ТЕОРЕТИЧЕСКИЙ И ПРИКЛАДНОЙ НАУЧНО-ТЕХНИЧЕСКИЙ ЖУРНАЛ

Издается с сентября 2010 г.

СОДЕРЖАНИЕ

Журнал зарегистрирован в Федеральной службе по надзору в сфере связи, информационных технологий и массовых коммуникаций. Свидетельство о регистрации ПИ № ФС77-38590 от 24 декабря 2009 г.

Журнал распространяется по подписке, которую можно оформить в любом почтовом отделении (индексы: по каталогу агентства "Роспечать"– **22765**, по Объединенному каталогу "Пресса России"– **39795**) или непосредственно в редакции.
Тел.: (499) 269-53-97. Факс: (499) 269-55-10.
Http://novtex.ru E-mail: prin@novtex.ru

УДК 004.94; 330.4; 519.683

В.Л. Макаров[1], д-р физ.-мат. наук, академик РАН, директор, e-mail: makarov@cemi.rssi.ru
А.Р. Бахтизин[1], д-р экон. наук, вед. науч. сотр., e-mail: albert@cemi.rssi.ru
В.А. Васенин[2], д-р физ.-мат. наук, проф., e-mail: vasenin@msu.ru
В.А. Роганов[2], стар. науч. сотр., e-mail: raduga@butovo.com
И.А. Трифонов[3], студент, e-mail: spaeciallyforyou@gmail.com

[1]Учреждение Российской академии наук Центральный экономико-математический институт РАН (ЦЭМИ РАН)
[2]НИИ механики МГУ имени М.В. Ломоносова
[3]МГУ имени М.В. Ломоносова

Средства суперкомпьютерных систем для работы с агент-ориентированными моделями

На примере разработанной в ЦЭМИ РАН агент-ориентированной модели рассматриваются этапы и методы эффективного отображения счетного ядра мультиагентной системы на архитектуру современного суперкомпьютера. За счет применения суперкомпьютерных технологий и оптимизации программного кода удалось добиться очень высокой производительности.

Ключевые слова: *агент-ориентированные модели, параллельные вычисления, суперкомпьютеры*

Введение

Компьютерное моделирование – интенсивно развивающаяся область научных исследований, востребованная сегодня практически во всех сферах человеческой деятельности. Агент-ориентированный подход к моделированию очень универсален и удобен для прикладников в силу своей наглядности, однако его, как правило, отличает и высокая требовательность к вычислительным ресурсам. Прямое моделирование, например, достаточно длительных социальных процессов в масштабах страны и планеты в целом требует вычислительных средств высокой производительности и больших объемов памяти.

Суперкомпьютеры позволяют на несколько порядков увеличить число агентов и других количественных характеристик (узлов сети, территории охвата) в моделях, первоначально разработанных для использования на традиционных ЭВМ с последовательной архитектурой. По этой причине суперкомпьютерное моделирование является логичным и крайне желательным шагом для тех упрощенных моделей, которые уже прошли успешную практическую апробацию на последовательных компьютерах. Вместе с тем, архитектурно-технологические особенности современных компьютеров вовсе не гарантируют, что программное обеспечение уже апробированной модели немедленно заработает и на суперкомпьютере. Для такого перехода как минимум потребуется распараллеливание счетного ядра, а зачастую и его глубокая оптимизация, поскольку, в ином случае, применение дорогостоящего суперкомпьютерного счета не будет оправдано.

В данной статье, на примере разработанной в ЦЭМИ РАН агент-ориентированной модели (АОМ) рассматриваются этапы и методы эффективного отображения счетного ядра мультиагентной системы на архитектуру современного суперкомпьютера.

1. Теоретические основы агент-ориентированных моделей. Сложившиеся определения АОМ

Определений АОМ достаточно много (см., например, работы [6, 7, 11]). Далее предлагается свое, которое с одной стороны является симбиозом определений, данных наиболее авторитетными экспертами в этой области, а с другой – отражает понимание авто-

рами моделей этого класса. Итак, АОМ – это модель, обладающая следующими основными свойствами.

• *Автономия*. Агенты действуют независимо друг от друга и при этом предполагается, что в моделях нет единой регулирующей структуры, которая контролировала бы поведение каждого агента в отдельности. Однако при этом взаимодействие микро- и макроуровней в моделях осуществляется, как правило, следующим образом: на макроуровне задается общий для всех агентов набор правил, и, в свою очередь, совокупность действий агентов микроуровня может оказывать влияние на параметры макроуровня.

• *Неоднородность.* Агенты чем-то различаются друг от друга, что принципиально отличает АОМ от широко распространенных моделей с агентом-представителем, причем различия между агентами могут проявляться по многим параметрам. В случае агентов, отображающих людей, это могут быть параметры уровня здоровья, дохода, культурного уровня, а также правил принятия решений и аналогичные им.

• *Ограниченная интеллектуальность агентов* (или ограниченная рациональность). Это свойство означает, что агенты модели не могут познать нечто большее, выходящее за рамки макросреды модели.

• *Расположение в пространстве*. Имеется в виду некоторая "среда обитания", которая может быть представлена как в виде решетки (как в игре "Жизнь" [13]), так и в виде гораздо более сложной структуры, например, трехмерного пространства с заданными в нем объектами.

Кроме перечисленных, общей особенностью всех АОМ и вместе с тем их главным отличием от моделей других классов является наличие в них большого числа взаимодействующих друг с другом агентов. Существуют АОМ, число агентов в которых достигает нескольких миллионов, например, модель, разработанная под руководством Дж. Эпштейна [18]. Обычно в моделях социально-экономических систем присутствуют агрегированные агенты, представляющие собой либо отрасль, либо регион, либо совокупное домохозяйство. Спецификация агента при этом проводится путем оптимизации соответствующей функции полезности или в модель включаются рассчитанные ранее экзогенные параметры, отражающие результаты решений агента. В литературе эти два подхода часто подвергаются обоснованной критике, поскольку в большинстве случаев они не всегда позволяют получить в рамках таких моделей соответствующие реалиям оценки взаимодействия агрегированных агентов. В то же время за счет более детальной спецификации в АОМ агентов микроуровня можно более адекватным действительности образом добиться изменений параметров макроуровня.

Подводя итог изложенным выше соображениям, отметим, что согласно перечисленным свойствам агент в АОМ является автономной сущностью, как правило, имеющей графическое представление, с определенной для него целью функционирования и возможностью обучения в процессе существования до заранее принятого уровня, который определяется разработчиками соответствующей модели. Примерами агентов могут быть: люди (равно как и другие живые организмы), роботы, автомобили и другие подвижные объекты; недвижимые объекты; совокупности однотипных объектов. Вообще говоря, агентами в АОМ могут быть любые наблюдаемые в реальной жизни объекты, однако основной задачей описания таких агентов в рамках модели является их корректная, в большей степени отражающая реалии спецификация.

Что может быть "внутри" агента?

Как правило, для описания агента используются параметры, переменные, функции, поведенческие диаграммы, представляющие собой, например, схемы UML (*Unified Modeling Language*), отражающие состояния агентов в определенный момент времени.

Агенты, имеющие графическое представление, могут перемещаться в рамках:

• евклидова пространства (2D или 3D);

• ГИС (геоинформационной системы или системы, позволяющей создавать базы данных, сочетающие в себе графическое и атрибутивное представление разнородной информации);

• решетки (в этом случае перемещение агентов происходит строго из одной ячейки в другую);

• сетевой структуры.

Следует заметить, что необходимости в графическом представлении состояния агентов АОМ в пространстве не возникает.

Из истории возникновения АОМ

Концептуальный прототип первой АОМ был разработан в конце 1940-х гг. Однако широкое распространение эти модели получили в начале 1990-х гг. благодаря появлению на IT-рынке микро- и миникомпьютеров относительно высокой производительности и возможности проводить с их помощью крупномасштабные имитационные эксперименты. Принято считать, что АОМ берут свое начало с вычислительных машин Джона фон Неймана, являющихся теоретическими машинами, способными к самовоспроизводству [14]. Джон фон Нейман предложил использовать машины, которые следуют детальным инструкциям для создания точных копий самих себя. Впоследствии данный подход был усовершенствован его другом Станиславом Уламом, который предложил изображать машину на бумаге в качестве набора клеток на решетке [21]. Данный подход стал началом развития клеточных автоматов. Наиболее известной реализацией взаимодействия конечных автоматов стала игра "Жизнь", предложенная Джоном Хортоном Конвеем (*John Horton Conway*), отличающаяся от машины фон Неймана достаточно простыми правилами поведения агентов [13]. Одновременно с перечисленными выше возникла новая методология научного исследования – компьютерное имитационное моделирование, которое в настоящее время включает следующие ос-

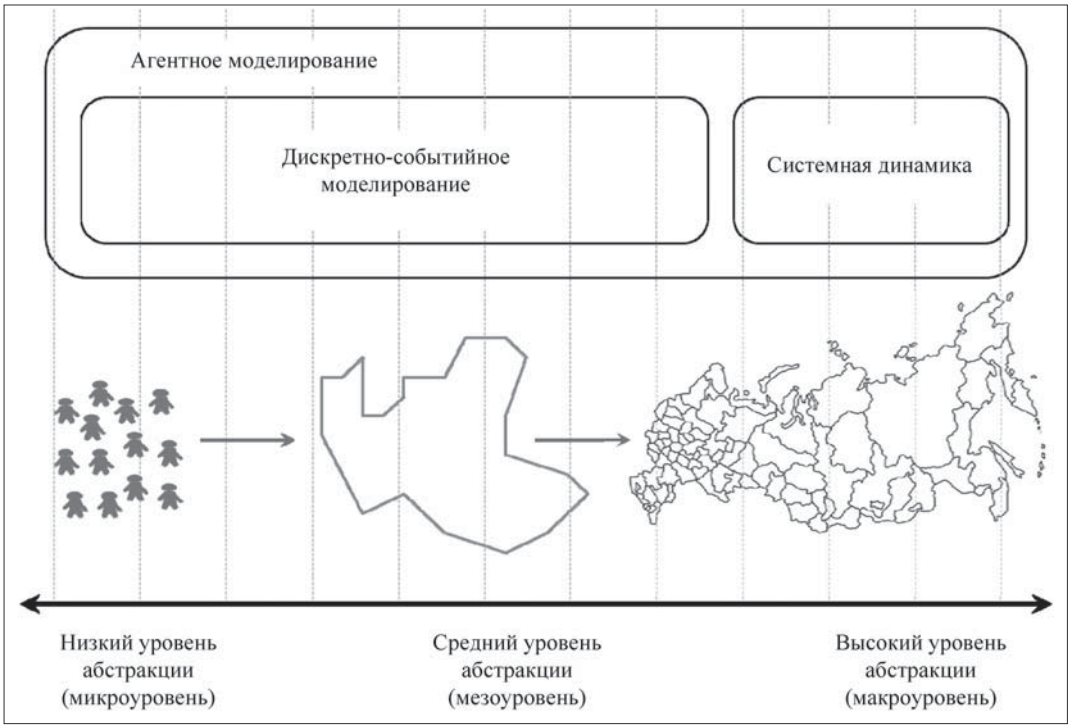

Рис. 1. Соотношение основных подходов к имитационному моделированию различных объектов и трех уровней абстракции

новные направления: системная динамика (СД); дискретно-событийное моделирование (ДС); агентное моделирование. Все эти виды моделирования применяются, в том числе, для решения социальных и экономических задач на разных уровнях абстракции. Агентное моделирование, развитие которого напрямую определяется увеличивающимися вычислительными возможностями современных компьютеров, позволяет представить (смоделировать) систему практически любой сложности из большого числа взаимодействующих объектов, не прибегая к их агрегированию. Появились программные средства (см. раздел 2), позволяющие сочетать все перечисленные выше направления имитационного моделирования.

Следует, однако, отметить, что наибольшие трудности возникают при совмещении объектов разного уровня абстракции в рамках одной модели. В этой связи разработчики математических моделей социально-экономических систем все чаще ставят вопрос об актуальности построения иерархических динамических моделей, включающих субъекты макроуровня и агентов микроуровня, поведение которых должно быть описано более адекватно реально наблюдаемому, нежели применяемые на практике методы их представления. Подходы на основе АОМ позволяют совмещать в себе агентов различного уровня абстракции и более того практически любая модель, основанная на двух других парадигмах имитационного моделирования (СД и ДС) может быть легко конвертирована в АОМ и уже в таком качестве использовать преимущества агентного подхода.

На рис. 1 в виде схемы отображено соотношение различных подходов к моделированию объектов различной степени детализации. Как видно, СД в основном используется для решения задач на высоком уровне абстракции. В свою очередь, модели ДС используются на нижнем и среднем уровнях и, как уже отмечалось, АМ применимо на всех уровнях [1]. Заметим, что к высокому уровню абстракции относятся, например, задачи прогнозирования динамики населения страны. На нижнем уровне реализуются микроуровневые модели (например, модели движения пешеходов). К среднему уровню относятся задачи, связанные, например, с оптимальным планированием перевозок внутри региона и подобные им.

С середины 1990-х гг. АОМ стали использовать для решения большого числа востребованных бизнесом и технологических задач, например:

- оптимизация сети поставщиков и планирование перевозок;
- планирование развития производства;
- прогнозирование спроса на продукцию и объема продаж;
- оптимизация численности персонала;
- прогнозирование развития социально-экономических систем (городов, регионов);
- моделирование миграционных процессов;
- имитация и оптимизация пешеходного движения;

• моделирование транспортных систем;

• прогнозирование экологического состояния окружающей среды.

Этот список можно продолжить.

Преимущества АОМ

Преимущества АОМ перед другими средствами имитационного моделирования заключаются в следующем.

• Методы на основе АОМ позволяют моделировать систему, максимально приближенную к реальности. Как уже отмечалось ранее, степень детализации АОМ, по сути, ограничивается только возможностями компьютеров. Более того, в ряде АОМ передвижение агентов задается без использования сложных формул, однако с помощью заранее определенных маршрутов и простых правил. Такие правила, с одной стороны, имитируют адаптивное мышление в процессе принятия решений, а с другой – позволяют получить неочевидные результаты на уровне агрегированных параметров. Примерами таких АОМ могут быть модели, имитирующие передвижение пешеходов, покупателей в крупных торговых центрах, спецтехники на складах и другие.

• Агентные модели обладают свойством эмерджентности. Например, в одной из разработанных в ЦЭМИ РАН АОМ имитируется работа транспортной системы г. Москвы, в ходе моделирования которой определялось поведение только отдельных агентов, в то время как более общие явления – автомобильные пробки или параметр, отражающий уровень загруженности дорог города, определялись уже в процессе расчетов с использованием другой модели [17].

• Как следует из предыдущего пункта, важным преимуществом агентного моделирования является возможность построения моделей с учетом отсутствия знаний о глобальных зависимостях в рамках моделирования соответствующей предметной области. Важно представлять логику поведения отдельных агентов, что, в свою очередь, может помочь в получении более общих знаний об изучаемом процессе.

• Агент-ориентированное моделирование является гибким инструментом, позволяющим легко добавлять и удалять агентов в модели, а также менять параметры и правила их поведения [7].

Считается, что агентное моделирование дополняет традиционные аналитические методы, а также включает в себя упомянутые выше подходы имитационного моделирования, поскольку последние могут применяться "внутри" агентной модели при формализации ее отдельных активных объектов или агентов [4]. Появление АОМ можно рассматривать как результат эволюции методологии моделирования, а именно – переход от мономоделей (одна модель – один алгоритм) к мультимоделям (одна модель – множество независимых алгоритмов).

АОМ могут объяснить причину возникновения таких явлений, как войны, обрушение рынка акций и других. В идеале АОМ могут помочь идентифицировать критические моменты времени, после наступления которых чрезвычайные последствия будут иметь необратимый характер.

* * *

Большая часть АОМ представлена в свободном доступе в Интернет (на страницах ученых-разработчиков или на популярных порталах, содержащих информацию по данной проблематике). Так, существуют специализированные издания, тематика которых напрямую связана с разработкой АОМ, например, онлайновый журнал JASSS (*Journal of Artificial Societies and Social Simulation*, http://jasss.soc.surrey.ac.uk/JASSS.html), уже ставший достаточно известным. Отметим также российский ежеквартальный интернет-журнал "Искусственные общества" (www.artsoc.ru).

Следует также упомянуть про новое направление в прикладной экономике – агент-ориентированную экономику (*Agent-based Computational Economics*, ACE), основой которой является моделирование виртуального мира, "населенного" автономными агентами (экономическими, биологическими и другой природы). В проект по созданию подобных миров вовлечено много исследователей, разработки которых представлены на сайте www.econ.iastate.edu/tesfatsi. Управление созданным виртуальным миром в соответствии с методологией ACE осуществляется без вмешательства извне, т.е. только посредством взаимодействия агентов [19]. При этом агенты должны обладать способностью к обучению.

2. Программное обеспечение для разработки агент-ориентированных моделей

В настоящем разделе перечислим наиболее известное программное обеспечение для построения АОМ. Вообще говоря, разработка АОМ не требует использования специализированных программ. Можно применять средства разработки широкого профиля (C#, Delphi и другие), однако специализированные программы содержат набор готовых библиотек для эффективного представления агентов и их среды. Реализация таких операций, как визуализация двумерной решетки, перемещение агентов по ней и ряд других с использованием подобных программ значительно упрощена, и в этой связи предпочтительнее использовать имеющийся арсенал описываемых далее средств.

Пакет SWARM является набором библиотек, написанных на языке Objective-C, служащих основой для разработок сложных мультиагентных систем. Этот пакет в свободном доступе представлен в Интернет по адресу http://www.swarm.org.

REPAST (*REecursive Porous Agent Simulation Toolkit*) Symphony представляет собой инструментальный комплекс IDE (*Integrated Development Environment*) для разработки АОМ, по функциональным возможностям сравнимый с описанным далее пакетом AnyLogic. Из достоинств REPAST следует отметить:

✓ удобство разработки моделей;

✓ открытые исходные коды и возможность использования дистрибутива бесплатно;

✓ удобные методы презентации;

✓ встроенную многопоточность.

Простой и доступной программой для разработки АОМ является бесплатно распространяемое приложение NetLogo. Изначально NetLogo был разработан как учебный инструмент, однако сейчас им пользуются не только студенты, но и тысячи исследователей. Он представляет собой программу, которая часто применяется в вузах для обучения студентов основам АОМ. Схожей функциональностью обладает программа StarLogo.

Следует также упомянуть программные средства MASON, EcoLab, SOARS, Cormas, однако больших отличий от предыдущих средств разработки АОМ у них нет.

Отдельного внимания заслуживает пакет AnyLogic [2]. Данный пакет является продуктом нового поколения для разработки и исследования имитационных моделей. Отметим, что он является единственным российским профессиональным инструментом имитационного моделирования, успешно конкурирующим с зарубежными аналогами.

В отличие от перечисленных выше пакетов для построения АОМ, AnyLogic является коммерческим продуктом, и в этой связи он лишен многих недостатков, присущих *open source* продуктам, ответственность за которые их разработчики снимают с себя, главным образом, в силу бесплатного характера распространения.

AnyLogic поддерживает различные подходы к моделированию, в том числе и агентный, для которого содержит специальную библиотеку классов AnyLogic *agent-based library*, предоставляющую возможность задания требуемой функциональности у агентов модели. Следует также отметить, что AnyLogic поддерживает все возможные способы задания поведения агентов – диаграммы состояний (стейтчарты), синхронное и асинхронное планирование событий.

Пакет AnyLogic обладает хорошими средствами визуализации и позволяет имитировать различные процессы производства, бизнеса и др.

Более подробно про саму среду разработки AnyLogic и некоторые теоретические аспекты имитационного моделирования можно прочитать в книге [2].

Основным недостатком перечисленных выше пакетов является отсутствие невозможности разработки проектов, выполняющихся на вычислительном кластере (не предусмотрен механизм распараллеливания процесса выполнения программного кода). Далее рассмотрен опыт зарубежных ученых и практиков по запуску АОМ на суперкомпьютерах.

3. Опыт зарубежных ученых и практиков

В сентябре 2006 г. стартовал проект по разработке крупномасштабной АОМ европейской экономики (EURACE, *Europe ACE – Agent-based Computational Economics*) с очень большим числом автономных агентов, взаимодействующих в рамках социально-экономической системы [8]. В выполнение этого проекта вовлечены экономисты и программисты из восьми научно-исследовательских центров Италии, Франции, Германии, Великобритании и Турции, а также консультант из Колумбийского университета США – нобелевский лауреат Джозеф Стиглиц.

Исследование базируется на методологии ACE в целях преодоления ограничений, лежащих в основе широко распространенных моделей, рассматривающих агрегированных агентов, а также предполагающих их рациональное поведение и состояние равновесия. Для модели используется ГИС с широким перечнем рассматриваемых объектов – предприятия, магазины, школы, транспортные сети и другие.

Как отмечают разработчики, практически все существующие АОМ рассматривают либо отдельные отрасли, либо относительно небольшой географический район и, соответственно, небольшую популяцию агентов. В свою очередь в рамках EURACE рассматривается весь Европейский Союз. И по своим масштабам и сложности разрабатываемая модель является уникальной, а ее численное разрешение требует использования суперкомпьютеров и специально разработанного программного обеспечения.

Для наполнения модели статистической информацией использовались данные (в виде ГИС-карт) статистической службы Европейского союза уровня NUTS-2[1], представляющие информацию о 268 регионах 27 стран Европейского союза.

В модели три типа агентов: домашние хозяйства (около 10^7), предприятия (около 10^5) и банки (около 10^2). Все они имеют географическую "привязку", а также связаны друг с другом посредством социальных сетей, деловых и других отношений.

EURACE реализован с помощью гибкой, масштабируемой среды для моделирования агентных моделей FLAME (*Flexible Large-scale Agent Modeling Environment*), разработанной С. Коакли и М. Холкомбом (более подробно см. www.flame.ac.uk), изначально созданной для моделирования роста биологических клеток, выращиваемых в различных условиях. Используемый во FLAME подход основан на так называемых *X*-машинах, напоминающих конечные автоматы, однако отличающихся от них. Суть в том, что у каждой *X*-машины есть набор данных (своего рода память), а переходы между состояниями являются функциями не только состояния, но и набора данных памяти.

[1] NUTS (фр. *nomenclature des unités territoriales statistiques*) – номенклатура территориальных единиц для целей статистики – стандарт территориального деления стран для статистических целей, разработанный Европейским Союзом и детально охватывающий только входящие в него страны. Существуют NUTS-единицы трех уровней, при этом второй уровень (NUTS-2) соответствует административным округам Германии, графствам в Великобритании и т.д.

Таким образом, каждый агент в системе FLAME представлен *X*-машиной, причем предусмотрено общение между агентами посредством передачи сообщений. При работе в параллельном режиме на суперкомпьютере обмен сообщениями между агентами требует больших вычислительных затрат, в связи с чем агенты изначально были распределены по процессорам в соответствии со своим географическим положением. Тем самым разработчики программы минимизировали вычислительную нагрузку, исходя из первоначального предположения о том, что в основном общение между агентами происходит в рамках небольшой социальной группы, проживающей приблизительно в одной местности. Таким образом, весь модельный ландшафт был поделен на небольшие территории и распределен между узлами суперкомпьютера.

С помощью разработанной модели был проведен ряд экспериментов в целях исследования рынка труда. Не рассматривая детально получившиеся числовые результаты, отметим, что, по мнению авторов, основной вывод исследования заключается в том, что даже у двух регионов со схожими условиями (ресурсы, развитие экономики и др.) в течение продолжительного периода (10 лет и более) макропоказатели могут значительно измениться за счет первоначальной неоднородности агентов.

Подробнее информацию организационного и научного содержания по этому проекту можно найти на веб-странице проекта: www.eurace.org.

В рамках АОМ EpiSims, разработанной исследователями из Института биоинформатики Вирджинии (*Virginia Bioinformatics Institute*), рассматриваются как перемещения агентов, так и их контакты в рамках среды, максимально приближенной к реальности и включающей в себя дороги, здания и прочие инфраструктурные объекты [9]. Для построения модели потребовался большой массив данных, включающий информацию о здоровье отдельных людей, их возрасте, доходе, этнической принадлежности и ряде других характеристик.

Изначально цель исследования заключалась в построении АОМ большой размерности для ее использования на суперкомпьютере, с помощью которой можно будет изучать вопросы распространения болезней в обществе. Однако впоследствии в процессе работы решалась также задача по созданию специализированного программного обеспечения ABM++, которое позволяет осуществлять разработку АОМ на языке C++, а также содержит функции, облегчающие распределение исполняемого программного кода на отдельные узлы суперкомпьютера. Кроме перечисленных функций, ABM++ предоставляет возможность для динамического перераспределения потоков вычислений, а также методы по синхронизации происходящих событий.

ABM++ является модернизированной версией разработанного в 1990–2005 гг. в Лос-Аламосской национальной лаборатории (США) инструментального комплекса, появившегося в процессе построения крупномасштабных АОМ (EpiSims, TRANSIMS, MobiCom). Модернизация этого инструментария и появление первой версии ABM++ произошли в 2009 г.

Межпроцессорные связи между вычислительными узлами в АОМ часто требуют синхронизации происходящих в модели событий. Средства ABM++ позволяют разрабатывать модели, отвечающие этому требованию. Например, в социальных моделях агенты часто перемещаются между различными точками пространства (работа, дом и другие), а на программном уровне этому соответствует смена узла кластера и здесь важно, чтобы модельное время принимающего узла было синхронизировано со временем узла, который агент только что покинул.

Также в ABM++ реализована библиотека MPIToolbox, которая соединяет интерфейс C++ API (*Application Programming Interface*) и *Message Passing Interface* (MPI) суперкомпьютера и обеспечивает более быструю реализацию задач по передаче данных между узлами кластеров.

Разработка ABM++ осуществлялась в Ubuntu Linux – операционной системе с компиляторами gcc/g++. В качестве интегрированной среды разработки рекомендуется Eclipse с модулем расширения (плагином) для поддержки C и C++, а также с плагином PTP (*Parallel Tools Platform*), обеспечивающим разработку и интеграцию приложений для параллельных компьютерных архитектур. Среда Eclipse поддерживает интеграцию с TAU (*Tuning and Analysis Utilities*) Performance System – инструментальным комплексом для анализа и отладки параллельных программ, что также упрощает разработку агентных моделей.

Специалисты другой исследовательской группы также из Института биоинформатики Вирджинии разработали инструментарий для изучения особенностей распространения инфекционных заболеваний внутри различных групп населения – EpiFast. К его достоинствам можно отнести масштабируемость и высокую скорость исполнения. Например, имитация социальной активности жителей Большого Лос-Анджелеса (агломерации с населением свыше 17 млн человек) с 900 млн связей между людьми на кластере с 96 двухъядерными процессорами POWER5 заняла менее 5 минут. Такая достаточно высокая производительность связана с механизмом распараллеливания, предложенным в работе [16].

Как правило, для реализации модели социума на группе процессоров используют несколько естественных способов. Например, один из них базируется на представлении социума в виде набора неструктурированных связей между индивидуумами. При распараллеливании эти связи равномерно делятся на группы, число которых соответствует числу процессоров. Отрицательная сторона такого подхода заключается в сложности отслеживания статуса отдельного человека. Если, например, *i*-й индивидуум инфицирован, то информация об этом

должна быть синхронизирована во всех группах, так как нет сведений о том, в каких из них содержатся связи данного человека. В свою очередь, это влечет за собой большую вычислительную нагрузку. Другой подход основан на разделении людей – агентов модели в соответствии с их географическим месторасположением. Однако в этом случае, учитывая, что население обычно распределено неравномерно, распределение вычислительной нагрузки требует применения достаточно сложных алгоритмов для ее выравнивания, что также создает дополнительную вычислительную нагрузку.

Предлагаемый в настоящей работе метод заключается в равномерном распределении агентов, с соответствующими им односторонними исходящими связями по группам, число которых равно числу процессоров. Соответствующий этому методу вычислительный алгоритм основан на взаимодействии ведущих (*master*) и ведомых (*slave*) процессоров, организованном следующим образом: ведущий процессор "знает", какой из процессоров обслуживает конкретного агента, а каждый из ведомых процессоров "знает" только обслуживаемых (локальных) агентов. В процессе вычислений ведомые процессоры посылают запросы ведущим процессорам относительно связей с нелокальными агентами. В свою очередь ведущие процессоры не осуществляют никаких расчетов, кроме поиска нелокального агента для каждой внешней связи и пересылают запросы соответствующим процессорам. Таким образом, по мнению разработчиков, предлагаемый алгоритм позволяет проводить эффективное и высокоскоростное имитационное моделирование систем с очень большим числом агентов.

Классические модели распространения эпидемий преимущественно основывались на использовании дифференциальных уравнений, однако такой аппарат затрудняет учет связей между отдельными агентами и многие другие их индивидуальные особенности. В свою очередь, АОМ позволяют преодолеть указанные недостатки. Так, в 1996 г. Дж. Эпштейн и Р. Акстелл опубликовали описание одной из первых АОМ, имитирующей процесс распространения эпидемий [10]. Агенты модели, отличающиеся друг от друга восприимчивостью к заболеванию, зависящей от состояния иммунной системы, географически распределены в рамках определенной территории. Вместе с тем, по мнению авторов модели, агенты, число которых всего несколько тысяч, реализуют достаточно примитивное поведение.

В дальнейшем под руководством Дж. Эпштейна и Дж. Паркера в Центре социальной и экономической динамики в Брукингском институте (*Center Social and Economic Dynamics at Brookings*) была построена одна из самых больших АОМ, включающая в себя все население США, т.е. порядка 300 млн агентов [18].

Построенная модель удовлетворяет нескольким требованиям. Во-первых, она позволяет предсказать последствия распространения заболеваний различного типа. Во-вторых, модель ориентирована на поддержку двух сред для вычислений. Одна из них состоит из кластеров с установленной 64-битной версией операционной системы (ОС) Linux, а другая – из серверов с четырехъядерными процессорами и установленной на них ОС Windows. В этой связи в качестве языка программирования был выбран Java , хотя разработчики и не уточняют, какую именно реализацию Java они тогда использовали. Последнее требование заключается в возможной поддержке численности агентов от нескольких сотен миллионов до 6 миллиардов.

Способ распределения агентов между аппаратными ресурсами состоит из двух фаз. Первая фаза заключается в распределении агентов по компьютерам, задействованным в работе, а вторая – в распределении агентов модели по потокам на каждом из компьютеров. В процессе моделирования каждый поток может остановиться (в заранее обусловленное время) для передачи сообщений другим потокам. Все подготовленные к отправке сообщения до определенного момента времени хранятся в пуле сообщений, а затем одновременно отправляются на обработку.

В реализации модели есть также две вспомогательные утилиты. Первая из них управляет потоками на отдельном компьютере, а вторая следит за тем, чтобы все сообщения между потоками были выполнены до момента возобновления вычислительного процесса.

При распределении агентов по аппаратным ресурсам следует учитывать следующие два обстоятельства: производительность узла напрямую зависит от числа инфицированных агентов; контакты, предполагающие передачу сообщений между потоками, требуют гораздо больше вычислительных затрат, нежели контакты, ограничивающиеся локальной информацией. Исходя из этого, возможно различное распределение агентов. С одной стороны, можно поделить все рассматриваемое географическое пространство на равные части, число которых должно соответствовать числу узлов, а затем определить для каждого узла какой-либо географический регион. За счет этого возможна балансировка вычислительной нагрузки между узлами. С другой стороны, можно закрепить определенную территорию, представляющую собой единую административную единицу, за конкретным узлом. Это может способствовать снижению вычислительной нагрузки за счет уменьшения числа контактов, требующих передачи сообщений между потоками. Если первый способ влечет за собой увеличение вычислительной нагрузки за счет ресурсоемких контактов, то второй способ в ряде случаев несет в себе потенциальную опасность значительного дисбаланса между аппаратными ресурсами. Вспышка, например, какого-либо заболевания в одном из регионов загрузит один из вычислительных узлов, в то время как некоторые другие будут простаивать.

Разработанная модель (*US National Model*) включает в себя 300 млн агентов, перемещающихся по карте страны в соответствии с матрицей корреспонденций

размерностью 4 000×4 000, специфицированной с помощью гравитационной модели. С ее помощью был проведен вычислительный эксперимент, имитирующий 300-дневный процесс распространения болезни, особенности которой заключаются в 96-часовом периоде инкубации и 48-часовом периоде заражения. Один из результатов исследования заключался в том, что распространение заболевания идет на спад после того, как 65 % людей выздоровели и приобрели иммунитет. Эта модель была неоднократно использована в Университете Джонса Хопкинса (*Johns Hopkins University*), а также в Департаменте национальной безопасности США для исследований по быстрому реагированию на различного рода эпидемии [12].

В 2009 г. на основе описанной выше модели была разработана ее вторая версия, включающая 6,5 млрд агентов, спецификация действий которых проводилась с учетом имеющихся статистических данных. С ее помощью имитировались последствия от распространения вируса гриппа A(H1N1/09) в масштабах всей планеты.

Отметим, что подобная модель была разработана ранее в Лос-Аламосской национальной лаборатории. Результаты работы с этой моделью были представлены в широкой печати 10 апреля 2006 г. [5]. Для имитационного моделирования на основе этой модели был использован один из мощнейших по тем временам суперкомпьютеров Pink, состоящий из 1024 узлов, с двумя процессорами с частотой 2,4 ГГц и 2 ГБ памяти на каждом из них. С помощью этой крупномасштабной модели, включающей 281 млн агентов, были рассмотрены сценарии распространения различных вирусов (в том числе и H5N1) с учетом различного рода оперативных вмешательств (вакцинация, закрытие школ и введение карантина на некоторых территориях).

Естественно, что для подобных задач можно использовать не только суперкомпьютер, а более дешевые решения. Показателен пример ученых Университета Ньюкасла из Австралии (*The University of Newcastle, Australia*), построивших кластер, включающий в себя 11 узлов – отдельных персональных компьютеров, объединенных в сеть с пропускной способностью до 100 Мбит/с [20].

Программное обеспечение узлов кластера было одинаковое – ОС Debian GNU/Linux и JavaParty – кроссплатформенное программное расширение языка Java (ПО JavaParty включает в себя средства для распределения потоков выполнения кода по узлам кластеров).

Для оценки производительности кластера были использованы три версии одной АОМ, в рамках которой тестировались различные стратегии агентов, принимающих участие в аукционе. Версии различались сложностью стратегий агентов.

Основной вывод по результатам тестирования заключается в том, что использование кластеров оправдано лишь в том случае, если вычислительная нагрузка отдельных узлов достаточно интенсивная. В противном случае распределение потоков наоборот уменьшает скорость работы модели. Отметим, что такое поведение кластерных приложений не является традиционным.

Представляет большой интерес опыт ученых из Оак-Риджской национальной лаборатории (*Oak Ridge National Laboratory*) по распараллеливанию агентных моделей на суперкомпьютерах, построенных с использованием графических процессоров (*graphics processing unit,* GPU) вместо обычных центральных процессоров (*central processing unit,* CPU). Напомним, что для графических процессоров компания NVIDIA разработала специальную архитектуру параллельных вычислений – CUDA (*Compute Unified Device Architecture*), которая позволяет эффективно управлять памятью графического ускорителя, организовывать доступ к его набору инструкций и эффективно организовывать параллельные вычисления, используя упрощенную версию языка C.

Основное отличие между CPU и GPU (помимо механизмов доступа к памяти) заключается в том, что первые созданы для выполнения одного потока последовательных инструкций с максимальной производительностью, а GPU – для выполнения большого числа параллельных потоков.

В настоящее время решения на базе GPU приобретают большую популярность. Так ПК на базе процессора NVIDIA Tesla C2070 с 448 ядрами CUDA может обеспечить пиковую производительность до 1 терафлопа, т.е. он в сотни раз быстрее обычного ПК. Кроме того, подобный суперкомпьютер имеет компактные размеры (чуть больше обычного системного блока), не требует дополнительных систем охлаждения и электропитания (требуется лишь блок питания от 700 Вт до 1 кВт в зависимости от конфигурации). В свою очередь стоимость такого персонального суперкомпьютера с такой потрясающей для своих размеров производительностью составляет 6 000–7 000 долл. США, что несопоставимо меньше стоимости суперкомпьютеров на основе CPU.

Ученые Оак-Риджской национальной лаборатории К. Перумалла и Б. Ааби в многочисленных статьях, размещенных на сайте http://kalper.net/kp, описывают алгоритмы запуска АОМ на суперкомпьютерах двух видов (на основе графических и центральных процессоров). В общих чертах эти алгоритмы схожи с описанными выше. Они основаны на распределении агентов по процессорам, исходя из их географического расположения, однако больший интерес в этих статьях представляет сравнение производительности CPU и GPU при реализации некоторых агентных моделей, предусматривающих графическое отображение.

Например, при тестировании производительности выполнения игры "Жизнь" размерностью 3 750×3 750 (с максимальным числом агентов приблизительно равным 14 млн), по словам исследователей, скорость GPU была почти в 14 раз выше, чем у CPU [15]. Авторы при этом отмечают, что такое преимущество

имеет место только для отдельных задач и, конечно же, GPU не являются полноценной заменой CPU (в их нынешнем виде они просто для этого и не предназначены).

Стоит отметить, что как GPU приближаются к CPU, становясь более универсальными за счет увеличивающихся возможностей по расчетам чисел с двойной и одинарной точностью, так и в CPU увеличивается число ядер, а соответственно, и способность к параллельным расчетам.

4. Адаптация агент-ориентированных моделей для суперкомпьютера: предлагаемый подход

Проблема масштабирования

Важно понимать, что проблема масштабирования программ для суперкомпьютеров довольно фундаментальна. Несмотря на то, что и традиционная последовательная, и параллельная суперкомпьютерная программы реализуют одни и те же функции, целевые установки на их разработку, как правило, разные.

При начальной разработке сложного прикладного ПО, в первую очередь, стараются сократить издержки на программирование, обучение персонала, повысить уровень переносимости с одной платформы на другую и добиться ряда других целей, а оптимизацию откладывают "на потом". Такой подход вполне разумен, так как на ранних стадиях приоритетом разработки является исключительно выполнение функций. Однако после того, как разработанное ПО уже начало внедряться, часто выясняется, что на больших реальных данных для его реализации не хватает производительности. Принимая во внимание тот факт, что современные суперкомпьютеры – это вовсе не "разогнанные" в тысячи раз ПК, для запуска на суперкомпьютере программу приходится существенно видоизменять. Причем эффективно сделать это без специальных знаний и навыков удается далеко не всегда.

При грамотно проведенной работе значительное повышение эффективности обычно достигается на трех уровнях:

- распараллеливание счета;
- специализация вычислительных библиотек, учитывающая особенности задачи;
- низкоуровневая оптимизация.

Специализация и низкоуровневая оптимизация

Перед тем, как начать вычислительные эксперименты на суперкомпьютере, программу следует максимально оптимизировать и адаптировать к целевой аппаратной платформе. Если этого не сделать, то параллельная версия будет лишь хорошим тестом для суперкомпьютера, а собственно сам счет будет весьма неэффективным. По аналогии – нецелесообразно посылать на боевое задание полк необученных новобранцев. Их нужно сначала хорошо обучить выполнять их задачу (специализация, оптимизация программного обеспечения), а также научить эффективно владеть оружием (низкоуровневая оптимизация программного обеспечения). Только в этом случае использование суперкомпьютера будет по-настоящему эффективным.

В универсальных системах моделирования типа AnyLogic предоставляемые процедуры универсальны. А универсальный код часто можно прооптимизировать для конкретного семейства задач.

Выбор системы поддержки моделирования

Прежде всего следует заметить, что АОМ можно программировать и без специальной среды, непосредственно на любом объектно-ориентированном языке (об этом уже упоминалось ранее).

Однако, несомненно, более разумным подходом будет использование одной из хорошо зарекомендовавшей себя системы для АОМ, по причине унифицированной реализации типичных способов взаимодействия агентов. Наиболее известные из них уже были отмечены выше, здесь же рассмотрим систему ADEVS.

Программная система ADEVS представляет собой набор низкоуровневых библиотек для дискретного моделирования, выполненных на языке C++. Из ее достоинств следует отметить:

– простоту реализации;

– высокую производительность моделей;

– поддержку основных численных методов при построении моделей;

– встроенное распараллеливание процесса имитационного моделирования (симуляции) при помощи OpenMP;

– возможность применения стандартных средств распараллеливания;

– достаточно быстрое на настоящее время развитие библиотек;

– кроссплатформенность;

– низкоуровневость (текущие функциональные требования не накладывают никаких ограничений на модель);

– независимость скомпилированного кода от нестандартных библиотек;

– открытый исходный код.

Однако существенными недостатками этого продукта являются полное отсутствие средств презентации и достаточно сложная, по сравнению, например, с AnyLogic, разработка моделей. Как следствие, этот продукт не может использоваться для построения моделей на уровне заказчика. Однако он представляет собой эффективную платформу для реализации параллельных программ имитационного моделирования (симуляторов).

Основными элементами программы с использованием библиотеки ADEVS при построении АОМ обычно являются:

✓ симулятор adevs::Simulator< X >;

✓ примитив агентов adevs::Atomic< X >;

✓ модель (контейнер агентов) adevs::Digraph < VALUE, PORT >.

При этом агент представляет собой следующий класс (рис. 2).

Соответствующие виртуальные функции определяют необходимые для симуляции параметры:

void delta_int() – реакция агента на микроправило;

void delta_ext(...) – микрореакция агента на макроправило;

void delta_conf() – макрореакция агента на макроправило;

```
template <class X> class Atomic: public Devs<X>
    public:
        virtual void delta_int() = 0;
        virtual void delta_ext(double e, const Bag<X>& xb) = 0;
        virtual void delta_conf(const Bag<X>& xb) = 0;
        virtual void output_func(Bag<X>& yb) = 0;
        virtual double ta() = 0;
        virtual void gc_output(Bag<X>& g) = 0;
```

Рис. 2. Представление агента в ADEVS

void output_func() – вычисление выходных воздействий агента;

double ta() – время следующего события для агента;

void gc_output(...) – данные для процедуры сборки мусора.

В силу перечисленных выше достоинств, суперкомпьютерную версию описываемой ниже программы было решено реализовывать на базе ADEVS . В рамках этой работы были разработаны MPI-версия симулятора ADEVS, а также система визуализации процесса счета на базе библиотеки Qt.

Далее даны краткие описания разработанной модели и последующего ее запуска на суперкомпьютере.

Исходная агент-ориентированная модель

Первый этап разработки описываемой ниже АОМ заключался в построении инструментария, успешно решающего задачу исследования на обычных, последовательных компьютерах, а также в настройке параметров модели. После ее успешной апробации с небольшим числом агентов (учитывая их сложность, персональный компьютер с хорошей производительностью способен проводить вычисления с удовлетворительной скоростью над числом агентов около 20 тыс.) было решено конвертировать ее для суперкомпьютера, что являлось вторым этапом разработки. Отметим, что на первом этапе был использован пакет AnyLogic, технические возможности которого позволили достаточно быстро отладить модель и настроить ее параметры.

Вернемся к описанию разработанной модели. Итак, в 2009 г. в ЦЭМИ РАН была разработана АОМ воспроизводства научного потенциала России на базе ГИС.

По своей сути ГИС – это системы, позволяющие создавать базы данных, сочетающие в себе графическое и атрибутивное представление разнородной информации, а также обеспечивающие возможность пространственного анализа данных и представление его результатов в наиболее привычной для пользователей форме (в виде графиков, диаграмм, таблиц, карт и так далее).

Общая схема построения АОМ на базе ГИС состоит из следующих трех основных этапов.

I. Прорисовка среды для функционирования агентов (например, карты страны).

II. Для каждого элемента карты задаются свойства и методы, инициализируемые перед запуском модели с помощью соответствующих запросов к базе данных ГИС.

III. Для каждого элемента карты создается определенное число экземпляров объектов типа "агент".

По указанной схеме была разработана АОМ распространения знаний. Ниже приведено ее концептуальное описание (в виде набора тезисов).

1. Жизненный цикл агента состоит из двух основных стадий (рождение и смерть) и промежуточных состояний, отслеживаемых на каждом шаге работы модели.

2. От момента рождения и до определенного возраста (по умолчанию 18 лет) агент не участвует в процессе производства ВВП.

3. В течение жизни агент может стать либо обычным работником, либо ученым, либо "прикладником". Прослойка ученых создает базис для формирования прослойки "прикладников".

4. Становление ученого. По достижении работоспособного возраста, агент с некоторой вероятностью может стать ученым. Если до 25 лет агент не становится ученым, то он не будет им никогда.

Ученые не участвуют в создании ВВП, но в то же время:

- производят знания, потребляемые "прикладниками", которые участвуют в процессе производства ВВП;
- формируют среду, которая оказывает влияние на число "прикладников".

5. Агент перестает быть ученым (или "прикладником") из-за низкой зарплаты (если заработная плата ученого (или "прикладника") заметно ниже, чем в социуме, то он уходит на работу в другие отрасли). Агент – бывший ученый (или "прикладник") может снова вернуться в науку (или на работу в инновационно-активные предприятия), если заработная плата в науке (или прикладной науке) станет выше, чем в среднем по социуму, и если время отрыва от научной деятельности не превышает некоторого порога (по умолчанию 5 лет).

6. Продолжительности жизни агента-ученого (и "прикладника") выше, чем у обычного человека (по умолчанию на 10 лет). Однако в модели фактор продолжительности жизни не принимается в расчет при выборе профессии.

7. С задаваемой вероятностью (рассчитанной на основе российской статистики) агенты могут иметь ребенка. При этом ребенок агента-ученого (или "прикладника") становится ученым (или "прикладником") с большей вероятностью.

8. В модели предусмотрен экзогенный параметр – средняя зарплата высокоразвитых стран мира. Если в моделируемом социуме средняя зарплата (как у ученых, так и у представителей других профессий) становится намного ниже, чем в других странах, то ученый (или "прикладник") выбывает из социума навсегда (переезд в другую страну).

С помощью разработанной модели можно рассчитать последствия:

– от увеличения заработной платы (всем типам работников);

– от организации инновационных центров;

– от дополнительных инвестиций в науку.

Спецификация агентов модели осуществлялась с учетом следующих параметров (рис. 3, см. вторую сторону обложки):

- возраст;
- продолжительность жизни;
- специализация родителей;
- место работы;
- регион проживания;
- доход.

Спецификация регионов (элементов ГИС) проводилась на основе следующих параметров:

- географические границы;
- число жителей;
- число работников (по типам);
- валовый региональный продукт (ВРП);
- ВРП на душу;
- объем инвестиций;
- объем инвестиций на душу;
- средняя заработная плата;
- средняя продолжительность жизни;
- показатель прироста населения и ряда других.

Для наполнения модели данными использовались статистические сборники: Регионы России; Наука России в цифрах; Индикаторы науки; Индикаторы образования. Кроме того, были использованы социологические базы данных RLMS, а также результаты, полученные с помощью вычислимой модели экономики знаний [3].

На рис. 4 (см. вторую сторону обложки) изображено рабочее окно разработанной АОМ (точки – агенты). Благодаря возможностям ГИС в процессе работы системы можно получать оперативную информацию о социально-экономическом положении всех регионов России, в том числе с использованием картографической информации, меняющейся в режиме реального времени в зависимости от значений эндогенных параметров.

Конвертация модели в суперкомпьютерную программу

Выше уже отмечалось, что существует ряд проблемных вопросов, связанных с использованием средств разработки АОМ для реализации программных средств имитационного моделирования на высокопроизводительных вычислительных кластерных установках. Так и у AnyLogic в связи со сложностью отделения вычислительной части комплекса от той, которая обеспечивает функции визуализации, а также за счет реализации его кода на языке высокого уровня Java производительность выполнения приложений существенно ниже, чем у ADEVS. Кроме того, очень проблематично (либо очень трудоемко) переработать генерируемый код в параллельно выполняемую программу.

Далее представлен алгоритм конвертации модели AnyLogic в суперкомпьютерную программу.

Трансляция модели

На рис. 5 (см. вторую сторону обложки) приведен фрагмент XML-кода (дерево, отображающее структуру модели), сгенерированного программой AnyLogic.

В процессе работы конвертера это дерево транслируется в код C++ программы, вычисляющей эту модель. Проход дерева совершается "в глубину", выделяя при этом перечисленные далее ключевые стадии и совмещая их с выполнением задачи трансляции.

1. Генерация основных параметров, включая поиск корня дерева и считывание параметров дочерних вершин, таких как имя модели, адрес сборки, тип модели, тип презентации.

2. Генерация классов, в том числе:

- построение списка классов;
- считывание основных параметров класса;
- считывание переменных;
- считывание параметров;
- считывание функций;
- генерация списка функций;
- считывание кода функций;
- преобразование кода функций: Java $\rightarrow$ C++;
- считывание используемых фигур и элементов управления;
- генерация кода инициализации фигур и элементов управления;
- генерация кода конструктора, деструктора, визуализатора;
- генерация структуры класса;
- генерация кода заголовочного и *source*-файлов.

3. Генерация симулятора, а именно поиск вершины, хранящей информацию о процессе симуляции (управляющие элементы, значения важных констант, элементы презентации и так далее).

4. Генерация общих файлов проекта, в том числе main.cpp, mainwindow.h, mainwindow.cpp и подобных им.

Импортирование входных данных

В качестве входных данных в модель загружаются данные геоинформационной составляющей исходной модели (карты России), содержащей всю необходимую информацию.

Генерация классов и преобразование кода функций

В ходе трансляции неоднократно возникает задача о преобразовании исходного кода функций из языка

Java в язык C++. Его можно представить в виде последовательных замен конструкций.

В Java нет столь явного, как в C++, различия между объектом и указателем на объект, поэтому структуры работы с ними не отличаются. В силу этого обстоятельства вводится список классов, в которых важно использовать операции с указателем на объект, а не с самим объектом. При этом отслеживаются все переменные этих классов, с последующей заменой в пределах данной функции при обращении к ним на соответствующие обращения к указателям.

В Java и конкретно в библиотеке AnyLogic есть некоторое число функций и классов, аналогов которым нет ни в C++, ни в библиотеке ADEVS. В этой связи были реализованы дополнительные библиотеки shapes.h, mdb-work.h, в которых и реализованы недостающие функции.

На этапе генерации основных параметров списка классов получается название основного класса и названия моделируемых классов-агентов. В код основного класса встраивается процедура добавления агента в зону видимости симулятора.

Статистика и визуализация временных срезов

В силу неинтерактивного режима запуска программы на суперкомпьютерных установках, сбор выходных данных и визуализация были разделены. Такой подход связан с неравномерностью нагрузки на такие установки в разное время суток, а также тем фактором, что монопольный доступ к ним невозможен. После пересчета модели, получившаяся на выходе информация может быть снова визуализирована, например, следующим образом (рис. 6, см. третью сторону обложки).

Доступность для расчетов

На момент проведения тестовых испытаний рассматриваемой модели были доступны три суперкомпьютера (см. таблицу), входящих в первую пятерку суперкомпьютерного рейтинга Top50 среди стран СНГ (http://top50.supercomputergs.ru//?page=rating в редакции от 21.09.2010).

Из них для проведения имитационного моделирования использовались два суперкомпьютера – "Чебышев" и МВС-100К.

Сравнение производительности

За счет применения суперкомпьютерных технологий и оптимизации программного кода удалось добиться очень высокой производительности.

Как уже отмечалось ранее, обычный ПК с хорошей производительностью способен проводить вычисления с удовлетворительной скоростью над числом агентов около 20 тыс. (поведение каждого из них задается приблизительно 20 функциями), и при этом среднее время пересчета одной единицы модельного времени (один год) составляет около минуты. При большем числе агентов (например, 100 тыс.) компьютер попросту "зависает".

В свою очередь, использование всего 200 процессоров суперкомпьютера и выполнение оптимизированного кода позволило увеличить число агентов до 100 млн, а число модельных лет до 50. При этом такой гигантский массив вычислений был выполнен за период времени, приблизительно равный 1 мин 30 с (в зависимости от типа используемых процессоров).

Отметим также еще одну интересную особенность. По результатам экспериментов с разработанной АОМ было выяснено, что масштабирование модели само по себе имеет определенное значение. Так, при запуске одной и той же версии модели на 50 модельных лет с одинаковыми параметрами (за исключением числа агентов – в первом случае 100 млн агентов, а во втором – 100 тыс. агентов) были получены результаты (а именно масштабированное число агентов), отличающиеся приблизительно на 4,5 %.

В этой связи можно предположить, что, по всей видимости, в сложных динамических системах одни и те же параметры (рождаемость, продолжительность жизни и подобные им) могут приводить к различным результатам в зависимости от размера социума.

Дальнейшее развитие

В дальнейшем планируется обеспечить поддержку более широкого подмножества операторов/конструкций языка Java при определении исходной модели (Subset of Java → C++), а также автоматизировать запуск суперкомпьютерной программы из среды Eclipse (полностью автоматизировать процесс трансляции и распараллеливании программного кода).

Заключение

Несмотря на то что подобная работа может быть проделана практически для любой начальной реализации, в перспективе нужны системы, которые по возможности ликвидировали бы этот непростой этап, который сегодня затрудняет широкое внедрение суперкомпьютерного моделирования.

Доступные для исследовательской группы суперкомпьютеры

Позиция в Top50	Суперкомпьютеры	Узлы	CPU	Ядра	RAM/узел, ГБайт	TFlops
1	"Ломоносов" (МГУ)	4 420	8 840	35 360	12	414
3	МВС-100К (МСЦ РАН)	1 278	2 556	10 224	8	123
4	"Чебышев" (МГУ)	633	1 250	5 000	8	60

Тот факт, что в исходном представлении модели значительную роль играет код на языке программирования, отражает фундаментальное свойство агент-ориентированного подхода "модель – это программа".

Декларативный подход (набор правил вместо обычной программы) может сильно упростить задачу определения поведения системы.

Проиллюстрируем изложенные выше соображения (рис. 7, см. третью сторону обложки). При возникновении новой задачи мы получаем для нее некоторые исходные данные, а в свою очередь аналитики определяют, к какому классу задач она относится в целях подбора соответствующей спецификации для построения модели. Для этого используется наиболее адекватный DSL (*Domain Specific Language*), который позволяет описывать модели (в качестве IDE, к примеру, может быть Eclipse). Для этого языка, как правило, уже имеется ряд наработок на уже прошедших апробацию задачах этого класса, которые содержат в себе уже адаптированные, учитывающие специфику этих задач библиотеки. В своей совокупности все это образует метаязык, который затем реализуется в некоторой среде разработки и именно в ней, с точки зрения прикладников, в итоге и решается задача. После тестовых прогонов готовой программы на локальной машине следует финальная стадия – запуск на суперкомпьютере.

Общая последовательность действий в процессе разработки будет при этом следующая (рис. 8, см. третью сторону обложки).

1. Выбор языка описания (ЯО) модели (метамодели и ЯО моделей). Первый шаг контролируется специалистами формализации моделей и собственно программистами, которые выбирают язык, наиболее подходящий для описания данной задачи.

2. Описание модели на выбранном языке (этот шаг контролируется заказчиком).

3. Поиск и выбор необходимых шаблонов программирования, по сути, – выбор удобной среды разработки с наиболее подходящим инструментарием.

4. Подготовка алгоритмов распараллеливания и выбор аппаратной части.

5. Выбор библиотек, других программных составляющих и непосредственно программирование.

На четвертом и пятом шагах происходит перевод модели в низкоуровневые языки, подходящие для распараллеливания и, собственно, проводится запуск программы, настройка средств отображения результатов и представление их заказчику (после чего возможны дополнительные итерации).

СПИСОК ЛИТЕРАТУРЫ

1. **Борщев А.В.** Практическое агентное моделирование и его место в арсенале аналитика // Exponenta PRO. 2004. № 3–4 (7–8). С. 38–47.

2. **Карпов Ю.** Имитационное моделирование систем. Введение в моделирование с AnyLogic 5. СПб.: БХВ-Петербург, 2006.

3. **Макаров В.Л., Бахтизин А.Р., Бахтизина Н.В.** Вычислимая модель экономики знаний // Экономика и математические методы. 2009. № 1.

4. **Паринов С.И.** Новые возможности имитационного моделирования социально-экономических систем // Искусственные сообщества. 2007. № 3–4. С. 26–62.

5. **Ambrosiano N.** Avian flu modeled on supercomputer. Los Alamos National Laboratory NewsLetter. 2006. Vol. 7. No. 8.

6. **Axelrod R.** The Complexity of Cooperation: Agent-Based Models of Competition and Collaboration. Princeton: Princeton University Press, 1997.

7. **Bonabeau E.** Agent-based modeling: methods and techniques for simulating human systems. // Proc. National Academy of Sciences. 2002. 99(3): 7280–7287.

8. **Deissenberg Christophe, Sander van der Hoog, Dawid Herbert.** EURACE: A Massively Parallel Agent-Based Model of the European Economy / Document de Travail. 2008. 39. 24 Juin.

9. **Roberts D.J., Simoni D.A., Eubank S.** A National Scale Microsimulation of Disease Outbreaks. RTI International. Research Triangle Park, NC. Virginia Bioinformatics Institute. Blacksburg, VA, 2007.

10. **Epstein J.M., Axtell R.L.** Growing Artificial Societies: Social Science from the Bottom Up Ch. V., MIT Press, 1996.

11. **Epstein J.M.** Remarks on the foundations of agent-based generative social science. Handbook on Computational Economics, Vol. II/ K. Judd and L. Tesfatsion, eds. North Holland Press, 2005.

12. **Epstein J.M.** Modelling to contain pandemics // Nature 460, 687, 2009. 6 August.

13. **Gardner M.** Mathematical Games // Scientific American. 1970. October. P. 120–123.

14. **John von Neumann** Theory of Self-Reproducing Automata. University of Illinois Press. Urbana, 1966.

15. **Perumalla K. and Aaby B.** Data Parallel Execution Challenges and Runtime Performance of Agent Simulations on GPUs. // Proc. of Spring Computer Simulation Conf. Ottawa. Canada, 2008. April.

16. **Bisset K.R., Jiangzhuo Chen, Xizhou Feng, Anil Kumar V.S., Marathe M.V.** EpiFast: A Fast Algorithm for Large Scale Realistic Epidemic Simulations on Distributed Memory Systems ICS'09. Yorktown Heights. New York. USA. 2009. June 8–12.

17. **Makarov V.L., Zhitkov V.A., Bakhtizin A.R.** Moscow Traffic Jam Is Under Attack of an Intelligent Agent-based Model // Proc. of Conf. (edited by Bruce Edmonds and Nigel Gilbert). The 6th Conf. of the European Social Simulation Association. University of Surrey. Guildford. United Kingdom. 2009. 14–18 September.

18. **Parker J.** A Flexible, Large-Scale, Distributed Agent Based Epidemic Model. Center on Social and Economic Dynamics, Working Paper. 2007. No. 52.

19. **Tesfatsion L.** Agent-Based Computational Economics: Modelling Economies as Complex Adaptive Systems. URL: http://www.econ.iastate.edu/tesfatsi.

20. **Lynar T.M., Herbert R.D., Chivers W.J.** Implementing an agent based auction model on a cluster of reused workstations // International Journal of Computer Applications in Technology. 2009. Vol. 34. Issue 4.

21. **Ulam S.** Sets, Numbers and Universes. Cambridge. Massachusetts, 1974.