Krassimir Markov, Vitalii Velychko, Oleksy Voloshin
(editors)

Natural and Artificial Intelligence

ITHEA
SOFIA
2010

Krassimir Markov, Vitalii Velychko, Oleksy Voloshin (ed.)

Natural and Artificial Intelligence

ITHEA®

Sofia, Bulgaria, 2010

ISBN 978-954-16-0043-9

First edition

Recommended for publication by The Scientific Concil of the Institute of Information Theories and Applications FOI ITHEA

This book is engraved in prof. Zinoviy Lvovich Rabinovich memory. He was a great Ukrainian scientist, co-founder of ITHEA International Scientific Society (ITHEA ISS). To do homage to the remarkable world-known scientific leader and teacher this book is published in Russian language and is concerned to some of the main areas of interest of Prof. Rabinovich.

The book is opened by the last paper of Prof. Rabinovich specially written for ITHEA ISS. Further the book maintains articles on actual problems of natural and artificial intelligence, information interaction and corresponded intelligent technologies, expert systems, robotics, classification, business intelligence; etc. In more details, the papers are concerned in:conceptual problems of the natural and artificial intelligent systems: structures and functions of the human memory, ontological models of knowledge representation, knowledge extraction from the natural language texts;network technologies;evolution and perspectives of development of the mechatronics and robotics;visual communication by gestures and movements, psihology of vision and information technologies of computer vision, image processing;object classification using qualitative characteristics;methods for comparing of alternatives and their ranging in the procedures of expert knowledge processing;ecology of programming – a new trend in the software engineering;decision support systems for economics and banking;systems for automated support of disaster risk management;and etc.

It is represented that book articles will be interesting for experts in the field of information technologies as well as for practical users.

General Sponsor: Consortium FOI Bulgaria (www.foibg.com).

Printed in Bulgaria

ISBN 978-954-16-0043-9

C\o Jusautor, Sofia, 2010

МУЛЬТИАГЕНТНЫЙ ПОДХОД К РЕШЕНИЮ ПРОБЛЕМЫ РАВНОМЕРНОГО РАСПРЕДЕЛЕНИЯ ВЫЧИСЛИТЕЛЬНОЙ НАГРУЗКИ

Александр Миков, Елена Замятина, Арсений Козлов

***Аннотация**: В работе рассматриваются различные подходы к решению проблемы балансировки во время распределенного имитационного эксперимента. Авторы предлагают решать эту проблему с помощью управляемой балансировки, которая позволяет при распределении нагрузки по вычислительным узлам использовать не только знания о состоянии вычислительной системы, но и знания о поведении конкретной имитационной модели. При этом предполагается использовать мультиагентный подход. Авторы приводят архитектуру мультиагентной системы балансировки, рассматривают вопросы реализации и результаты проведенных экспериментов.*

***Keywords**: Распределенные вычисления; мультиагентная система; динамическая балансировка; имитационная модель.*

***ACM Classification Keywords**: I.6 Simulation and Modelling – I.6.2 Simulation Languages; I.2 Artificial Intelligence – I.2.5 Programming Languages and Software – Expert system tools and techniques.*

Введение

Метод имитационного моделирования является известным, признанным, а иногда, и единственным методом исследования сложных систем. Зачастую имитационный эксперимент требует больших вычислительных ресурсов. В этом случае целесообразно проводить распределенный имитационный эксперимент, используя ресурсы нескольких узлов вычислительной системы (ВС) (компьютерной сети, многопроцессорной ЭВМ и т.д.). Использование вычислительных ресурсов нескольких узлов вычислительной системы позволяет оптимизировать имитационный эксперимент по времени и по надежности (при выходе из строя одного или нескольких вычислительных узлов их функции берут на себя другие узлы). Распределенная имитационная модель (ИМ) представляет собой совокупность логических процессов LP_i, где $i = 1..n$, которые выполняются параллельно на различных узлах вычислительной системы (ВС) и обмениваются друг с другом сообщениями. Для синхронизации логических процессов в системах распределенного имитационного моделирования применяют специальные алгоритмы: консервативный, оптимистический [Fujimoto, 2003]. Оба этих алгоритма следят за тем, чтобы не возник парадокс времени – ситуация, когда процесс посылает сообщение со штампом времени меньшим, чем локальное время процесса, которому отсылают это сообщение. Однако выигрыши от параллельного выполнения распределенной имитационной модели могут быть сведены к нулю в случае, если возникнет дисбаланс загрузки вычислительных узлов.

Причина дисбаланса заключается в следующем:

- гетерогенность вычислительной системы (узлы ВС имеют разную производительность, а линии связи – разную пропускную способность);
- гетерогенность имитационной модели (некоторые процессы имеют гораздо большую интенсивность обменов, нежели другие, часть процессов оказывает большую нагрузку на вычислительные узлы, выполняя одно событие за другим, в то время как другие могут быть приостановлены, ожидая прихода того или иного сообщения);
- нагрузка, которую оказывают на вычислительные узлы сторонние приложения.

Таким образом, целесообразно корректировать возникающий дисбаланс. С этой целью обычно разрабатывают специальное программное обеспечение, которое следит за нагрузкой вычислительных узлов и восстанавливает равномерное распределение приложений по вычислительным узлам, выполняя перенос части компонентов приложений на другие, менее загруженные узлы. При этом программное обеспечение, выполняющее балансировку, следит за передачей сообщений по линиям связи. Нагрузка линий связи также должна быть сбалансирована.

Восстановление баланса нагрузки является хорошо известной задачей, которая получила множество решений. Разработано большое количество алгоритмов. Однако очень часто эти алгоритмы применимы только для конкретного приложения, для решения конкретной задачи. Существуют и решения, которые применимы для оптимизации распределенного имитационного эксперимента. Однако, несмотря на то, что разработчики подсистемы балансировки SPEEDES [Wilson, 1998] и Charm++ [Zheng, 2005] пытались разработать алгоритмы балансировки, которые могли бы адаптироваться к изменяющейся обстановке, к характерным особенностям той или иной имитационной модели, результаты экспериментов показали, что эффективности от применения этих алгоритмов им удалось добиться лишь в частных случаях. Действительно, найти универсальный алгоритм, который мог бы быть эффективным (обеспечить сокращение времени выполнения имитационного эксперимента) для любой имитационной модели практически невозможно.

Предлагается хотя бы частично разрешить эту проблему, применив *управляемую балансировку* [Миков, 2007]. Управляемая балансировка предполагает, что программное обеспечение, применяемое для восстановления равномерной загрузки должно включать экспертный компонент, который на основании логического вывода предлагает решения для переноса избыточной нагрузки с перегруженного вычислительного узла на менее загруженный. Программные средства балансировки предназначены для системы распределенного имитационного моделирования Triad.Net. (Triad – система автоматизированного проектирования и имитационнного моделирования вычислительных систем [Mikov, 1995], Triad.Net – распределенная ее версия [Миков, 2008], [Миков, 2009], для описания имитационной модели используют язык Triad).

Кроме того, для разработки оригинального алгоритма балансировки, учитывающего все особенности конкретного параллельного (распределенного) приложения, авторы предлагают специальные языковые средства. Эти языковые конструкции написаны на входном языке системы моделирования Triad.Net (язык Triad). Для того чтобы можно было сократить временные затраты на саму балансировку, авторы предлагают использовать мультиагентный подход, который позволяет применить децентрализованный алгоритм балансировки и тем самым сократить время на обмены между компонентами самой подсистемы балансировки. Архитектура мультиагентной подсистемы балансировки, протоколы и алгориты взаимодействия агентов будут описаны ниже.

Задача балансировки

Задача балансировки – это задача отображения неизоморфных связных графов B: $TM \rightarrow NG$, где TM – множество графов моделей, а NG – множество графов конфигураций компьютерной сети. Граф G из NG , $G = (C, Ed)$, определяется множеством вычислительных узлов C и множеством ребер Ed, представляющих линии связи. Граф M из TM, задает имитационную модель. Имитационную модель (ИМ) M можно представить и как совокупность логических процессов $MP = \{LP_j\}$, $j = 1..n$, взаимодействующих между собой путем передачи сообщений.

Описание имитационной модели

Описание модели в системе Triad можно определить как *M* = (*STR*, *ROUT*, *MES*), где *STR* – слой структур, *ROUT* – слой рутин, *MES* – слой сообщений.

Слой структур представляет собой совокупность объектов, взаимодействующих друг с другом посредством посылки сообщений. Каждый объект имеет полюсы (входные и выходные), которые служат соответственно для приёма и передачи сообщений. Основа представления слоя структур – графы. В качестве вершин графа следует рассматривать отдельные объекты. Дуги графа определяют связи между объектами.

Объекты действуют по определённому алгоритму поведения, который описывают с помощью *рутины*. Рутина представляет собой последовательность *событий*e_i, планирующих друг друга (*E* – множество событий; множество событий рутины является частично упорядоченным в модельном времени). Выполнение события сопровождается изменением состояния объекта. Состояние объекта определяется значениями переменных рутины. Таким образом, система имитации является событийно-ориентированной. Рутина так же, как и объект, имеет входные и выходные полюса. Входные полюса служат соответственно для приёма сообщений, выходные полюса – для их передачи. В множестве событий рутины выделено входное событие e_{in}. Все входные полюса рутины обрабатываются входным событием. Обработка выходных полюсов осуществляется остальными событиями рутины. Для передачи сообщения служит специальный оператор *out* (*out*<сообщение>*through*<имя полюса>). Совокупность рутин определяет *слой рутин ROUT*.

Слой сообщений (MES) предназначен для описания сообщений сложной структуры.

Система Triad реализована таким образом, что пользователю необязательно описывать все слои. Так, если возникает необходимость в исследовании структурных особенностей модели, то можно описать в модели только слой структур.

Алгоритмом имитации называют совокупность объектов, функционирующих по определённым сценариям, и синхронизирующий их алгоритм.

Алгоритм исследования

Для сбора, обработки и анализа имитационных моделей в системе Triad.Net существуют специальные объекты – *информационные процедуры* и *условия моделирования*. Информационные процедуры и условия моделирования реализуют алгоритм исследования модели. Информационные процедуры ведут наблюдение за элементами модели (событиями, переменными, входными и выходными полюсами), указанными пользователем. Если в какой-либо момент времени имитационного эксперимента пользователь решит, что следует установить наблюдение за другими элементами или выполнять иную обработку собираемой информации, он может сделать соответствующие указания, подключив к модели другой набор информационных процедур. Условия моделирования анализируют результат работы информационных процедур и определяют, выполнены ли условия завершения моделирования. Информационные процедуры и условия моделирования используют и для сбора информации о поведении модели, о ее характеристиках, и в системе балансировки для сбора информации о модели.

Для имитационной модели в Triad возможно использование операций над моделью (добавление и удаление вершины, добавление и удаление дуг и ребер и др.). Операции изменяют структуру имитационной модели во время имитационного прогона, а это в свою очередь требует перераспределения нагрузки на вычислительных узлах.

Кроме того, авторы разрабатывают систему имитации с удаленным доступом. Это предполагает балансировку запросов при обращении удаленных пользователей к имитационной модели. В настоящее время реализуется распределенная версия Triad.Net. Синхронизация объектов модели, располагающихся на разных вычислительных узлах, выполняется оптимистическим алгоритмом.

Разработка подсистемы управляемой балансировки (централизованный алгоритм)

Как уже говорилось ранее, управляемая балансировка Triad.Net предполагает наличие экспертного компонента. База знаний экспертного компонента включает знания исследователя о конкретной имитационной модели. К примеру, исследователь знает, что через 100 единиц модельного времени с начала моделирования два объекта имитационной модели, представленные логическими процессами LP_i и LP_j, должны интенсивно обмениваться информацией в течение некоторого временного интервала.

В этом случае целесообразно к моменту времени

t = "начало моделирования" + 100 ед. модельного времени

перенести эти два процесса на один вычислительный узел или расположить их на соседних вычислительных узлах, если линия связи между ними имеет хорошую пропускную способность. При переносе нагрузки экспертный компонент использует правила, учитывающие знания о топологии ВС и ИМ, знания о нагрузке узлов, линий связи и о функционировании имитационной модели.

Таким образом, система балансировки в Triad.Net включает следующие компоненты: подсистему анализа и принятия решения, подсистему миграции, подсистему мониторинга имитационной модели, подсистему мониторинга вычислительной системы, базу знаний с правилами перераспределения нагрузки, редактор правил, механизм вывода и подсистему оценки качества оптимизации. *Подсистема мониторинга ВС* собирает информацию о текущем состоянии вычислительной системы, на которой выполняется имитационный эксперимент. *Подсистема мониторинга имитационной модели* собирает информацию о текущем состоянии ИМ (использует механизм информационных процедур). *Подсистема анализа* получает информацию от подсистем мониторинга и принимает решение о перераспределении нагрузки, после чего обращается к *экспертной системе* и получает от нее рекомендации о том, какие объекты имитационной модели следует перенести и на какие вычислительные узлы. Далее управление передают *подсистеме миграции*, которая и осуществляет перенос объектов. *Подсистема оценки качества оптимизации* определяет, насколько изменилось время имитационного прогона и условия этого прогона. Управление является централизованным, все правила хранятся в единой базе знаний, подсистема мониторинга вычислительной системы также располагается на одном из выделенных вычислительных узлов, взаимодействуя с остальными узлами. Централизованный алгоритм характеризуется достаточно ощутимыми временными затратами на коммуникацию.

Мультиагентная подсистема балансировки (децентрализованный алгоритм)

Динамическая система балансировки TriadBalance является *мультиагентной*, она состоит из *совокупности агентов разных типов*: агента-датчика вычислительного узла; агента-датчика имитационной модели; агента анализа; агента миграции; агента распределения. Агенты каждого типа действуют по своему сценарию для достижения цели, а вместе они реализуют балансировку распределенной имитационной модели.

Агент-датчик вычислительного узла постоянно собирает информацию о нагрузке вычислительного узла, о состоянии линий связи, о наличии свободной памяти. Этот агент взаимодействует с агентом анализа, используя доску объявлений.

*Агент-датчик имитационной модели*ведет наблюдение за объектами имитационной модели во время имитационного прогона, регистрируя интенсивность обмена между объектами, частоту выполнения тех или иных событий, частоту изменения, переменных и т.д. Агент-датчик имитационной модели использует механизм информационных процедур. Он взаимодействует с агентом распределения (опять же используя доску объявлений).

Агент анализа, взаимодействуя с агентом-датчиком вычислительного узла (эти агенты являются реактивными), принимает решение о необходимости перераспределения нагрузки. Он является когнитивным агентом и, принимая решения, использует правила из базы знаний агента. В этих правилах анализируются данные о работе аппаратуры: общее время мониторинга; загруженность процессора (в процентах); общее количество дескрипторов в системе; общее количество потоков в системе и т.д.

Агент распределения выполняет следующую роль: он должен решить, какой из объектов, если их несколько на одном узле, следует перенести на другой узел. Для решения этой проблемы агент должен располагать информацией, полученной от агента-датчика имитационной модели. Эту информацию агент распределения извлекает с доски объявлений. Агент распределения определяет, на какой вычислительный узел стоит перенести объект имитационной модели. С этой целью он обращается к агентам распределения вычислительных узлов-соседей.

Таким образом, алгоритм балансировки является *децентрализованным* и представляет собой последовательность шагов:

Агенты-датчики ВС (они располагаются на каждом вычислительном узле) регулярно на всем протяжении имитационного эксперимента собирают статистику о загруженности узлов и размещают ее в базе данных (доска объявлений). Каждый вычислительный узел располагает своей доской объявлений.

*Агент анализа***,** сканируя доски объявлений, с помощью правил из базы знаний определяет, необходимо ли выполнять балансировку. Например, при превышении показателя загрузки вычислительного узла (*Node.Pwr >PwrLimit*) агент анализа принимает решение о необходимости выполнения балансировки. В этом случае агент анализа обращается к агенту распределения.

Агент распределения извлекает информацию из базы данных. Информация в базе данных появляется в результате работы агента-датчика имитационной модели. Для извлечения информации о модели используют механизм информационных процедур. В частности, агенты-датчики собирают информацию о частоте выполения событий имитационных объектов, о частоте обменов между ними (частота появления сообщений на входных и выходных полюсах рутин). Таким образом, агент распределения на основании анализа данных на доске объявлений может сделать вывод о том имитационном объекте, который следует перенести на другой узел. Кроме того, он общается с агентами распределения, расположенными на соседних вычислительных узлах. Он сообщает о перегрузке, о том, что ему необходимо освободиться от некоторых своих имитационных объектов. Агенты распределения других вычислительных узлов извещают о своей нагрузке, пополняя базу данных агента распределения, корректируя правила в базе знаний этого агента. Во время сеанса взаимодействия агенты-распределения других узлов могут сообщить о том, что они незагружены. Далее агент распределения действует по правилам, которые хранятся в базе знаний. Для принятия решения об адресе целевого узла агент распределения, наряду с другими правилами, использует, в частности, правила топологических характеристик ИМ и ВС. При выборе целевого узла сообщение направляется агенту-миграции.

Агент миграции, получив запрос от агента распределения, выполняет непосредственно перенос выбранных объектов имитационной модели на выбранные целевые узлы сети.

Таким образом, выполняется *децентрализованный алгоритм балансировки*. Действительно, нет единого управления процессом балансировки, группа агентов управляет балансировкой на собственном узле, учитывая информацию только соседних вычислительных узлов.

Для повышения гибкости мультиагентной системы балансировки, адаптируемости когнитивных агентов к изменющимся условиям используют два типа метаправил: это *метаправила, определяющие структуру правил* (их используют агенты распределения и анализа для распознавания нужных правил) и *метаправила, определяющие, как можно изменить правила*.

Второй вид метаправил определяет данные и правила, а также критерии, на основании которых будут изменяться правила балансировки когнитивных агентов. Так для агента анализа определены правила, примеры которых приведены ниже:

- *IF %Выражение1% ЗНАК %Выражение2% THEN ОТДАТЬ* (Если значения *Выражения1* и *Выражения2* удовлетворяют условию, то необходимо перенести часть объектов модели имитации с данного узла на другие).
- *IF %Выражение1% ЗНАК %Выражение2% THEN ПРИНЯТЬ* (Если значения *Выражения1* и *Выражения2* удовлетворяют условию, то необходимо перенести часть объектов модели имитации с других узлов на данный).
- *IF %Выражение% BETWEEN %Выражение1% AND %Выражение2% THEN ПРИНЯТЬ* (Если значение *Выражения* находится между значениями выражений *Выражение1* и *Выражение2*, то необходимо перенести часть объектов модели имитации с данного узла на другие).

Агент анализа для вычисления выражений использует информацию, размещенную на доске объявлений, агрегированную по узлу. Напротив, агент распределения использует детализированную информацию для выбора объектов модели имитации. В системе определены следующие типы структур правил для агента распределения: *IF %Выражение1% ЗНАК %Выражение2% THEN ОТДАТЬ; IF %Выражение% BETWEEN %Выражение1% AND %Выражение2% THEN ОТДАТЬ.*

В процессе имитационного эксперимента может сложиться ситуация, при которой некоторые из определенных пользователем правил не будут выполняться и, следовательно, уменьшится эффективность динамической балансировки, основанной на правилах. В таком случае система балансировки должна сама следить за состоянием правил и корректировать их, удалять или добавлять новые, в зависимости от сложившейся ситуации в процессе имитационного эксперимента. Для того чтобы агенты могли правильно понимать, в каких случаях необходимо корректировать, добавлять или удалять правила, в системе введены метаправила, определяющие правила изменения правил, например:«*Если частота выполнения правила больше* (>, <, <=, >=, ==) *заданного значения, то необходимо изменить правило*»; «*Если правило не выполняется никогда, то необходимо изменить правило*»; «*Если правило не может быть выполнено, то необходимо изменить правило*»и т.п. Таким образом, система балансировки для каждого правила ведет статистику по эффективности работы правил, на основе которой определяется необходимость их изменений.

Реализация подсистемы мультиагентной балансировки и результаты экспериментов

Настоящая версия мультиагентной подсистемы динамической балансировки PSU.HPC.TriadBalance распределённой имитационной модели реализована на кластере из 8 вычислительных узлов (рис. 1), работающих под управлением операционной системы (ОС) Windows HPC Server 2008 (далее WinHPC), которая является специализированной кластерной версией (High Performance Computing) операционной системы Windows Server 2008. Подсистема разработана с использованием .Net Framework 3.5 и пакета

сборок HPC Pack 2008, позволяющего управлять распределённым выполнением приложений в используемой авторами ОС. Вычислительным узлом кластера может считаться как отдельный компьютер, так и серверная стойка. Выделяются узлы трех типов: головной узел (Head Node), вычислительный узел (Compute Node) и узел брокера WCF (WCF Broker Node). Головной узел может также выполнять и другие функции, т.е. быть вычислительным узлом или узлом брокера WCF. Для управления кластерными вычислениями использовался «HPC Job Manager».

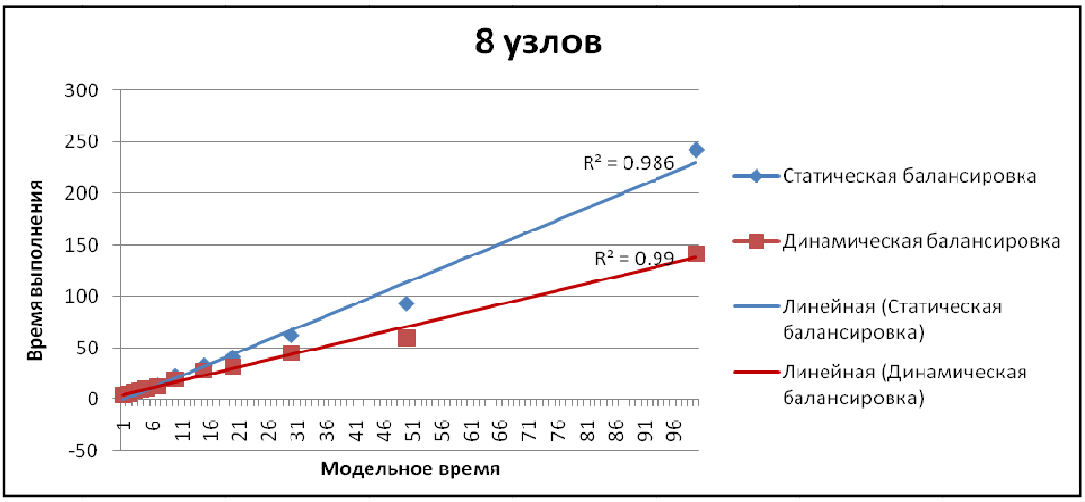

Рис. 1. Результаты прогона модели «Клиент-сервер» на 8 вычислительных узлах кластера

Перед запуском системы выполнялась *предварительная статическая балансировка модели* (начальное размещение объектов модели на вычислительных узлах сети). Следующий шаг – *настройка системы*. Пользователь, основываясь на знаниях о модели (он знает, как должна работать модель), модифицирует правила балансировки. Система балансировки включает *редакторы правил и метаправил*. Правила формируются в виде XML-файла. На основании этих правил агенты принимают решения о переносе объектов модели с одного вычислительного узла сети на другой.

В систему балансировки включена *подсистема визуализации*, с помощью которой можно определить, на каких узлах располагаются объекты имитационной модели. Текущее расположение объектов имитационной модели на вычислительных узлах можно получить в любой момент времени имитационного эксперимента по запросу пользователя. Кроме того, подсистема визуализации позволяет оценить качество оптимизации, отображая в удобных формах загрузку процессоров (и других показателей о функционировании модели и вычислительных узлов во время имитационного прогона).

Для проведения экспериментов была разработана имитационная модель "Клиент-Сервер". В результате экспериментов удалось получить данные, которые представлены на рис.1. Результаты показали, что время, затраченное на имитационный эксперимент при использовании подсистемы мультиагентной балансировки, действительно снижается. Применение подсистемы становится более эффективным при больших значениях системного времени моделирования.

Заключение

В работе рассмотрены вопросы реализации динамической балансировки, предложен централизованный и децентрализованный алгоритмы. Децентрализованный алгоритм основан на мультиагентном подходе. В работе рассматривается также архитектура подсистем управляемой балансировки, реализующих оба алгоритма. Для подсистемы балансировки, основанной на мультиагентном подходе, приведены перечень агентов, их назначение, алгоритмы их взаимодействия и особенности реализации. Кроме того, приводятся

результаты прогона модели «Клиент-Сервер», которые показывают выигрыш во времени при проведении распределенного имитационного эксперимента с применением мультиагентного подхода.

Благодарности

Статья частично финансирована из проекта **ITHEA XXI** Института Информационных теорий и Приложений FOI ITHEA и консорциума FOIBulgaria (www.ithea.org, www.foibg.com)

Библиографический список

[Fujimoto, 2003] Fujimoto R.M. Distributed Simulation Systems // Proceedings of the 2003 Winter Simu-lation Conference S. Chick, P. J. Sanchez, D. Ferrin, and D. J. Morrice, eds. Pp. 124-34.

[Wilson,1998] Wilson L.F. and Shen W. Experiments In Load Migration And Dynamic Load Balancing In Speedes // Proceedings of the 1998 Winter Simulation Conference. D.J. Medeiros, E.F. Watson, J.S. Carson and M.S. Manivannan, eds. Washington. 1998. Pp.483-490.

[Zheng, 2005] Zheng G. Achieving High Performance on Extremely Large Parallel Machines: Performance Prediction and Load Balancing // Ph.D. Thesis, Department of Computer Science. University of Illinois at Urbana-Champaign, 2005,165 p. [http://charm.cs.uiuc.edu/].

[Миков, 2007] Миков А.И., Замятина Е.Б., Осмехин К.А. Динамическоераспределениеобъектовимитационноймодели, основанноеназнаниях // Proceedings of XIII International Conference “Knowedge-Dialogue-Solution” KDS), ITHEA, Sofia, 2007. Vol.2., Pp. 618-624.

[Mikov, 1995] Mikov A.I. Simulation and Design of Hardware and Software with Triad //Proc.2nd Intl.Conf. on Electronic Hardware Description Languages. Las Vegas, USA, 1995. Pp. 15-20.

[Миков, 2008] Миков А.И., Замятина Е.Б. Технология имитационного моделирования больших систем // Труды Всероссийской научной конференции «Научный сервис в сети Интернет» – М.: Изд-во МГУ, 2008. С.199-204.

[Миков, 2009] Миков А.И., Замятина Е.Б., Козлов А.А. Оптимизация параллельных вычислений с применением мультиагентной балансировки // Труды международной научной конференции «Параллельные Вычислительные Технологии». Нижний Новгород – Челябинск, Изд. ЮУрГУ, 2009. С. 599-604.

Сведения об авторах

Александр Миков *– Кубанский государственный университет, профессор, заведующий кафедрой «Вычислительные технологии»; Россия, г. Краснодар, ул. Аксайская, 40/1-28; e-mail: alexander_mikov@mail.ru.*

Елена Замятина *– Пермский государственный университет, доцент кафедры математического обеспечения вычислительных систем; Россия, г. Пермь, 614017, ул. Тургенева, 33–40; e-mail: e_zamyatina@mail.ru.*

Арсений Козлов *– Пермский государственный университет, студент кафедры математического обеспечения вычислительных систем; Россия, г. Пермь, ул. Левченко, 6-87; e-mail: Arko87@yandex.ru.*