ТЕХНОЛОГИЯ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ БОЛЬШИХ СИСТЕМ

А.И. Миков, Е.Б. Замятина

ВВЕДЕНИЕ

Большие системы состоят из тысяч, десятков тысяч элементов и еще большего количества связей между ними. Такие системы характеризуются неоднородностью элементов (большим количеством различных типов элементов) и неоднородностью связей. Это приводит к тому, что классические математические методы становятся практически непригодными для описания и анализа больших систем.

Примером большой системы может быть современный процессор или глобальная компьютерная сеть. Несмотря на то, что отдельные элементы или связи прекрасно описываются моделями дискретной математики или теории массового обслуживания, о системе в целом этого сказать нельзя. Естественной альтернативой является использование имитационного моделирования, позволяющего соединить между собой разнородные математические модели элементов.

Имитационное моделирование успешно используется для анализа систем различного масштаба. Но при исследовании больших систем возникают специфические проблемы, требующие особой технологии имитационного моделирования. Составными частями этой технологии являются использование параллельного или распределенного моделирования, балансировка нагрузки, обеспечение безопасности процесса моделирования. Ниже мы рассмотрим основные компоненты предлагаемой технологии.

В докладе представлена распределенная система имитации Triad.Net с удаленным доступом. Доступ к имитационной модели осуществляется через специально разработанный портал «Имитационное моделирование».

Распределенную имитационную модель представляют обычно совокупностью логических процессов (LP_i, $i = 1, n$). Логические процессы обмениваются друг с другом сообщениями. Эти сообщения и синхронизируют распределенный процесс имитации. Один или несколько логических процессов размещаются на вычислительном узле компьютерной сети (или многопроцессорной ЭВМ).

Распределенная система имитации обеспечивает большую надежность имитационного эксперимента, поскольку при выходе из строя одного или нескольких вычислительных узлов, объекты имитационной модели, расположенные на них, переносятся на работающие узлы. Для широкого распространения систем распределенного моделирования требуются (но не всегда обеспечиваются [1] существующими системами):

- дружественный графический интерфейс;
- удаленный доступ через интерфейс для работы нескольких географически удаленных друг от друга исследователей;
- визуализация.

Дружественный интерфейс и удаленный доступ позволяют исследователям, которые находятся на удалении друг от друга выполнять совместные проекты, проводить исследования с одной и той же имитационной моделью, совместно разрабатывать и усовершенствовать эту модель. Кроме того, удаленные пользователи получают доступ к мощным вычислительным ресурсам.

ИМИТАЦИОННАЯ МОДЕЛЬ В TRIAD.NET

Имитационная модель в Triad [2,3] представляет собой совокупность объектов, которые действуют по определённым сценариям и обмениваются информацией друг с другом и может быть представлена тройкой: $\mu = \{Str, Rout, Mes\}$ –слоями структур, рутин и сообщений.

Слой структур (*Str*) предназначен для описания моделируемых объектов и связей между ними, слой рутин (*Rout*) представляет собой набор алгоритмов для описания поведения моделируемых объектов, а слой сообщений (*Mes*) даёт возможность описывать сообщения сложной структуры. Моделируемые объекты часто имеют иерархическую структуру. Имитационная модель в Triad также является иерархической. Каждый из уровней можно описать как граф с полюсами $P = \{U, V, W\}$, где V – множество вершин графа, каждая вершина представляет собой моделируемый объект, который находится на конкретном уровне иерархии. W – набор дуг, связывающих вершины графа (моделируемые объекты). U – набор внешних полюсов. Внутренние полюсы используют для передачи сообщений на одном уровне иерархии. Их разделяют на входные $In(V)$ и выходные $Out(V)$. Набор внешних полюсов служит для передачи информации объектам, находящимся на различных (смежных) уровнях иерархии.

Для описания структур в Triad используют конструкцию *structure* <наименование структуры> *def* (<список настроечных параметров>)(<список входных и выходных параметров>) <описание переменных><последовательность операторов>) *endstr*. В языке Triad введена переменная типа «модель». Особенностью СИМ Triad является тот факт, что при определении модели пользователь либо строит её с

нуля, добавляя или удаляя вершины, добавляя или удаляя дуги, ребра, полюса, объединяя графы, или извлекает их из базы данных. При построении модели можно воспользоваться графовыми константами, соответствующими основным топологиям вычислительных систем (цепь, звезда, дерево, полный граф, решётка и т.д.).

Рутина представлена множествами внутренних событий *E*, состояний *Q*, моментов времени *T*. Каждое состояние определяется набором значений локальных переменных (множество Var) каждой конкретной рутины. Система имитации Triad.Net является параллельной дискретно-событийной системой имитации (PDES – Parallel Discrete Event Simulation), в которой события планируют друг друга. Множество событий может быть представлено в виде графа запланированных событий, каждая вершина которого $e_i \in E$. Наряду с внутренними событиями выделяют множество событий (*EM*), связанных с посылкой сообщений другим процессам. Для описания рутины (сценарий поведения объектов) используют конструкцию *routine* <*имя*>(<настроечные параметры>)(<входные и выходные формальные параметры>) initial <последовательность операторов> *endi event* <последовательность операторов> *ende event* <имя события><последовательность операторов>…*endrou*t. При описании рутины также можно воспользоваться операциями над рутинами (добавление или удаление события и т.д.).

Как уже было отмечено ранее, слой сообщений используют для описания данных сложной структуры. Для слоя сообщений также определены операции (добавление, удаление типов, селекторов и т.д.). В СИМ Triad применяются также операции над моделями в целом: совмещение одного слоя с другим (если они были описаны отдельно), операции расшифровки вершин структурами нижнего уровня (для создания иерархических моделей), подключения рутин к вершинам. Таким образом, структура имитационной модели, логика поведения могут быть изменены динамически во время имитационного прогона.

АЛГОРИТМ ИССЛЕДОВАНИЯ ИМИТАЦИОННОЙ МОДЕЛИ

Для сбора статистических данных о ходе моделирования, для анализа и представления результатов имитационного эксперимента в Triad используют специальные средства – информационные процедуры и условия моделирования. Информационные процедуры и условия моделирования реализуют алгоритм исследования. *Алгоритм исследования отделён от модели.* Пользователю предоставлена возможность изменить алгоритм исследования в ходе моделирования, при этом модель остаётся неизменной, нет необходимости вносить в неё какие-либо изменения, чтобы указать алгоритму исследования те элементы модели, за изменением которых надо вести наблюдение (как это делается, например, в GPSS и др. языках имитационного моделирования). Информационную процедуру описывают следующим образом: *information procedure*<имя>(<настроечные параметры>)(<входные и выходные формальные параметры>) *initial* <последовательность операторов> *endi* <последовательность операторов> *processing* <последовательность операторов>…*endinf*. Информационные процедуры следят за изменением таких элементов модели, как локальные переменные рутины, события и сообщения, появившиеся на входном или выходном полюсе рутины. Часть processing информационных процедур предназначена для получения интегральных характеристик модели (среднее значение переменной, дисперсия и т.д.). В условиях моделирования указывают перечень информационных процедур, которые ведут наблюдение за моделью, условия завершения моделирования, алгоритм обработки результатов нескольких информационных процедур и т.д. Подсистема визуализации представляет результаты информационных процедур.

АРХИТЕКТУРА РАСПРЕДЕЛЕННОЙ СИСТЕМЫ ИМИТАЦИИ TRIAD.NET

Распределенная система имитации Triad.Net включает следующие компоненты: компилятор, ядро, графический редактор, подсистему синхронизации распределенных объектов модели, подсистему балансировки [4,5], подсистему организации удаленного доступа [6], подсистему защиты от внешних и внутренних угроз [7].

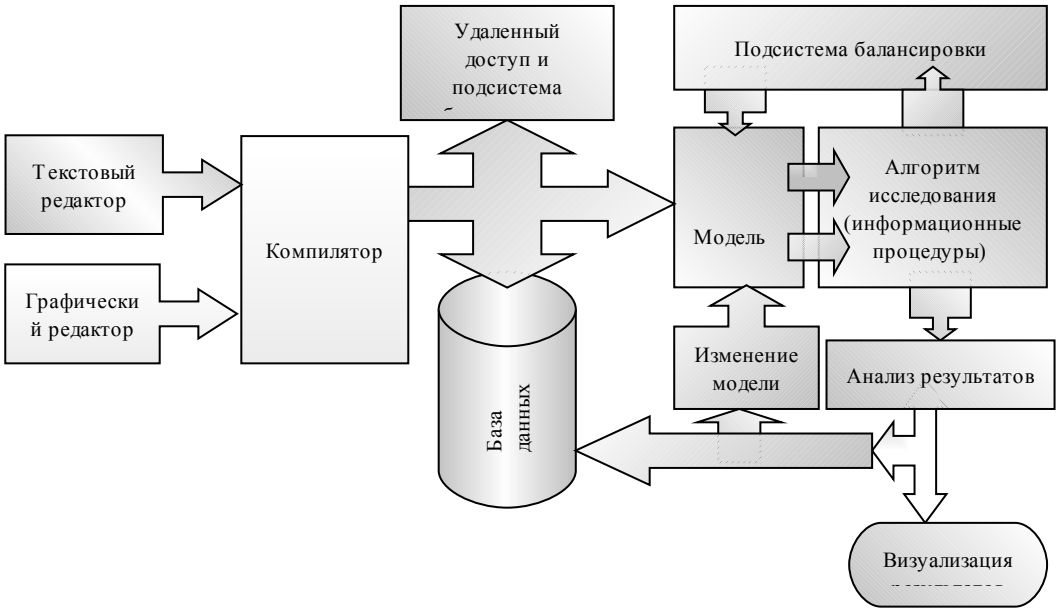

Рис. 1. Архитектура распределенной системы имитации

Перечисленные компоненты представлены на рис.1. Имитационная модель, написанная на алгоритмическом языке Triad, или сформированная исследователем с помощью графического редактора, преобразуется компилятором во внутреннее представление. Наряду с *моделью* формируется и *алгоритм исследования*, который наблюдает за изменениями в элементах модели, накапливает статистические данные, вычисляет интегральные характеристики модели, подготавливает данные для *визуализации*, определяет, следует ли завершить сеанс моделирования. Результаты имитационного моделирования и имитационная модель помещаются в базу данных.

Модель может быть сформулирована на языке Triad и удаленным пользователем с помощью Web сервисов, предоставляемых *подсистемой удаленного доступа*. Для взаимодействия удаленного пользователя с моделью используют механизм информационных процедур.

Имитационный прогон предваряется распределением объектов имитационной модели по вычислительным узлам компьютерной сети или многопроцессорной ЭВМ. Предварительное распределение может быть выполнено либо автоматически (статическая балансировка подсистемы балансировки в зависимости от структурных характеристик имитационной модели), либо с помощью языковых средств Triad.Net [5].

Во время имитационного прогона также необходимо следить за равномерным распределением нагрузки на вычислительных узлах, поскольку дисбаланс сводит выигрыш от использования ресурсов нескольких вычислительных узлов для проведения имитационных прогонов к нулю. За равномерным распределением нагрузки отвечает *подсистема балансировки*.

Во время имитационного прогона объекты имитационной модели должны быть синхронизированы.

ОСОБЕННОСТИ СИНХРОНИЗАЦИИ РАСПРЕДЕЛЕННЫХ ОБЪЕКТОВ ИМИТАЦИОННЫХ МОДЕЛЕЙ

Синхронизация объектов распределенных имитационных моделей выполняется с помощью специальных алгоритмов: консервативного и оптимистического. Консервативный алгоритм придерживается безопасной политики продвижения времени (время продвигается от события к событию, события сохраняют порядок причинности). Согласно этому классу алгоритмов рутина объекта модели выполняется только в том случае, если соблюдены все условия для её безопасного выполнения, т.е. существует уверенность в том, что объект не получит сообщения em_j с временем $t_j < t_i$ (t_i – время уже выполненного события рутины объекта). Оптимистический алгоритм позволяет выполнять рутины до тех пор, пока не произойдет ошибка, то есть пока рутиной не будет получено сообщение em_j с меньшим временем, чем время выполненных событий e_i ($t_j < t_i$). В этом случае реализуется протокол, который выполняет откат и исправляет ошибку. В случае применения консервативного и оптимистического алгоритмов в Triad.Net существуют механизмы, которые «расширяют горизонт времени», т.е. определяют «безопасные» события. В случае оптимистических алгоритмов сокращается количество откатов, в случае консервативных – происходит более «оптимистичное» продвижение рутины. «Безопасные» события определяются в Triad.Net в результате предварительного анализа топологических характеристик модели, исследования графа последовательности выполнения событий или в результате использования правил, которые задает пользователь на основании знаний поведения конкретной имитационной модели.

ПОДСИСТЕМА БАЛАНСИРОВКИ TRIAD.NET

При проведении распределенного имитационного эксперимента возможно возникновение дисбаланса нагрузки на вычислительных узлах (гетерогенность вычислительной системы, гетерогенность имитационной модели). При возникновении дисбаланса необходимо восстановить равномерное распределение вычислительной нагрузки на узлы компьютерной сети. Балансировку необходимо проводить не только до имитационного эксперимента, но и во время его проведения. В Triad.Net используют управляемую динамическую балансировку. Алгоритм балансировки учитывает не только топологические особенности вычислительной системы, на которой проводится имитационный эксперимент, не только структурные характеристики имитационной модели, но и особенности поведения конкретной модели. Кроме того, Triad.Net располагает лингвистическими и программными средствами для описания оригинального алгоритма балансировки, наилучшим образом соответствующего особенностям вычислительной среды и конкретной имитационной модели.

ОРГАНИЗАЦИЯ УДАЛЕННОГО ДОСТУПА

Удалённый доступ в Triad.Net поддерживается соответствующими инструментальными средствами, включающими следующие модули:

IMPortal – Интернет-портал, основанный на метаданных, который может использоваться отдельно от всего остального проекта как самостоятельное приложение. Изменяя метаданные, можно настроить структуру данных и наполнить портал содержимым из любой другой области, в том числе и не связанной с имитационным моделированием. В этом смысле IMPortal является каркасом Интернет-портала, а каким содержимым он будет наполнен, зависит от администратора портала.

TriadCore – представляет собой библиотеку типов, содержащую описание базовых структур, используемых в имитационной модели, таких как: BaseObjectClass – класс, описывающий структуру базового класса объектов; BaseObject – класс, описывающий структуру базового объекта, – от него наследуются все создаваемые в систему объекты; BaseSpy – класс, описывающий структуру базовой информационной процедуры (от этого класса наследуются все создаваемые в системе информационные процедуры); ModelRunner – класс, который выполняет построенную модель; Message, Event и другие менее значимые базовые классы, функциональное назначение которых понятно из названия (сообщение, событие).

TriadEditor – модуль, предоставляющий графический пользовательский интерфейс для редактирования моделей. TriadEditor является пользовательским элементом управления ОС Windows и, следовательно, может встраиваться в другие приложения ОС Windows. Модель, являющуюся результатом действий пользователя, можно получить в виде xml-документа, обратившись к свойству этого компонента. TriadEditor используется в модуле IMPortal для предоставления пользователям системы возможности совместной удаленной работы над моделями.

TriadClient – приложение ОС Windows, которое можно применять при однопользовательской работе с моделью или когда нет доступа к Интернет. Это приложение использует модули TriadEditor для редактирования и TriadCore для выполнения моделей.

TriadService –сервис ОС Windows, используемый для выполнения моделей.

Как видно из структуры Triad.NET, пользователь может работать с системой как удаленно, так и локально. Отдельные компоненты модели могут быть выполнены на различных серверах. Количество серверов для выполнения имитационных моделей не ограничено и зависит от загрузки системы.

Для создания, редактирования и запуска имитационных моделей используется компонент TriadEditor. Этот компонент может встраиваться как в Web-страницу, так и в обычное приложение Windows.

ОРГАНИЗАЦИЯ ПОДСИСТЕМЫ БЕЗОПАСНОСТИ TRIAD.NET

Поскольку система имитационного моделирования Triad предоставляет пользователям возможность удалённого доступа к модели, то возникает необходимость в защите её от внешних атак злоумышленников.

Кроме удаленного режима работы Triad предоставляет сервисы для локальной работы с имитационной моделью. Однако компоненты модели распределены по отдельным компьютерам вычислительной системы и обмениваются сообщениями во время выполнения. Следовательно, злоумышленник может вмешаться в работу СИМ, заменив нужное сообщение, и результаты проведения имитационного эксперимента в этом случае станут недостоверными. Это может повлечь за собой принятие неверных решений. Для устранения этой опасности необходимо, чтобы система обнаружения вторжений отвечала не только за защиту компонента IMPortal, но и проводила анализ внутренних сообщений, которыми обмениваются компоненты системы Triad.NET.

Итак, ко всем перечисленным модулям системы Triad.NET, добавляется еще один модуль TriadSecurity, который представляет собой систему обнаружения вторжений в объектно-ориентированную распределенную систему имитации Triad.NET. При разработке подсистемы безопасности выбран мультиагентный подход, обеспечивающий: а) эффективную реализацию задач, связанных с работой в компьютерных сетях (поиск, распределенная обработка информации); б) гибкую настройку системы, возможность в рамках одной системы реализовать разные подходы к решению задачи, поручив их реализацию разным

агентам; в) масштабируемость за счёт добавления или удаления агентов; г) увеличение функциональности за счет введения новых агентов; д) дополнительную степень защиты от злоумышленников за счет децентрализованного управления агентами.

В модуле TriadSecurity выделяют 2 уровня: а) внешний уровень модуля защиты системы Triad.NET (TriadOutSecurity); б) внутренний (TriadInSecurity). Внешний уровень модуля защиты находится на сервере для публикации Web-приложений, взаимодействует с модулем IMPortal и предназначен для защиты системы от несанкционированного вмешательства через удаленный доступ. Внутренний уровень находится на сервере БД и предназначен для защиты системы от попыток взлома изнутри, то есть от сообщений злоумышленника, направленных на разрушение таких компонентов системы, как TriadCore и TriadService.

При реализации системы защиты были выделены следующие классы агентов: а) агент реагирования; б) агент взаимодействия с пользователем; в) агенты выявления внешних атак; в) агенты выявления внутренних атак; г) агент взаимодействия с внешним уровнем защиты; д) агенты-датчики внешнего уровня; е) агенты-датчики внутреннего уровня.

Главную роль играет внешний уровень модуля защиты. Это объясняется тем, что вероятность проведения атак через удаленный доступ выше, чем вероятность проведения атак изнутри системы, так как чтобы преодолеть внутренний уровень модуля защиты, необходимо сначала преодолеть внешний. В этом модуле хранится база сигнатур атак. Связывает эти два уровня агент взаимодействия с внешним уровнем, который информирует внешний уровень обо всем, что происходит на внутреннем уровне. Для взаимодействия агентов используют доски объявлений (на внешнем и внутреннем уровнях). Информация, собранная агентами-датчиками внутреннего уровня модуля защиты, попадает на обе доски объявлений. На внутреннем уровне эту информацию использует агент выявления внутренних атак, а на внешнем уровне информация агентов датчиков внутреннего уровня помечается специальным флагом. Это необходимо для того, чтобы ею не мог воспользоваться агент выявления внешних атак, но она представляет интерес для агента реагирования, который определяет реакцию системы на текущую ситуацию.

Если агент выявления внутренних атак, обнаруживает проблемы на внутреннем уровне модуля защиты, то информация об этом сразу же появится и на внешнем уровне модуля защиты, опять же с помощью агента взаимодействия с внешним уровнем.

Агенты-датчики выполняют постоянный мониторинг системы как на внешнем (анализ сетевого трафика, системных журналов), так и на внутреннем уровнях (здесь роль агентов-датчиков играют информационные процедуры). Выделяют два типа агентов-датчиков внешнего уровня: агенты-датчики, собирающие информацию из сетевого трафика; агенты-датчики, собирающие информацию из системных журналов. Информация от агентов-датчиков поступает на доску объявлений (нормализованная база данных), причем каждый агент «знает», в какую таблицу он вывешивает результаты своей работы, а из какой таблицы он берет данные для начала работы. На доске объявлений хранится такая информация, как длина заголовка пакета, тип сервиса, IP-адрес отправителя; IP-адрес назначения и т.д. Для перехвата сетевого трафика используют «слушающий сокет». Доступ к данным реализован при помощи технологии ADO.NET. Итак, агенты-датчики имеют очень простую структуру, а это объясняется тем, что они являются реактивными агентами, не располагает данными о внешней среде, не обладает мотивацией.

Информация, «вывешенная» на доске объявлений агентами-датчиками, используется агентом выявления атак внешнего уровня (он сопоставляет информацию с базой сигнатур атак и принимает решение о наличии атаки на систему) и агентом взаимодействия с пользователем (этот агент предоставляет информацию пользователю, предварительно преобразовав ее). Агент взаимодействия с пользователем – это единственный агент системы, который ведет диалог с пользователем (чаще всего – это системный администратор), тогда как остальные агенты даже «не подозревают» о существовании пользователя. В сферу его деятельности попадают все таблицы доски объявлений без исключения (другие агенты используют конкретные таблицы), информацию из которых он предоставляет пользователю. Агент взаимодействия реализован таким образом, что пользователь имеет возможность настройки представления информации.

ЗАКЛЮЧЕНИЕ

В докладе представлена архитектура распределенной системы имитации Triad.Net, ориентированной на моделирование больших систем. Ускорение процессов моделирования достигается за счет использования большого количества компьютеров, соединенных каналами связи. Система оснащена инструментальными средствами для оптимизации времени проведения имитационного эксперимента и удаленного доступа. Удаленный доступ дает возможность географически удаленным друг от друга исследователям работать в одном проекте.

Авторы благодарят РФФИ за финансовую поддержку проектов 08-07-90005Бел_а и 08-07-90006Бел_а.

ЛИТЕРАТУРА:

1. Perumalla K., and Fujimoto R. Interactive Parallel Simulations with the JANE Framework//Workshop on Parallel and Distributed Simulation(http://www.cc.gatech.edu/ computing/ pads/papers.html)
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.I. Simulation and Design of Hardware and Software with Triad// Proc.2nd Intl.Conf. on Electronic Hardware Description Languages, Las Vegas, USA, 1995. pp. 15-20
4. Миков А.И., Замятина Е.Б., Осмехин К.А. Динамическое распределение объектов имитационной модели, основанное на знаниях.//Proceedings of XIII International Conference “Knowedge-Dialogue-Solution” (KDS), ITHEA, Sofia, 2007, Vol. 2, pp.618-624
5. Миков А.И., Замятина Е.Б. Балансировка распределенной нагрузки в системе моделирования Triad.// Труды Всероссийской научной конференции «Научный сервис в сети Интернет» -М.: Изд-во МГУ, 2007, с.104-108
6. Mikov A., Zamyatina E., Firsov A. Software for Remote Parallel Simulation.// International Journal «Information Theories & Applications», Vol.14, № 4, 2007, pp. 389-395
7. Миков А.И., Замятина Е.Б., Панов М.П. Мультиагентная система защиты имитационной модели с удаленным доступом.//Труды Международных научно-технических конференций «Интеллектуальные системы»(AIS’07) и «Интеллектуальные САПР» (CAD-2007),-М.: Физматлит, 2007, Т1. сс.323-330