# ВЫЧИСЛИТЕЛЬНАЯ СЕТЬ НА ОСНОВЕ ПЕРСОНАЛЬНЫХ КОМПЬЮТЕРОВ

**Р.Ю. Лопаткин, С.А. Петров, В.А. Иващенко (Сумы, Украина)**

**Введение**

Современные научные задачи требуют все больших вычислительных ресурсов. Для удовлетворения этих потребностей ученых в последнее время бурно развивается грид-технология [1], на что тратятся огромные ресурсы. Вместе с тем каждый научный сотрудник имеет у себя на рабочем месте персональный компьютер (ПК), зачастую с неплохими характеристиками. Возникает вопрос: насколько рационально используются ресурсы ПК? Ведь неполную загрузку можно расценивать как потерю ресурсо-часов. Мы считаем, что человечество накопило огромные вычислительные ресурсы именно в секторе ПК и вопрос об их использовании для вычислений до сих пор остается открытым.

На базе персональных компьютеров возможно построение целой инфраструктуры для решения достаточно больших вычислительных задач. Но к такой системе с другой стороны выдвигаются дополнительные требования. К ним можно отнести адаптацию и сохранение работоспособности в условиях динамического изменения доступных ресурсов: во-первых, система должна минимально влиять на удобство работы владельца машины, во-вторых, любой персональный компьютер в сети может быть выключен в любой момент и это не должно привести к краху системы.

Таким образом, задача использования вычислительных мощностей персональных компьютеров является крайне актуальной. Для ее решения необходима разработка гибких, самонастраивающихся, отказоустойчивых программных систем, которые позволили бы использовать ресурсы персональной компьютерной техники для решения больших и сложных задач.

Такие попытки были сделаны достаточно давно [2, 3]. Основным недостатком существующих систем является то, что участники, которые передали в общее пользование свой персональный компьютер, не решают свои задачи. Они скачивают и устанавливают себе на компьютер программный модуль (одинаковый для всех), который получает от главного сервера определенную порцию данных для расчета и по окончании вычислений возвращает серверу результат. Таким образом, подобные системы нельзя назвать многопользовательскими, они не пригодны для решения поставленной задачи.

**Описание системы**

Объектом нашего исследования является многопользовательская агентная вычислительная система (далее АВС), которая может разворачиваться на базе локальных и глобальных неоднородных компьютерных сетей. Под неоднородной сетью понимается сеть произвольной топологии с различными каналами связи, состоящая из компьютеров различной конфигурации и мощности, на которых установлены различные операционные системы. Такая вычислительная сеть предназначена для предоставления сервисов по обработке и хранению данных многих пользователей.

При разработке и моделировании АВС мы использовали агентный подход. В предложенной системе присутствуют три основных вида агентов:

– «Представители» – агенты, которые предоставляют пользователям интерфейсы для работы с системой;

– «Вычислители» – агенты, которые производят вычисления;

– «Контейнеры» – агенты, которые оперируют ресурсами компьютеров.

На каждой машине обязательно существует один экземпляр Контейнера, не имеющий возможности мигрировать, а Представители и Вычислители могут мигрировать согласно своей внутренней логике на более подходящую, с их точки зрения, машину.

Для компьютерного имитационного моделирования АВС перечисленные выше агенты были представлены в виде конечных автоматов.

**Результаты моделирования**

Целью моделирования было определение зависимости эффективности работы АВС от различных параметров (параметры и топология сети, распределение задач по трудоемкости, плотность потока, распределение компьютеров по мощности, алгоритмы агентов, заданных в виде конечных автоматов). Самой основной целью было выявить эффективные алгоритмы поведения агентов, при которых система сможет устойчиво и максимально эффективно справляться с потоком задач пользователей, а также найти область применимости такой системы. Было выявлено, что способность вычислителей к миграции хоть и приводит к значительным накладным расходам по передаче данных между Контейнерами, но оправдана с точки зрения равномерности нагрузки на систему. На рисунке показан один из результатов – зависимость эффективности использования растущих ресурсов системы при постоянном потоке задач от определенного числа пользователей (в данном случае их 10) для четырех различных ситуаций: 1. агенты не мигрируют; 2. способны мигрировать только Представители; 3. способны мигрировать только Вычислители; 4. могут мигрировать и Вычислители, и Представители.

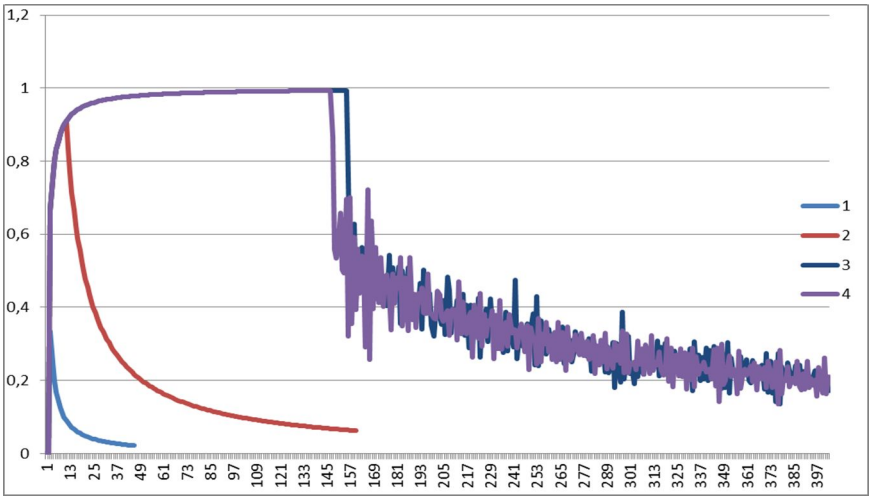


**Рис. 1. Эффективность работы системы для различных вариантов поведения агентов в зависимости от количества компьютеров**

Как видно из графика, на начальном этапе, когда в системе очень мало компьютеров, очевидно, они не справляются с потоком задач от пользователей и очередь задач растет. В систему входит положительный поток ресурсов, т.е. число компьютеров постоянно растет. Если агенты не мигрируют, то эффективность системы резко падает, т.к. все Представители, которые генерируют Вычислителей так и остаются на первом компьютере, а остальные простаивают. При этом очередь задач растет, т.к. система с ними не справляется. Красной линии на рисунке соответствует ситуация, когда Представители способны мигрировать. В этом случае эффективность первое время поднимается до 0,9, но потом опять стремительно падает. Это объясняется тем, что все Представители мигрировали в другие Контейнеры и равномерно распределились по системе. Далее они создают Вычислителей в своих Контейнерах, которые не могут мигрировать. Получается, что для обработки потока задач задействовано столько Контейнеров, сколько задействовано Представителями. Графики 3 и 4 очень схожи, т. к. отображают эффективность работы системы для ситуации, когда могут мигрировать Вычислители. Но поскольку в случае 4 мигрируют также и Представители, то можно сделать вывод, что миграция Представителей практиче-

ски не влияет на распределение нагрузки. Резкий обрыв графиков в момент времени, когда количество компьютеров превышает 150, объясняется тем, что на более раннем этапе система не могла справиться с входящим потоком задач. Затем настал такой переломный момент, когда количество компьютеров в системе стало больше накопленных задач – и все Вычислители благодаря способности мигрировать равномерно распределились по системе. Как раз в момент, когда к системе присоединился 150-й компьютер, накопленная очередь задач перестала накапливаться и дальше ресурсы используются не полностью.

Результаты моделирования показывают, что описанная выше АВС:

– может быть развернута на сети персональных компьютеров;

– система достаточно устойчива и может быть защищена от сбоев специальными алгоритмами;

– может работать при достаточно широком диапазоне внутренних параметров;

– имеет высокий КПД, порядка 95% для разных условий.

На основе алгоритмов жизненных циклов агентов, отработанных в результате моделирования АВС, с целью проверки адекватности результатов моделирования, был разработан действующий прототип системы.

**Прототип агентной вычислительной сети**

Архитектура системы представлена на рис. 2. Как видно из рисунка, она представляет собой иерархическую пятиуровневую систему. Самый базовый уровень – Аппаратный, который состоит из компьютеров и связывающего их сетевого оборудования. Второй уровень – Транспортный, обеспечивающий логическую связь между компьютерами. Третий уровень служит базой для построения агентной вычислительной сети. Им может быть любая платформа, которая дает возможность работать с такими сущностями, как агенты и обеспечивать общение между ними. Хорошим примером такой платформы является Jade [4] (именно его мы использовали для разработки прототипа системы). Следующий уровень системы содержит логику работы агентов, обеспечивает основные сервисы, такие как решение задач, хранение данных, интерфейсы пользователя, авторизацию и т.п. Разработка и выявление, а затем и оптимизация с помощью моделирования, основных алгоритмов работы агентов этого уровня как раз и является целью данного исследования. Пятый уровень дает возможность получить услуги по расчету необходимых задач.


**Рис. 2. Архитектура вычислительной системы**

Для реализации логического и прикладного уровня была создана иерархия классов, представленная на рис. 3.

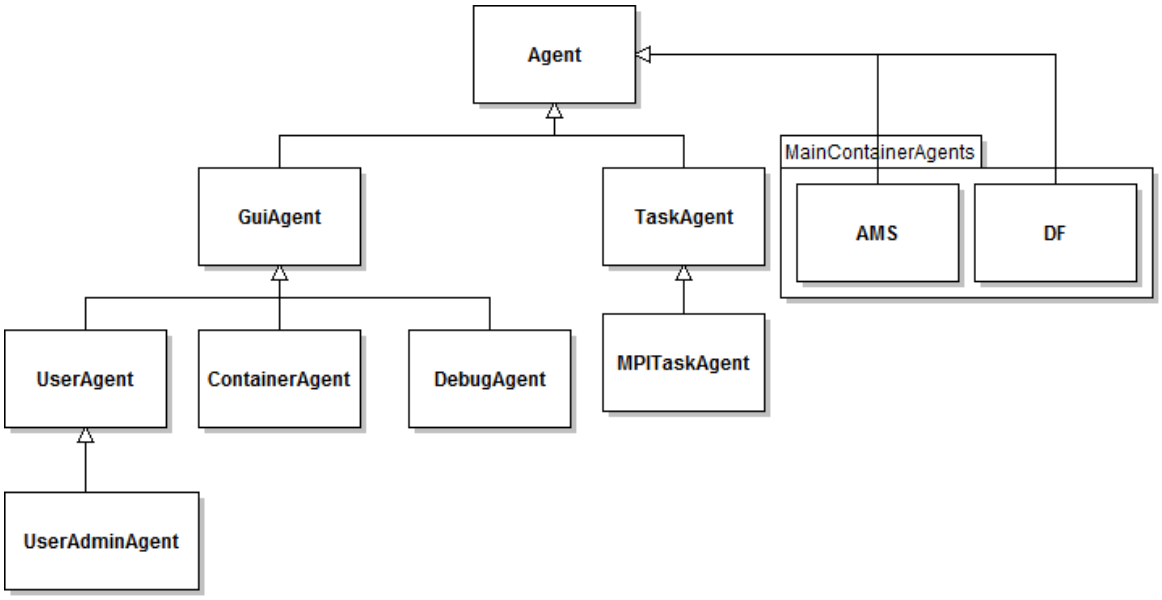


**Рис. 3. Иерархия агентов**

**Agent** – базовый для всех агентов системы класс, находится в пакете jade.core.agent. Содержит в себе готовый функционал по созданию, удалению, миграции агентов. **UIAgent** – агент с графическим интерфейсом. **UserAgent** – агент обычного пользователя. Он является виртуальным представителем пользователя в системе. Может мигрировать в другой контейнер при выключении машины или согласно внутренним алгоритмам. **UserAdminAgent** – агент администратора системы. Предоставляет возможность конфигурировать систему, администрировать пользователей системы, а также предоставляет сервис авторизации в системе. **DebugAgent** – агент для отладки работы системы. **ContainerAgent** – агент, который находится в каждом контейнере и отвечает за сбор и передачу статистики о доступном аппаратном обеспечении компьютера и его загруженности. Также предоставляет информационные сервисы о количестве **TaskAgent**-ов в своем контейнере. **TaskAgent** – агент, который создается под каждую новую задачу. На протяжении своего жизненного цикла этот агент благодаря миграции находит контейнер, пригодный для расчета задачи, затем производит расчет задачи и потом ожидает определенный период времени пользователя, чтобы отдать ему результаты вычислений, затем самоуничтожается. Этот агент привязан к **UserAgent**-у конкретного пользователя. **AMS и DF** – стандартные агент системы, автоматически запускаются при запуске главного контейнера. **AMS** (система управления агентами) обеспечивает службу имен (например, гарантирует, что каждый агент на платформе обладает уникальным именем) и осуществляет управление на платформе (к примеру, можно создавать и ликвидировать агентов на удаленном контейнере по запросу **AMS**). **DF** – (координатор каталог) обеспечивает сервис «желтых страниц», с помощью которого агенты могут искать друг друга по описанию предоставляемых сервисов.

Работа происходит в несколько этапов. Сначала пользователь пишет программный код для решения своей задачи согласно предоставленному шаблону. Он может не пользоваться шаблоном, но тогда Вычислитель, созданный под эту конкретную задачу, будет не-

управляемым и не сможет мигрировать в другой Контейнер в случае необходимости. Затем пользователь отправляет через своего Представителя программный код и подгружает необходимые данные. Представитель создает Вычислителя под данную задачу, и он уже действует самостоятельно ищет подходящий Контейнер, рассчитывает задачу и по окончании вычислений переходит в состояние ожидания, когда пользователь запросит результаты вычислений. По истечении определенного времени (выставляется администратором АВС) Вычислитель самоуничтожается. Все этапы решения всех своих задач пользователь может отследить через менеджера задач.

Испытания показали, что система является достаточно устойчивой к изменениям внешних условий и может с успехом применяться на практике.

**Заключение**

Доказана состоятельность предложенного подхода для построения многопользовательских вычислительных сетей и выявлены эффективные алгоритмы работы агентов, что позволяет на основе данных имитационного моделирования разработать действующий прототип системы. За основы был взять фреймворк Jade, который одновременно обеспечивает функционал компонентов мультиагентных систем и взаимодействие ее участников (агентов).

**Литература**

[1] Uwe Schwiegelshohna, Rosa M. Badiab, Marian Bubak Perspectives on grid computing // Future Generation Computer Systems – 2010. – Volume 26, Issue 8. – P. 1104–1115.
[2] SETI@home – Search for ExtraTerrestrial Intelligence at home [Электронный ресурс]. – Режим доступа: http://setiathome.berkeley.edu/. – оглавление с экрана.
[3] Rosetta@home – Protein Folding [Электронный ресурс]. – Режим доступа: http://boinc.bakerlab.org/rosetta/. – оглавление с экрана.
[4] Jade – Java Agent DEvelopment Framework [Электронный ресурс]. – Режим доступа: http://jade.tilab.com/. – оглавление с экрана.