Переносимая система имитационного моделирования для многопроцессорных вычислительных систем

В.В. Окольнишников, С.В. Рудометов

Рассматривается система распределенного имитационного моделирования Мера, реализованная для МВС 1000/128 в Сибирском Суперкомпьютерном центре (ИВМиМГ СО РАН, Новосибирск). Система Мера предназначена для решения сложных и большого масштаба задач математического моделирования. Система Мера является переносимой. Система Мера перенесена на Новосибирский кластерный суперкомпьютер НКС-160. Система Мера может быть перенесена на другие МВС или распределенные вычислительные системы, использующие библиотеку MPI.

1. Введение

Имитационное моделирование, по-прежнему, остается одним из наиболее распространенных и эффективных инструментов исследования сложных систем и процессов. При этом наблюдаются две тенденции: рост потребности в вычислительных ресурсах для исполнения имитационных моделей большого масштаба и рост предложения вычислительных ресурсов за счет использования многопроцессорных вычислительных систем (МВС).

О темпах роста вычислительных ресурсов МВС свидетельствуют данные седьмой редакции списка 50 самых мощных суперкомпьютеров СНГ [1]. Суммарная производительность суперкомпьютеров России и некоторых стран СНГ выросла за год более чем вдвое. Использование многопроцессорных вычислительных систем для решения задач имитационного моделирования сдерживается в настоящее время практическим отсутствием систем параллельного имитационного моделирования. Имеющиеся разработки в МГУ [2] и Пермском государственном университете [3] используются для специализированных приложений.

В связи с этим возникла задача разработки расширяемой, эффективной, переносимой системы параллельного моделирования. В статье рассматривается система имитационного моделирования *Мера* [4], реализованная для МВС 1000/128 (на базе процессоров *DEC Alpha* 21264), которая эксплуатируется в Сибирском Суперкомпьютерном центре (г. Новосибирск) при институте вычислительной математики и математической геофизики СО РАН (ИВМиМГ СО РАН).

2. Терминология

Исторически термин "параллельное моделирование" использовался для моделирования на компьютерах с *SIMD (Single Instruction Stream/Multiple Data Stream)* архитектурой, а термин "распределенное моделирование" относился к моделированию на компьютерах с *MIMD (Multiple Instruction Stream / Multiple Data Stream)* архитектурой.

Поскольку при параллельном и распределенном моделировании используется одна и та же техника, с появлением кластерных вычислительных систем и Grid-вычислений грань между этими двумя видами имитационного моделирования начала стираться. В настоящее время многие авторы, в том числе известный специалист по распределенному моделированию *R.M. Fujimoto* [5], используют термин "распределенное моделирование" (distributed simulation) для обозначения как параллельного, так и собственно распределенного моделирования. Название статьи "Система параллельного имитационного моделирования" выбрано для того, чтобы подчеркнуть, что система *Мера* реализована для кластерного суперкомпьютера. В следующих разделах будет рассказано, что систему *Мера* можно перенести и на другие многопроцессорные вычислительные системы, как параллельные, так и распределенные. Поэтому в дальнейшем будет использоваться термин "распределенное имитационное моделирование".

3. Проблемы распределенного имитационного моделирования.

Классическое процессно-ориентированное дискретно-событийное имитационное моделирование имеет более 40-летнюю историю, хорошо развитую теорию и методологию, большое количество систем моделирования и приложений [6].

В рамках этой парадигмы имитационная модель состоит из множества модельных процессов (аналогов объектов в объектно-ориентированном подходе), моделирующих какие-нибудь функции или элементы декомпозиции моделируемой системы. Исполнение каждого модельного процесса сводится к последовательному выполнению "активных фаз" (событий) процесса, имитирующих некоторые значимые с точки зрения разработчика модели изменения в моделируемой системе.

Каждое событие связано с некоторым значением условного модельного времени, имитирующего с некоторым масштабом и приближением реальное (физическое) время, в котором существует моделируемая система. Моменты модельного времени (метки времени), в которые должны выполняться все события всех модельных процессов, составляют дискретное множество точек на оси модельного времени.

Для того чтобы модель правильно воспроизводила поведение моделируемой системы, необходима организация выполнения событий в порядке не убывания их меток времени. Некоторые события в разных модельных процессах могут иметь одинаковые метки времени, хотя соответствующие изменения в моделируемой системе могут происходить в различные моменты физического времени, но разность между этими моментами не превышает точности моделирования. Порядок выполнения событий, имеющих одинаковые значения модельного времени, в этой статье не рассматривается и может быть произвольным.

Для организации правильного порядка выполнения событий требуется "управляющая программа", синхронизирующая выполнение модельных процессов. Эта управляющая программа содержит структуру – глобальные часы модельного времени (аналог физических часов). Управляющая программа активизирует некоторый модельный процесс для выполнения события, метка времени которого равна значению часов модельного времени. После окончания выполнения события текущий процесс приостанавливается, вычисляется новое значение часов модельного времени, активизируется процесс для выполнения события, метка времени которого равна новому значению часов модельного времени и т.д.

При такой организации выполнения модели события в модели выполняются последовательно, поэтому классическое имитационное моделирование (в отличие от распределенного) называется также последовательным имитационным моделированием. Концептуально модельные процессы выполняются параллельно в модельном времени, поэтому говорят, что процессы выполняются квазипараллельно.

Для преобразования последовательных программ математического моделирования в параллельные требуется изменение алгоритма и программы задачи. Имитационные модели с самого начала строятся как множество модельных процессов, исполняемых квазипараллельно в условном модельном времени. Для преобразования последовательных имитационных моделей в параллельные требуется изменить не столько алгоритмы процессов, сколько управляющую программу, планирующую выполнение активных фаз процессов.

Эффективному использованию параллелизма при выполнении имитационных моделей мешает наличие глобальных структур: глобальных данных для организации взаимодействия модельных процессов и системных глобальных часов модельного времени. Для устранения этих препятствий в распределенном имитационном моделировании принята концепция "логического процесса".

Логический процесс — это последовательная подмодель (или отдельный модельный процесс), исполняющаяся на собственном процессоре. Каждый логический процесс имеет собственные часы локального модельного времени и собственный экземпляр управляющей программы. Логические процессы взаимодействуют между собой исключительно с помощью посылки сообщений.

При параллельном исполнении логических процессов скорость изменения локального модельного времени в разных логических процессах может оказаться разной. Это может произойти, например, оттого, что число событий на одном отрезке модельного времени в этих

логических процессах разное, или выполнение событий требует разного количества процессорного времени. При этом может случиться ситуация, когда логический процесс получает сообщение от другого логического процесса, и локальное модельное время отправителя меньше, чем локальное модельное время получателя. Это означает, что у логического процесса получателя, как бы изменяется "прошлое". Такой "парадокс времени", часто описываемый писателями-фантастами, не может возникнуть при последовательном имитационном моделировании, а значит, приведет к неверному выполнению распределенного имитационного моделирования.

Для устранения таких "коллизий" на управляющие программы логических процессов возлагаются дополнительные (а по объему и сложности основные) функции по синхронизации локального модельного времени логических процессов. Используемые алгоритмы синхронизации основаны на посылке служебных (системных) сообщений. Для посылки служебных сообщений используется тот же механизм, что и для посылки пользовательских (запрограммированных разработчиком модели) сообщений. Алгоритмы синхронизации делятся на два основных класса: консервативные и оптимистические.

Консервативные алгоритмы предотвращают возникновение парадоксов времени путем задержки выполнения некоторых логических процессов. Оптимистические алгоритмы предполагают, что парадоксов времени не будет, что и отражено в их названии. В случае же возникновения парадокса времени оптимистические алгоритмы реализуют "откат" (rollback) логического процесса до значения модельного времени, в который ему было послано сообщение, вызвавшее парадокс времени.

Априорно нельзя сказать, что какой-то алгоритм консервативный или оптимистический работает быстрее. Все зависит от динамики исполнения модели. Для повышения эффективности распределенного имитационного моделирования имеются различные модификации базовых консервативного и оптимистического подходов. С более подробным описанием алгоритмов синхронизации можно познакомиться в [7, 8, 9].

Кроме проблемы синхронизации модельного времени в распределенном имитационном моделировании существуют проблемы корректного запуска и останова имитационной модели.

4. Стандарт *HLA*

По инициативе Министерства Обороны США в 2000 г. был принят стандарт *IEEE 1516*, более известный как *HLA* (*High Level Architecture*). Стандарт расширяет понятие имитационной модели до федерации, состоящей из набора разнородных логических процессов, называемых федератами. Разнородность заключается в различных способах продвижения локального модельного времени федератов (консервативного, оптимистического и др.). Использование при программировании федератов интерфейса взаимодействия с сервисами управления временем (*Time management services*), составляющих одну из частей среды исполнения *RTI* (*Runtime Infrastructure*), гарантирует отсутствие парадоксов времени при исполнении федерации [10].

Использованию *HLA* в России препятствует недоступность (*HLA* применяется главным образом для военных приложений) или слишком высокая цена (коммерческие разработки). Свободно распространяемое программное обеспечение *HLA* недостаточно отлажено и требует адаптации для использования на конкретных МВС.

Целью введения стандарта *HLA* является возможность переносимости (переиспользования) моделей, разработанных в рамках различных систем моделирования, число которых (за рубежом) исчисляется сотнями, для уменьшения времени и стоимости разработок новых моделей. Для России более актуальной задачей является переносимость систем моделирования, число которых исчисляется единицами, на новые МВС.

5. Структура системы *Мера*

Основными свойствами системы имитационного моделирования *Мера* являются: переносимость, эффективность, расширяемость. Эти свойства обеспечиваются за счет иерархической структуры программного обеспечения системы *Мера*, представленной на рис.1.

Рис. 1. Структура системы имитационного моделирования *Мера*

Каждый слой программного обеспечения системы распределенного имитационного моделирования *Мера* выполняет определенные функции. Каждый слой может послать команду вышестоящей по уровню иерархии машине. Формой посылки команды и получения ответа является вызов интерфейсной функции.

Ядро. Основными функциями ядра являются: задание структуры модели, передача пользовательских сообщений, задержка процессов на определенное модельное время или до получения сообщения во входной порт, инициализация и выполнение модели. Имеется возможность построения иерархической структуры модели и динамического изменения структуры модели во время выполнения.

Перечисленные выше функции определяют компактный, легко изучаемый программный пакет дискретного имитационного моделирования на *C++*. Этот пакет для практического использования расширен библиотеками общего пользования и библиотеками для специализированных приложений.

Последовательная имитационная машина является системой поддержки последовательного исполнения (*Run time system*) модельных процессов в локальном модельном времени.

Для исполнения модели на однопроцессорном компьютере требуется конфигурация системы *Мера* в составе ядра и последовательной имитационной машины. Для исполнения модели на многопроцессорной вычислительной системе требуется подключение дополнительных слоев программного обеспечения системы *Мера* – распределенной имитационной и коммуникационной машин.

Распределенная имитационная машина является системой поддержки исполнения модельных процессов в едином модельном времени, выполняемых на различных процессорах многопроцессорной или распределенной вычислительной системы.

В распределенной имитационной машине реализованы базовые классы для двух основных типов алгоритмов синхронизации модельного времени в распределенном имитационном моделировании — консервативный и оптимистический. На основе экспертной оценки разработчика модели и/или на основе тестового исполнения модели на ограниченном отрезке модельного времени со снятием статистики можно определить, какой из типов алгоритмов консервативный или оптимистический более эффективен для данной модели.

Коммуникационная машина обеспечивает обмен сообщениями, как пользовательскими, так и служебными, между параллельно исполняющимися модельными процессами. Передача сообщений в коммуникационной машине реализована на основе протокола посылки сообщений *MPI* (*Message passing interface*) [11].

6. Переносимость системы *Мера*

Использование *MPI* позволяет переносить систему распределенного имитационного моделирования *Мера* на другие вычислительные системы, где имеется реализация *MPI*.

Использование *MPI* не является ограничением для переносимости системы *Мера*. *MP*I – объемная и сложная библиотека, состоящая более чем из 125 функций. В системе *Мера* используются лишь 15 общеупотребительных функций *MPI*. Таким образом, для переносимости системы *Мера* может быть использована любая библиотека передачи сообщений, которая реализует эти 15 функций и имеет интерфейс, совпадающий с *MPI*.

Система команд (интерфейс) распределенной имитационной машины совпадает с интерфейсом *HLA* в части сервисов управления времени, которые синхронизируют продвижение локального модельного времени федератов при исполнении федерации.

При наличии библиотеки *HLA* выполнение большей части функций распределенной имитационной машины может заменить инфраструктура *RTI*, что делает систему распределенного имитационного моделирования *Мера* более надежной и переносимой, а модели, разработанные с ее помощью, способными к переиспользованию и взаимодействию с моделями, разработанными в рамках других систем имитационного моделирования, соответствующих стандарту *HLA*.

7. Семейство *Мера*

Особенностью системы имитационного моделирования *Мера* является наличие семейства совместимых по ядру входного языка реализаций для различных операционных систем. Семейство системы имитационного моделирования *Мера* представлено на рис. 2.

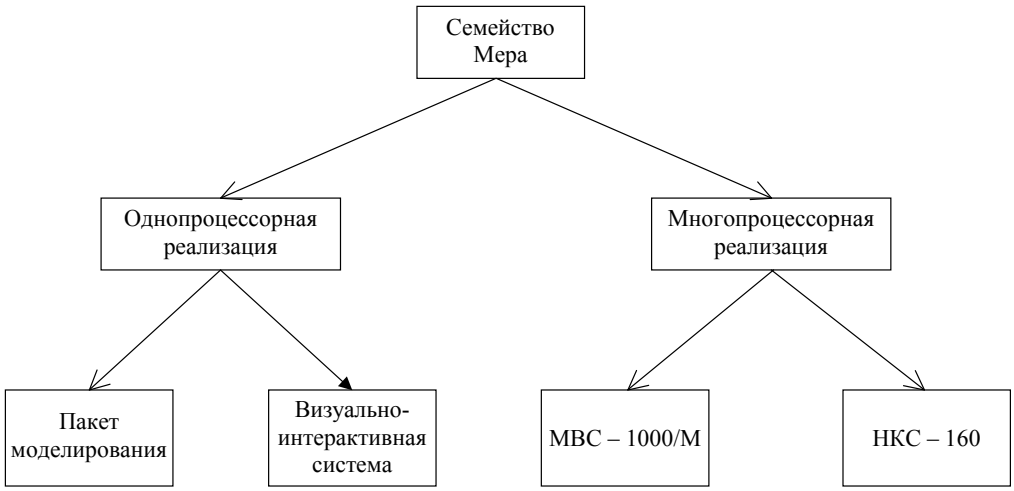

Рис. 2. Семейство системы имитационного моделирования *Мера*

Однопроцессорная реализация системы *Мера* под *Windows* на *C++* представляет собой пакет дискретного имитационного моделирования в объеме ядра системы, расширенного общими библиотеками. В среде *Java* на базе ядра системы *Мера* реализована специализированная визуально-интерактивная система моделирования транспортных систем [12]. Она позволяет строить из стандартных блоков транспортную сеть и визуализировать потоки транспортных средств по транспортной сети в соответствии с задаваемой дисциплиной обслуживания.

С использованием системы *Мера* под *Windows* был разработан имитационный стенд Автоматизированной системы управления технологическими процессами (АСУ ТП) Северомуйского тоннеля (Байкало-Амурская магистраль), включающий комплекс моделей для отладки и тестирования подсистемы управления АСУ ТП [13].

Основная реализация системы Меры выполнена для суперкомпьютера МВС–1000/М под операционной системой *Linux*. Для работы системы *Мера* требуется библиотека *MPI – MPICH2* версии 1.0.1.

С использованием свойства переносимости система *Мера* перенесена на Новосибирский кластерный суперкомпьютер НКС-160 (на базе процессоров *Intel Itanium* 2). Система *Мера* может быть перенесена и на другие многопроцессорные вычислительные системы.

Литература

1. Текущий рейтинг: [http://www.supercomputers.ru/?page=rating], 25.09.2007.
2. Смелянский Р.Л. Проблемы разработки и анализа функционирования встроенных систем реального времени // Труды Первой Всероссийской научной конференции "Методы и средства обработки информации". — М.: МГУ, 2003. — С. 57–73.
3. Миков А.И., Замятина Е.Б., Фатыхов А. Система оперирования распределенными имитационными моделями сетей телекоммуникаций // Труды Первой Всероссийской научной конференции "Методы и средства обработки информации". — М.: МГУ, 2003. — С. 437–442.
4. Окольнишников В.В. Разработка системы распределенного имитационного моделирования для различных операционных сред // Сборник докладов Второй Всероссийской научно-практической конференции по имитационному моделированию и его применению в науке и промышленности "Имитационное моделирование. Теория и практика" (ИММОД 2005). — Санкт-Петербург, 2005. — Т. 1. — С. 256–260.
5. Fujimoto R.M. Distributed Simulation Systems // Proceedings of the Winter Simulation Conference. — 2003. — P. 124–134.
6. Лоу А.М., Кельтон А.Д. Имитационное моделирование. — СПб.: Питер, 2004.
7. Ferscha **A.** Parallel and Distributed Simulation of Discrete Event Systems. Parallel and Distributed Computing Handbook. — McGraw-Hill, 1996. — P. 1003–1041.
8. Казаков Ю.П., Смелянский Р.Л. Об организации распределенного имитационного моделирования // Программирование. – 1994, № 2. — С. 45–63.
9. Окольнишников В.В. Представление времени в имитационном моделировании // Вычислительные технологии. — 2005, Т. 10, № 5. — С. 57–80.
10. Fujimoto R.M. Time Management in the High Level Architecture // Simulation. — 1998. — Vol. 71, № 6. — P. 388–400.
11. Корнеев В.Д. Параллельное программирование в MPI. — Новосибирск: Изд-во СО РАН, 2000.
12. Okol'nishnikov V.V., Rudometov S.V. Simulation of Complex Transportation Systems // Proceedings. of the Second IASTED International Multi-Conference Software Engineering (ACIT-SE). — Novosibirsk, 2005. — P. 60–64.
13. Окольнишников В.В. Использование имитационного моделирования при разработке Автоматизированной системы управления технологическими процессами Северомуйского тоннеля // Вычисл. технологии. — 2004, Т. 9, № 5. — С. 82–101.