МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное
образовательное учреждение высшего образования
«ПЕРМСКИЙ ГОСУДАРСТВЕННЫЙ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ»

ООО «Лаборатория АРГУМЕНТ»

*Посвящается 100-летию*
*Пермского университета*

# ВЫСОКОПРОИЗВОДИТЕЛЬНЫЕ ВЫЧИСЛЕНИЯ НА ГРАФИЧЕСКИХ ПРОЦЕССОРАХ

Материалы III Всероссийской
научно-практической конференции
с международным участием с элементами
научной школы для молодежи
(ВВГП–2016)

г. Пермь, ПГНИУ, 16–20 мая 2016 г.

ИЗДАТЕЛЬСКИЙ ЦЕНТР
"Perm University Press"

Пермь 2016

УДК 004.4:004.92
ББК 32.973
В93

В93 **Высокопроизводительные** вычисления на графических процессорах: материалы III Всерос. науч.-практ. конф. с междунар. участием с элементами науч. шк. для молодежи (ВВГП–2016); г. Пермь, ПГНИУ, 16–20 мая 2016 г. / отв. за вып. А. Г. Деменев, С. В. Русаков; Перм. гос. нац. исслед. ун-т. – Пермь, 2016. – 104 с.

**High-Performance** Computing on GPUs: Proceedings of the 3rd All-Russian Research and Practice Conference with International Participation and Young Scientists School (GPU-HPC-2016), Perm, May 16–20, 2016 / Executive editors: A. G. Demenev, S. V. Rusakov; Perm State University. – Perm, 2016. – 104 p.

ISBN 978-5-7944-2831-5

В сборнике представлены статьи и тезисы докладов на русском и английском языках участников III Всероссийской научно-практической конференции с международным участием с элементами научной школы для молодежи «Высокопроизводительные вычисления на графических процессорах», посвященной 100-летию Пермского университета, которая проводилась 16–20 мая 2016 г. в г. Перми в рамках Форума «Математика и глобальные вызовы XXI века». Публикуемые материалы отражают различные аспекты реализации высокопроизводительных вычислений на графических процессорах, включая общие вопросы организации вычислений, особенностей реализации численных методов, решение конкретных прикладных задач.

Сборник предназначен для преподавателей, научных работников, аспирантов, магистрантов и студентов высших учебных заведений.

**УДК 004.4:004.92**
**ББК 32.973**

*Печатается по решению оргкомитета конференции ВВГП–2016*

Ответственные за выпуск редакторы:
канд. физ.-мат. наук, доц. ***Алексей Геннадьевич Деменев,***
д-р физ.-мат. наук, проф. ***Сергей Владимирович Русаков***

Организаторы конференции ВВГП–2016:
**Федеральное государственное бюджетное образовательное учреждение высшего образования «Пермский государственный национальный исследовательский университет», ООО «Лаборатория АРГУМЕНТ»**

Спонсоры и партнёры конференции ВВГП–2016:
**Intel в России и СНГ, NVIDIA Ltd** и **ООО «Технический центр “Гармония”»**

ISBN 978-5-7944-2831-5 

УДК 004.85

# ОПТИМИЗАЦИЯ ЭТАПОВ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ С ИСПОЛЬЗОВАНИЕМ ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ ВЫЧИСЛЕНИЙ: РЕЗУЛЬТАТЫ ИССЛЕДОВАНИЙ [9]

*Замятина Елена Борисовна*

Национальный исследовательский университет «Высшая школа экономки», 614070, Россия, г. Пермь, Студенческая, 38, ezamyatina@hse.ru

Пермский государственный национальный исследовательский университет, 614990, Россия, г. Пермь, ул. Букирева, 15, e_zamyatina@mail.ru

*Миков Александр Иванович*

Кубанский государственный университет, 614990, Россия, г. Краснодар, Ставропольская, 49, alexander_mikov@mail.ru

В работе рассматривается проблема оптимизации этапов имитационного моделирования по времени. Необходимость в сокращении времени выполнения имитационного эксперимента объясняется тем, что очень часто метод имитационного моделирования применяется для исследования сложных систем, которым можно отнести и компьютерные сети. Рассматриваются проблемы оптимизации алгоритма синхронизации, балансировки нагрузки и анализа результатов моделирования.

Ключевые слова: имитационное моделирование, распределенная имитационная модель, балансировка нагрузки, оптимистический алгоритм синхронизации, алгоритм синхронизации, основанный на знаниях.

Известно, что метод имитационного моделирования широко применяется для исследования сложных динамических систем и, очень часто, является единственным методом исследования в силу того, что нет возможности исследовать эти системы, используя натурные эксперименты, или по той причине, что нельзя формализовать сложные системы из-за большого количества взаимосвязей ее элементов. К сложным динамическим системам можно отнести, например, компьютерные сети, параллельные и распределенные вычислительные системы (РВС). В настоящее время получили широкое распространение сетевые организации, виртуальные предприятия, социальные сети и т.д., для их исследования также целесообразно применять методы исследования, применяемые к исследованиям распределенных вычислительных систем. Существует большое количество систем имитационного моделирования (СИМ), реализующих методы имитационного



Публикация подготовлена при финансовой поддержке РФФИ: грант № 16-47-230336 и Администрации Краснодарского края.

моделирования. По той причине, что эти программные системы имитационного моделирования используются при решении сложных задач, возникает необходимость в том, чтобы вычислительный эксперимент выполнялся за приемлемое время, необходимо предъявлять к СИМ особые требования по производительности. В работе рассматриваются пути повышения производительности СИМ TriadNS.

## 1. Оптимизация выполнения имитационного эксперимента

Система имитационного моделирования (СИМ) TriadNS[1,3], предназначена для проектирования и исследования вычислительных систем. Объектом исследования TriadNS являются как структура параллельных и распределенных вычислительных систем (в частности, компьютерных сетей), так и алгоритмы, которые выполняются на их базе. Система имитации позволяет определить адекватность поведения параллельных и распределенных алгоритмов при значительном изменении количества вычислительных узлов (проиграть их функционирование в вычислительной среде, которой пока не существует). Очень часто количество вычислительных узлов в РВС таково, что имитационный эксперимент является затратным по времени. По этой причине возникает необходимость в оптимизации прогона имитационного эксперимента. С этой целью является целесообразным использование высокопроизводительной техники и соответствующих алгоритмов и технологий, позволяющих использовать ресурсы нескольких вычислительных систем. Процесс моделирования, наряду с имитационным экспериментом, включает и другие этапы, а именно, этапы валидации, верификации имитационной модели и анализа результатов имитационного моделирования. Рассмотрим более подробно решения, которые были приняты авторами для оптимизации этих этапов. Прежде всего, представим архитектуру СИМ TriadNS и особенности организации имитационной модели и подсистемы сбора и обработки статистики.

## 2. Имитационная модель в TriadNS

В TriadNS[1,2] принято трехуровневое представление имитационной модели (ИМ): 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 (отдельные вычислительные узлы кластеров, концентраторы, маршрутизаторы, серверы, рабочие станции, например, или одноканальное устройство, многоканальное устройство, очередь). Дуги графа определяют связи между объектами.

Имитационная модель имеет иерархическое представление. Отдельные объекты, представляющие вершины графа, могут быть расшифрованы подграфом более низкого уровня и т.д. Объекты действуют по определённому сценарию, который описывают с помощью рутины. Рутина представляет собой последовательность событий, планирующих друг друга (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.

Слой сообщений предназначен для описания сообщений сложной структуры.

### 3. Запуск модели на выполнение

После того, как модель полностью описана, т.е. определена структура модели (граф, состоящий из вершин и дуг (или ребер), их соединяющих), описаны все рутины и выполнено их наложение на соответствующие вершины (***put*** Action ***on*** M.G[i], где Action – экземпляр рутины, M – модель, G[i] – вершина), модель можно запускать на выполнение. Запуск на выполнение выполняется с помощью оператора ***simulate***. Имитационный эксперимент носит итерационный характер. Если результаты эксперимента не соответствуют критериям качества, то имитационный эксперимент запускается на выполнение снова и продолжается это до тех пор, пока полученная модель не будет соответствовать критериям качества. Критерии качества и оператор завершения моделирования описывают в специальной программной единице – в «условиях моделирования» («conditions of simulation»). Оператор ***simulate*** позволяет выполнять *параллельно* запуск на выполнение сразу несколько моделей: *simulate* M1.a, M2.h, M3.s *on* C1... (M1.a, M2.h, M3.s – элементы модели, которые предполагается оценить и проверить, соответствует ли их значение критериям качества модели; C1- имя «условий моделирования», предварительно описанных оператором “***conditions of simulation***”).

### 4. Распределенная имитационная модель и алгоритмы синхронизации

Выигрыш во времени при выполнении имитационного эксперимента может быть получен за счет использования высокопроизводительной аппаратуры, то есть за счет использования вычислительной мощности сразу нескольких вычислительных узлов. Имитационная модель в TriadNS может быть распределена по вычислительным узлам следующим образом: либо каждая вершина слоя структур ИМ может выполняться на отдельном вычислительном узле высокопроизводительной вычислительной системы, либо

группа вершин, которые характеризуются сильной связностью (возможен интенсивный обмен сообщениями между логическими процессами, которые связаны с этими вершинами). Как уже говорилось ранее, на каждую вершину должна быть наложена рутина, при запуске ИМ на выполнение каждая рутина представляет собой логический процесс, состоящий из событий, планирующих друг друга. Логические процессы могут выполняться параллельно, обмениваясь сообщениями по линиям связи. Для выполнения распределенной имитационной модели (РИМ) необходимо синхронизировать логические процессы с помощью специального алгоритма синхронизации.

Существует два класса классических алгоритмов синхронизации: консервативные и оптимистические[4]. *Консервативные алгоритмы* обычно «сдерживают» продвижение системного времени, очередное событие выполняется, если можно убедиться, что оно является безопасным. Событие является безопасным, если можно твердо сказать, что другой логический процесс не передаст текущему сообщение, помеченное меньшим штампом времени, чем рассматриваемое (текущее) событие. Для того чтобы расширить временные горизонты процессов и не попасть в тупиковую ситуацию, каждый логический процесс по всем выходным связам при изменении своего времени посылает сообщение – либо реальное, запрограммированное в модели, либо фиктивное или «нулевое». Классическим *оптимистическим алгоритмом* является алгоритм «Time Warp». Алгоритм позволяет обрабатывать любые события (не только безопасные), но при поступлении сообщения с временной меткой T’ (используют еще термин «штамп времени», которая меньше текущего времени T, происходит откат на момент времени T’). Для реализации отката необходимо восстановить состояние текущего логического процесса на момент времени T’. Кроме того, необходимо отменить сообщения, разосланные другим логическим процессам. Для этого используют механизм «анти-сообщений»: для каждого обработанного сообщения хранится противоположное ему. Для отмены сообщения посылают «анти-копию». При получении анти-сообщения процесс либо удаляет положительное сообщение из очереди, если оно не было еще обработано, либо так же производит откат до времени анти-сообщения.

В СИМ TriadNS был реализован оптимистический алгоритм, основанный на знаниях о модели [5]. Сама природа модели содержит не только зависимости между объектами по времени, которую традиционно используют алгоритмы, но и зависимость (событие, состояние) →событие. Проблема алгоритма состоит в том, что необходимо на каждом шаге определить, является ли обработка следующего локального события безопасной, т.е можно ли гарантировать отсутствие отката. Анализ текущей последовательности событий (в рамках одного прогона) или результаты предыдущих выполнений системы, а также, знания эксперта позволяют выявить зависимости между событиями и, следовательно, определить безопасное

событие. Наиболее адекватным решением является применение продукционной экспертной системы. Нельзя целиком полагаться на знания пользователя, необходимо уметь генерировать продукционные правила автоматически: в ходе статического анализа кода модели перед выполнением, а также в ходе выполнения: при откатах создавать правила, предотвращающие появление откатов в будущем. Если мы обработали событие покупки билета вторым клиентом (E2), а затем произошел откат из-за события покупки билета первым клиентом (E1), в системе генерируется связь ($E1 \rightarrow E2$).

Моделирование проводилось на кластере, с использованием нескольких простых моделей. Для сравнения производительности были выбраны классический консервативный и оптимистический алгоритмы.

По графику видно, что алгоритм показывает лучшую производительность, чем оба традиционных алгоритма, причем при увеличении количества вычислительных узлов, разница становится более заметна. Происходит это потому, что увеличивается количество логических процессов, а, следовательно, и общее количество внешних связей в системе, что для консервативного алгоритма приводит к уменьшению быстродействия, т.к. временной горизонт продвижения становится меньше, а для оптимистического приводит к увеличению количества откатов.

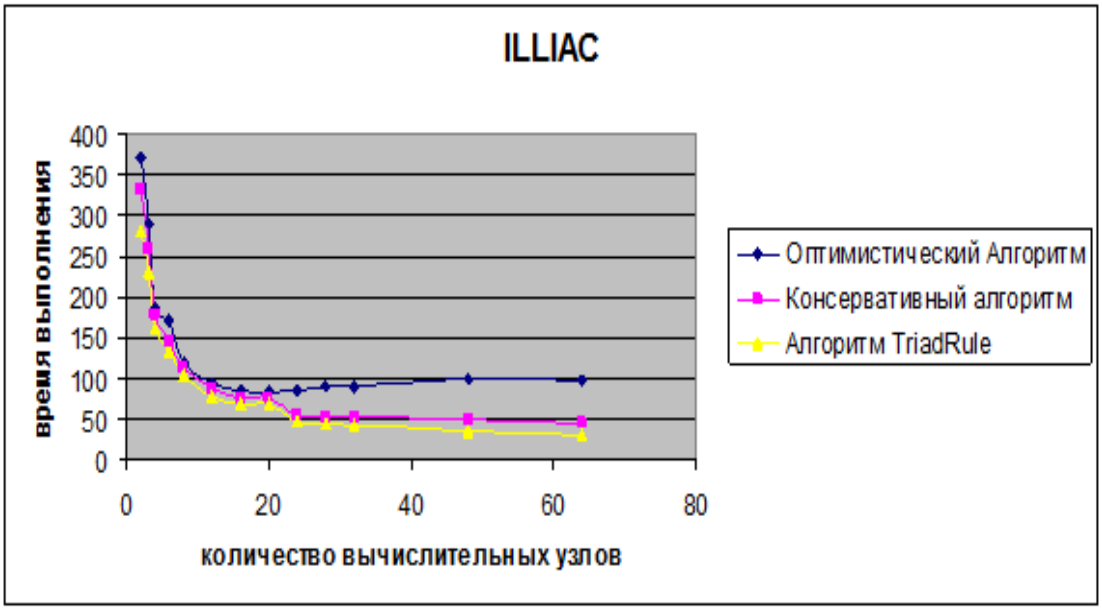


Рис. 2. Время выполнения имитационного эксперимента в зависимости от количества вычислительных узлов

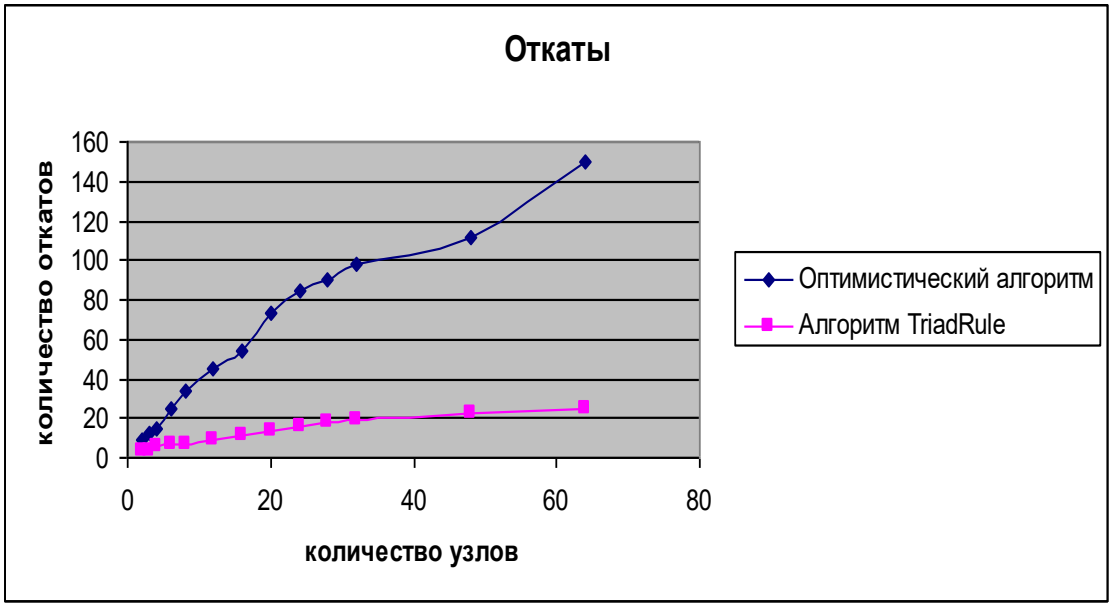


Рис. 3. Сравнение количества откатов

Эксперименты с алгоритмом синхронизации, основанном на знании, продолжились. Была разработана агентноориентированная система имитации на языке SCALA (в предыдущем случае система моделирования является событийно-ориентированной).

В системе был также реализован оптимистический алгоритм синхронизации, основанный на знаниях об имитационной модели. Знания извлекались из онтологий. Были проведены эксперименты с одной и той же моделью многократно при одних и тех же параметрах. Общая длительность моделирования составляла 300 (±15) ед. модельного времени. Эксперименты показали, что метрики, характеризующие скорость проведения распределенного эксперимента, управляемого модифицированным алгоритмом синхронизации (алгоритм KBASA, основанный на знаниях о модели) существенно сократились по сравнениям с метриками классического оптимистического алгоритма Time Warp. Действительно, (а) количество откатов сократилось с 71.2 до 3.8(рис.2.), (б) общее число неглубоких откатов сократилось практически до нуля (в среднем 0.1-0.2 против 7-11), (в) число глубоких откатов также сократилось с 19.4 до 3.2б, (г) количество отправленных антисообщений сократилось с 108.4 до 12.8 (Рис.3)[6,7].

## 5. Балансировка нагрузки на вычислительные узлы распределенного имитационного эксперимента

Еще одним фактором возможного выигрыша во времени при проведении имитационного эксперимента является балансировка нагрузки на вычислительных узлах, используемых для выполнения на них распределенной имитационной модели. Если какой-либо узел перегружен или перегружена линия связи между узлами (т.е., если возник

дисбаланс), то часть логических процессов имитационной модели простаивают, ожидая сообщения с перегруженного узла или линии связи.

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.

Предлагается хотя бы частично разрешить эту проблему, применив управляемую балансировку. 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

Были разработаны как централизованный, так и распределенный алгоритмы, выполняющие управляемую балансировку. Эксперименты показали преимущество распределенного алгоритма [6].

Распределенный алгоритм балансировки представлен взаимодействующими агентами и реализован специальным программным компонентом TriadBalance. Кратко опишем алгоритм. Каждый вычислительный узел располагает пятью агентами: два из них – это агенты датчики (один следит за состоянием аппаратуры, другой – за состоянием имитационной модели (длительность выполнения события, частота выполнения события и т.д.)), к остальным следует отнести агенты анализа, распределения и миграции.

*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*.

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.3. 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.

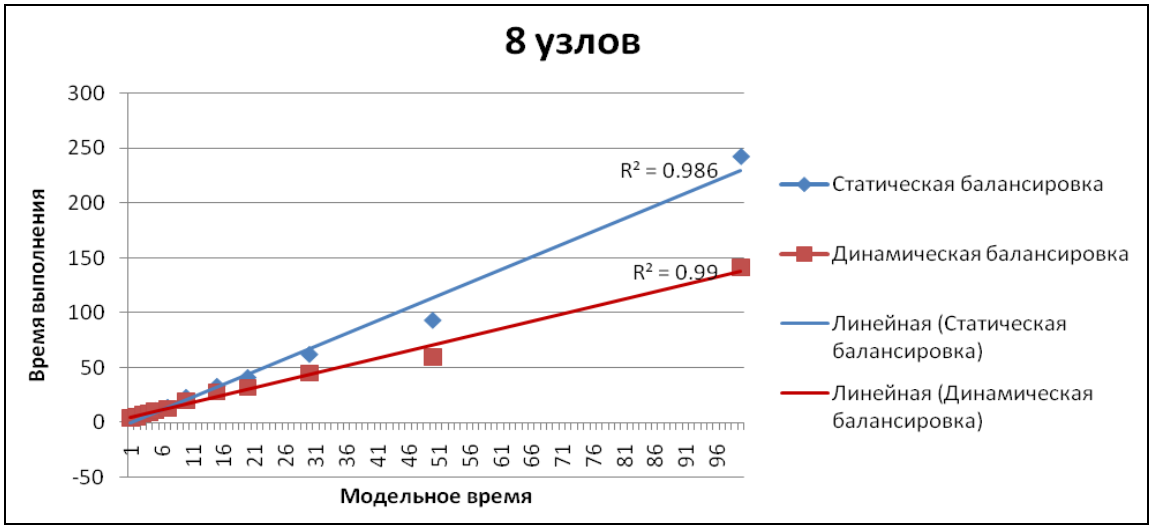


Рис. 3. Зависимость времени, затраченного на имитационный эксперимент, от алгоритма балансировки

## 6. Анализ результатов моделирования

Задача анализа результатов сеанса имитации – это задача разработки алгоритмов, исходными данными для которых являются длинные последовательности [8,9]. В этих условиях даже алгоритмы квадратичной сложности требуют большого времени исполнения, не говоря уже об алгоритмах переборного характера. Существенное ускорение в реализации может быть достигнуто при использовании вычислительных систем с массовым параллелизмом или больших распределенных систем.

Рассмотрим алгоритм поиска частного паттерна внутри длинной последовательности.

Описание алгоритма:

Входной протокол разбивается на последовательности, длины $l$, считываемые Map. Map ищет все совпадения между отрезками паттерна и входными последовательностями (так как паттерн в общем случае имеет длину больше $l$). Для каждого найденного совпадения, если только оно не содержит конец шаблона, Map возвращает 2 пары значений. Первая пара в качестве ключа содержит данные о позиции, в которой найдено совпадение, и в качестве значения – объект со статусом Is. Вторая пара содержит объект со статусом request и в качестве ключа имеет данные о позиции шаблона и номере последовательности, в которой предположительно должно быть продолжение шаблона. Таким образом, получим на входе Reduce список из 1 или 2 объектов. Если объектов 2, это означает, что мы нашли 2 соседние последовательности, содержащие шаблон. Далее Reduce повторяет выполнение до тех пор, пока не найден весь шаблон или не превышен лимит на количество итераций.

Если шаблон помещается в 8 отрезков исходного протокола, вывод reduce будет выглядеть следующим образом:

Для того чтобы полностью обнаружить шаблон потребовалось 3 этапа. На первом этапе формируются последовательности длины $2l$. На втором шаге получаем 5 последовательности длины $4l$, 1 последовательность длины $3l$, и последовательность длины $l$, содержащую конец шаблона. Необходимость всегда возвращать информацию о совпадении, содержащую конец отрезка в неизменном виде необходима для случая, когда паттерн делится на нечетное количество последовательностей. Как видно, Reduce имеет здесь избыточный выход, однако на последнем шаге гарантированно будет только 1 объект, содержащий информацию о всём шаблоне. Кроме того, выполняемый в reduce код имеет константную оценку сложности. Самая сложная часть алгоритма выполняется в функции Map над последовательностями незначительной длины и может быть перенесена на сторону данных. Оценка сложности всего алгоритма

$$O\left(\frac{n}{l*p_1}*(l^2+\ m*l)*C_1+\ \log_2\frac{m}{l}*C_2*\frac{m}{l}*\frac{k}{p_2}=\frac{n*(l+m)}{p_1}+\frac{m*k}{l*p_2}\log_2\frac{m}{l}\right),$$

где $p_1$ - количество параллельно выполняющихся map функций, $p_2$ - количество параллельно выполняющихся reduce функций, $n$ – длина протоколов, $l$ – длина считываемой Map строки, настраиваемый параметр, $m$ – длина шаблона, $C1, C2$ – константы, $k$ – количество повторений шаблона в протоколе. В этой формуле $(l^2+\ m*l)*C_1$ - оценка сложности функции FindPattern(); $\frac{n}{l*p_1}$ – количество выполняемых последовательно функций map, $\log_2\frac{m}{l}$ – количество этапов за которые в reduce попадёт полный шаблон, $\frac{k}{p_2}$ – грубая оценка количества reduce выполняемых последовательно на каждом шаге[9].

## 7. Выводы

В работе представлен алгоритм синхронизации, который управляет работой распределенной имитационной модели, алгоритм балансировки и алгоритм поиска частного паттерна. Все эти алгоритмы сопровождаются результатами экспериментов, показавших, что представленные алгоритмы действительно позволяют сократить и время прогона имитационного эксперимента, и время, необходимое на анализ результатов. В дальнейшем предполагается использовать технологию CUDA для реализации алгоритмов синхронизации на графических процессорах.

## Библиографический список


1. *Замятина Е.Б., Миков А.И.* Программные средства системы имитации Triad.Net для обеспечения ее адаптируемости и открытости. Информатизация и связь. 2012. № 5. С. 130–133.
2. *Mikov A.I.* Formal Method for Design of Dynamic Objects and Its Implementation in CAD Systems // Gero J.S. and F.Sudweeks F.(eds), Advances in Formal Design Methods for CAD, Preprints of the IFIP WG 5.2 Workshop on Formal Design Methods for Computer-Aided Design, Mexico, Mexico, 1995. pp.105-127.
3. *Mikov A., Zamiatina E.* Program Tools and Language for Networks Simulation and Analysis. /Mikov A.//Proceedings SDN & NFV – The Next Generation of Computational Infrastructure: 2014 International Science and Technology Conference «Mmodern

Networking Technologies (MoNeTec)» October 27-29, 2014 Lomonosov Moscow State University pp. 94-102

4. *Fujimoto R.M.* Distributed Simulation Systems. In Proceedings of the 2003 Winter Simulation Conference S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, eds. The 2003 Winter Simulation Conference 7–10 December 2003. The Fairmont New Orleans, New Orleans, LA. P. 124–134.
5. *Mikov A.,Zamyatina E.,Kozlov A. Ermakov S.* Some Problems of the Simulation Model Efficiency and Flexibility. Proceedings of «2013 8th EUROSIM Congress on Modelling and Simulation EUROSIM 2013», Cardiff, Wales, United Kingdom, 10–13 of September. P .532–538.
6. *Замятина Е.Б., Митраков А.А.* Алгоритмы синхронизации, основанные на знаниях о модели, в распределенных системах имитации. XIV Международная конференция «Высокопроизводительные и параллельные вычисления на кластерных системах» (HPC-2014), г. Пермь, 10-12 ноября 2014г., Пермский национальный исследовательский политехнический ун.-т., 2014, (ISBN 978-5-398-01312-2) с.175-182
7. *Митраков А.А.* Применение знаний для синхронизации агентов в параллельном дискретно-событийном моделировании. Прикладная информатика. №1(55) 2015, стр.35-45
8. *Kolevatov G.A., Zamyatina E.B.* Simulation Analysis Framework Based on Triad.Net. Proceedings of the 6-th Spring/Summer Young Reseachers' Colloquium on Software Engineering. SYRCoSE 2012, Perm, May 30-31, 2012-Perm, Russia, pp.160-163.
9. *Миков А.И., Мопсан Н.В.* Параллельная обработка паттернов в больших данных имитационного моделирования. Информатизация и связь. №2, 2013, стр. 10-17

## OPTIMIZATION OF THE SIMULATION STAGES USING HIGH PERFORMING CALCULATIONS: THE RESULTS OF INVESTIGATION [10]

*Zamyatina Elena B.*

National Research University "Higher School of Economics ", st. Studencheskaya, 38, Perm, Russia, 614070, ezamyatina@hse.ru

Perm State University, st. Bukireva, 15, Perm, Russia, 614990, e_zamyatina@mail.ru

*Mikov Alexander I.*

Kuban State Unverity, 614990, Russia, Krasnodar, Stavropolskaya, 49, alexander_mikov@mail.ru

The paper deals with the problem of optimizing the stages during the simulation experiment. That is an optimization by time. The need to reduce the execution time of the simulation experiment is explained by the fact that very often the simulation method is used for the study of complex systems (computer networks, for example). The problems of optimization of synchronization algorithm, load balancing algorithm and analysis of simulation results are discussed.

Key words: simulation, distributed simulation model, load balancing, optimistic synchronization algorithm, knowledge based synchronization algorithm

УДК 532.546.6

## ЧИСЛЕННОЕ МОДЕЛИРОВАНИЕ ДИНАМИКИ ИЗОЛИРОВАННОГО ВКЛЮЧЕНИЯ В ПОРИСТОЙ СРЕДЕ В РАМКАХ МОДЕЛИ БАКЛЕЯ-ЛЕВЕРЕТТА[11]

*Иванцов Андрей Олегович, Любимова Татьяна Петровна*

Пермский государственный национальный исследовательский университет, 614990, Россия, г. Пермь, ул. Букирева, 15, ivants2@psu.ru

Институт механики сплошных сред УрО РАН, 614013, Россия, г. Пермь, ул. Академика Королева, 1, lyubimovat@mail.ru

На основе модели Баклея-Леверетта проведено моделирование динамики фронтов вытеснения и изолированного включения (капля другой жидкости) в пористой среде в гравитационном поле. Показано, что горизонтальный фронт вытеснения устойчив при любом соотношении вязкостей в случае, когда вытесняющая жидкость менее плотная. Если плотность вытесняющей жидкости больше, чем вытесняемой, то горизонтальный фронт вытеснения неустойчив, при этом динамика системы зависит от соотношения вязкостей жидкостей. В случае, когда вытесняющая жидкость более вязкая, формируется известная пальцеобразная неустойчивость фронта вытеснения в пористой среде. Если вязкость вытесняющей жидкости меньше, чем вытесняемой, то неустойчивость связана с быстрым ростом толщина фронта вытеснения. Однако, при этом формируется вторичный тонкий фронт вытеснения с меньшим скачком насыщенности сред. Величина скачка насыщенности зависит от соотношения вязкостей. Динамика вторичного фронта аналогична наблюдаемой в случае, когда вытесняющая жидкость более вязкая.

Ключевые слова: пористая среда, фронт вытеснения, модель Баклея-Леверетта

Проведено численное моделирование динамики фронта вытеснения в пористой среде под действием вибраций в рамках модели Баклея-Леверетта. Для описания многофазной


Публикация подготовлена при финансовой поддержке РНФ (грант № 14-21-00090).