящиеся в пуле, представлены виде прямоугольной матрицы соответствий, в которой в строчки соответствуют номерам задач, а столбцы – номерам ресурсов. Если i-я задача может быть решена на j-м ресурсе, то в ячейке ij стоит значение 1. Метод находит наименьшее количество столбцов (ресурсов), которые могут покрыть (решить) все столбцы (задачи), находящиеся в пуле. После этого задачи распределяются между ресурсами GRID, вошедшими в наименьшее покрытие. Если ресурсы, вошедшие в наименьшее покрытие, полностью заполнены, а задачи в пуле еще остались, то распределяем их на другие свободные ресурсы. При этом оставшиеся задачи возвращаются обратно в пул. На рис. 3 приведен псевдокод работы алгоритма.

Для сравнения показателей работы предложенного алгоритма были выбраны следующие алгоритмы [11, 14], предоставляемые системой Alea 2: FCFS; EasyBackFill; Earliest suitable gap (ESG) .

После реализации алгоритма на языке Java необходимо интегрировать его в Alea 2. С точки зрения пользователя, Alea2 – средство для моделирования работы GRID-инфраструктуры при разных условиях (различных входных данных, различной конфигурации оборудования, различных топологий сети и различных алгоритмов планирования задач) и анализа полученных результатов (гораздо удобнее анализировать графическое представление результатов моделирования). Вместе с тем, с точки зрения программиста, который хотел бы расширить возможности данной системы, она весьма неудобна в использовании.

В исходных кодах системы Alea 2 повсеместно используются жестко закодированные целочисленные и вещественные константы, имена файлов с входными данными указаны прямо в коде, также были жестко закодированы размеры окон и их положение. В целом можно описать внутреннюю структуру кода как не ортогональную систему [6].

Поэтому, прежде чем интегрировать новый алгоритм, был проведен рефакторинг кода [8], в ходе которого целочисленные константы, соответствующие алгоритму планирования, были заменены на перечисления; а конкретные имена файлов с исходными данными были убраны из кода и загружались в качестве переменных окружения виртуальной Java-машины [8]. После этого часть кода, которая отвечает за планирование, стала более понятной и вносимые изменения в одной части программы не вызывали неожиданного поведения других ее частей.

Для добавления новых алгоритмов планирования был создан интерфейс DispatchingAlgorithm с единственным методом dispatch, который принимает два параметра: список заданий и список доступных ресурсов; метод возвращает список пар (задание + ресурс), который представляет собой результат планирования – назначение задач на ресурсы.

```
procedure TaskPlanningSMC(pool, resources)
 for each r ∈ resources
   do { for each t ∈ pool
        do { if r accepts t
             then { vsetR_r ← t
                    vsetT_t ← r

cover ← SearchCover(vsetR, vsetT)

 for each t ∈ vsetT
   do { for each r ∈ cover
        do { if vsetR_r accepts t
             then assign t to vsetR_r Id


procedure SearchCover(vsetR, vsetT)
 for each r ∈ vsetR
   do cols ← r

 for each t ∈ vsetT
   do if cols_t > 0
   then rows ← t

 coverIsFound ← false
 while true
     { repeat
        for each r ∈ rows
          do { if length(r) == 1
               then { cover ← r
                      rows ← r × rows
                      if length(rows) == 0
                        then return (cover)
                      coverIsFound ← true
        until coverIsFound

        repeat
         coverIsFound ← false
         for each c ∈ cols
          do { if length(c) == 1
               then { cover ← c
                      cols ← c × cols
                      if length(cols) == 0
                        then return (cover)
                      coverIsFound ← true
               else { erase rows_c
                      erase cols_c
        until coverIsFound
        if length(rows) == 0
          then return (cover)
        maxCover ← Max(rows, vsetR)
        cover ← maxCover
        rows ← maxCover × rows
```

Рис. 3. Псевдокод работы алгоритма, основанного на решении задачи о наименьшем вершинном покрытии

Для возможности разработки алгоритмов планирования вне зависимости от среды моделирования были также разработаны интерфейсы Task и Resource, представляющие собой абстракции задания и ресурса соответственно. Для повторного использования разработанного алгоритма в другой среде моделирования необходимо будет реализовать интерфейсы Task и Resource для этой системы, а реализацию алгоритма менять не придется.

В пакете impl находятся реализации следующих интерфейсов:

MinimalVertexCover – реализация алгоритма планирования по методу нахождения наименьшего вершинного покрытия;

GridSimResource – реализация абстракции ресурса, транслирующая все вызовы классу ресурсов из GridSim;

GridSimTask – реализация абстракции задачи, транслирующая все вызовы классу заданий из GridSim.

Выбор алгоритма планирования был изменен согласно методу рефакторинга «Замена условного оператора полиморфизмом (Replace Conditional with Polymorphism)» [7] и вынесен в класс AlgorithmFactory, реализованный согласно шаблону проектирования Фабрика [14], что сделало код планирования более гибким и расширяемым.

Для сравнения эффективности работы предложенного и имеющихся алгоритмов планирования был выбран показатель средней загрузки ресурсов за весь процесс моделирования для каждого режима обслуживания (алгоритма планирования). Для проведения моделирования в качестве исходных данных были взяты данные о загрузке французского грида AuverGrid [15] в период с 15 декабря 2005 г. по 7 февраля 2006 г., предоставляемые некоммерческой организацией «The Grid Workloads Archive». Данные представляют собой записи о загрузке кластеров GRID, представленные в формате GWF[16].

Результаты экспериментальных исследований

При проведении экспериментов были использованы следующие алгоритмы: FCFS – First Come First Served; MC – Minimal Vertex Cover (решения задачи о наименьшем покрытии); Easy BF – Easy Backfilling; ESG – Earliest suitable gap.

Полученные результаты иллюстрируют среднюю загрузку кластеров в процентах к максимальному значению (100%).

На рис. 4 показаны моделирования со следующими исходными данными:

– количество кластеров в GRID – 10;

– количество процессорных элементов в каждом кластере– 10;

– размер промежуточного пула – 10;

– количество заданий – 15000.

На рис. 5 показаны результаты моделирования со следующими исходными данными:

– количество кластеров в GRID – 10;

- количество процессорных элементов в каждом кластере – 20;

– размер промежуточного пула – 10;

– количество заданий – 15000.

На рис. 6 показаны результаты моделирования со следующими исходными данными:

– количество кластеров в GRID – 20;

– количество процессорных элементов в каждом кластере – 10;

– размер промежуточного пула – 10;

– количество заданий – 15000.

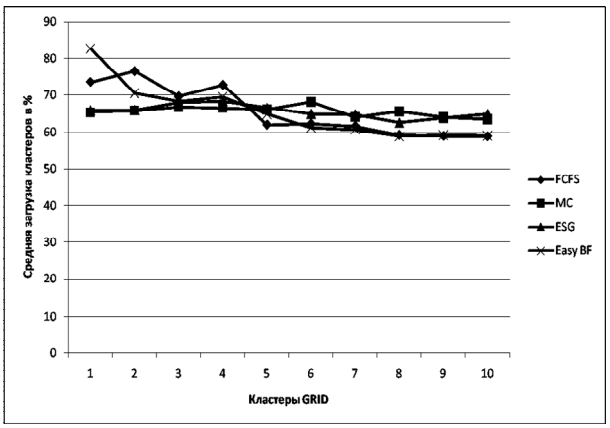

Рис. 4. Результаты моделирования

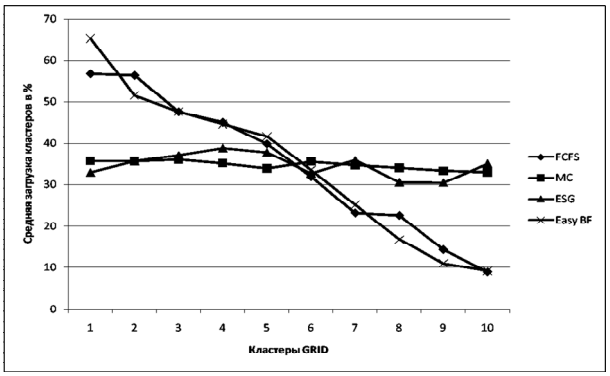

Рис. 5. Результаты моделирования

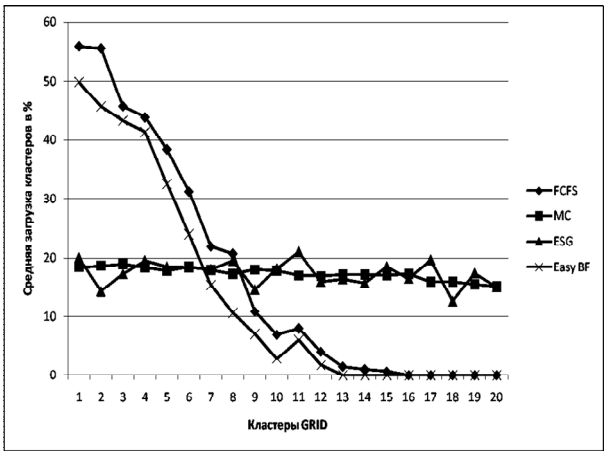

Рис. 6. Результаты моделирования

Приведенные результаты свидетельствуют о том, что с увеличением количества кластеров неравномерность их загрузки возрастает, что в значительной степени может повлиять на выбор режимов их эксплуатации в условиях динамически изменяющихся потоков задач в GRID,, а также в реальных условиях работы GRID, когда количество кластеров может достигать нескольких десятков и сотен.

Выводы

В статье рассмотрена среда моделирования Alea 2, основанная на проекте GridSim, позволяющая оценивать поведение распределенных вычислительных систем при изменяющихся условиях. В ходе исследований были проведены эксперименты по

моделированию загрузки вычислительного кластера GRID с методом формирования очереди на кластер на основе решения задачи о наименьшем вершинном покрытии.

Полученные результаты показали, что использование предложенного алгоритма дает преимущества, заключающиеся в сбалансированной загрузке кластеров GRID по сравнению с известными и часто используемыми в настоящее время алгоритмами FSCF, Easy BackFill и Earliest suitable gap.

Дальнейшие исследования предполагается провести в направлении усовершенствования существующих методов планирования на основе комбинированных алгоритмов, а также изучения влияния динамики поступающих в глобальную очередь за определенные промежутки времени на адаптационные возможности предложенного в данной работе алгоритма и наиболее используемых для планирования политик обслуживания.

Список литературы

1. Методы и модели планирования ресурсов в GRID-системах: монография / В.С. Пономаренко, С.В. Листровой, С.В. Минухин, С.В. Знахур. – Х.: ИД «ИНЖЭК», 2008. – 408 с.

2. Grid Scheduling Architecture Research Group (GSA-RG). [Электронный ресурс]. – Режим доступа к ресурсу: https://forge.gridforum.org/projects/gsa-rg/.

3. Buyya R. Gridsim: a toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing / R. Buyya, M. Murshed // Concurrency and computation: practice and experience. – 2002. – Vol. 14. – P. 1175-1220.

4. Cheduling in HPC resource management systems: Scheduling in HPC resource management systems: Queuing vs. planning / M. Hovestadt, O. Kao, A. Keller, A. Streit // Job Scheduling Strategies for Parallel Processing: 9th International Workshop, JSSPP 2003, Seattle, Wa, Usa, June 24, 2003. – 120 p.

5. Howell F.W. McNab, R.: Simjava: A Discrete Event Simulation Package for Java with Applications in Computer Systems Modelling / F.W. Howell // In Proceedings of the 1998 SCS International Conference on Web-Based Modeling and Simulation, San Diego, CA, 11-14 January 1998.

6. Хант Є. Программист-прагматик. Путь от подмастерья к мастеру / Є. Хант, Д. Томас. – М.: ЛОРИ, 2004. – 289 с.

7. Фаулер М. Рефакторинг: улучшение существующего кода / М. Фаулер. – СПб: Символ-Плюс, 2004. – 430 с.

8. Хорстманн К.С. Java 2. Библиотека профессионала. Том 1. Основы / К.С. Хорстманн, Г. Корнелл.. – М.: Вильямс, 2008. – 816 с.

9. Marco A. A Flexible Resource Co-Allocation Model based on Advance Reservations with Rescheduling Support [Электронный ресурс] / A. Marco, S. Netto, R. Buyya. – Режим доступа к ресурсу: http://www.cloudbus.org/reports/FlexCoallocation2007.pdf.

10. Latest GridSim API [Электронный ресурс]. – Режим доступа к ресурсу: http://www.buyya.com/ gridsim/doc/api/.

11. Klusáček D. Alea 2 - Job Scheduling Simulator / D. Klusáček, H. Rudová //Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques (SIMUTools 2010), ICST, 2010.

12. Li K. Job scheduling and processor allocation for grid computing on Metacomputers / K.Li //Journal of Parallel and Distributed Computing. –2005. – V. 11. – P. 1406-1418.

13. Klus´a˘cek D. Grid scheduling simulation environment / D. Klus´a˘cek, L. Matyska, H. Rudov´a, //Submitted to MISTA, 3rd Multidisciplinary International Scheduling Conference: Theory and Applications, France, 2007.

14. Приемы объектно-ориентированного проектирования. Паттерны проектирования / Є. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес. – СПб.: Питер, 2007. – 366 с.

15. The Grid Workloads Archive [Электронный ресурс]. – Режим доступа к ресурсу: http://gwa.ewi.tudelft.nl/pmwiki/pmwiki.php?n=Workloads.Gwa-t-4.

16. The Grid Workloads Archive / A. Iosup, H. Li, C. Dumitrescu, L. Wolters, D.H.J. Epema //Future Generation Comp. Syst. – 2008. – 24 (7). – P. 672-686.

Поступила в редколлегию 15.04.2011

Рецензент: д-р техн. наук, проф. С.В. Лістровий, Українська державна академія залізничного транспорту, Харків.

МОДЕЛЮВАННЯ ПЛАНУВАННЯ РЕСУРСІВ GRID ЗАСОБАМИ ПАКЕТА GRIDSIM

С.В. Мінухін, А.В. Коровін

Розглянуто принципи моделювання розподілу та планування ресурсів для вирішення завдань GRID в пакеті GridSim. Запропоновано ефективний метод планування ресурсів, що використовує ні-хід на основі завдання про мінімальний вершинному покритті. Розроблено алгоритм і програмне забезпечення на мові Java, що реалізує цей метод. Проведена інтеграція програми у середу пакута GridSim. Проведено експериментальні дослідження для різних значень параметрів моделіруємого GRID-середовища в умовах застосування існуючих і запропонованого методу планування ресурсів GRID. Наведено порівняльний аналіз отриманих результатів, що обгрунтовує ефективність запропонованого в роботі методу.

Ключові слова: *архітектура, кластер, мінімальне вершинне покриття, моделювання, планування, ресурс, Alea 2, GRID-середовище, GridSim.*

MODELING OF GRID RESOURCES PLANNING FUNDS PACKAGE GRIDSIM

S.V. Minukhin, A.V. Korovin

The principles of distribution modeling and planning resources for solving problems in the GRID package GridSim are developed. An effective method for resource planning, using sub-course on the basis of the minimal vertex cover is develped. The algorithm and software in Java, implement this method. Performed integration of the program on Wednesday, pack-ta GridSim. Experimental studies for the various parameters of the model-fix GRID-environment in the context of existing and proposed method of planning of resources GRID. A comparative analysis of the results, justifying the efficiency of the proposed in work method.

Keywords: *architecture, the cluster, the minimum vertex cover, modeling, planning, resource, refactoring, Alea 2, GRID environment, GridSim.*