МОНИТОРИНГ РАСПРЕДЕЛЕННОЙ ИМИТАЦИОННОЙ МОДЕЛИ ДЛЯ ПОДСИСТЕМЫ БАЛАНСИРОВКИ

А. И. Миков, Е. Б. Замятина (Пермь)

Метод имитационного моделирования находит широкое применение при исследовании сложных систем и, следовательно, часто требует большого количества вычислительных ресурсов, и разработки новых программных средств, алгоритмов, использования новых технологий для выполнения имитационного эксперимента с моделью, компоненты которой распределены по узлам вычислительной системы (многопроцессорной ЭВМ или сети компьютеров). Использование распределенного моделирования и алгоритмы, синхронизирующие совместную работу компонентов распределенной имитационной модели (консервативный и оптимистический алгоритмы), рассматриваются в большом количестве работ R. Fujimoto [1] и других зарубежных и отечественных авторов.

Однако при проведении распределенного имитационного алгоритма возникают проблемы, связанные с возникновением дисбаланса нагрузки вычислительных узлов. Возникновение дисбаланса объясняется:

гетерогенностью вычислительной системы (разная пропускная способность линий связей между вычислительными узлами, разная производительность вычислительных узлов);

гетерогенностью имитационной модели (разная частота обменов между компонентами имитационной модели, разная частота обращения к различным компонентам имитационной модели);

изменением загрузки вычислительного узла другими приложениями, которые также выполняются вычислительной системой.

Несбалансированное выполнение имитационного эксперимента может привести к тому, что ускорение, которое может быть достигнуто за счет распределения компонентов имитационной модели по различным вычислительным узлам и их параллельного выполнения, будет сведено к нулю. По этой причине целесообразно применять специальные методы: статическую и динамическую балансировки. Кратко рассмотрим эти методы.

Балансировка загрузки процессоров

Распределенная имитационная модель обычно представляет собой набор компонентов (или объектов, или логических процессов) [2]. Компоненты имитационной модели взаимодействуют друг с другом, обмениваясь сообщениями. Перед проведением вычислительного эксперимента над такой моделью ее компоненты распределяют по вычислительным узлам. При распределении компонентов важно знать, какую нагрузку будет оказывать компонент на вычислительный узел, какова производительность этого узла, какова частота обменов между компонентами, какова производительность линии связи, с помощью которой отдельные компоненты распределенной имитационной модели обмениваются друг с другом.

Для равномерного распределения нагрузки по узлам вычислительной системы применяют **статическую балансировку**, выполняемую до начала имитационного эксперимента. При распределении нагрузки по узлам используют дополнительную информацию. Эта дополнительная информация должна содержать структурные характеристики вычислительной системы, на узлах которой выполняется имитационный эксперимент (именно так организована статическая балансировка для распределенной версии системы имитационного моделирования Triad.Net). В других реализациях подси-

стемы балансировки (например, система моделирования SPEEDES) учитывают особенности выполнения предыдущих прогонов распределенной имитационной модели. В этом случае для лучшего распределения объектов модели по вычислительным узлам ВС целесообразно применить генетические алгоритмы, нейронные сети [3].

Однако статическая балансировка не решает проблемы равномерного распределения объектов имитационной модели по узлам ВС во время проведения имитационного эксперимента. Причины возникновения дисбаланса перечислены выше, можно еще добавить возможность выхода из строя ряда вычислительных узлов или изменения структуры имитационной модели.

Динамическая балансировка предполагает выявление дисбаланса во время выполнения имитационного эксперимента и перенос объектов имитационной модели с наиболее загруженных узлов на менее загруженные. При переносе объектов имитационный эксперимент не прерывается. Существует достаточное большое количество исследований, в которых предлагаются различные алгоритмы автоматической динамической балансировки [2, 3], однако все они не учитывают специфических особенностей конкретной имитационной модели. Авторы настоящей статьи предложили *управляемую* динамическую балансировку [5]. Перенос объектов с одного вычислительного узла на другой выполняется по правилам, в которых учитываются:

характеристики вычислительных узлов ВС, на которой выполняется эксперимент, а именно: загрузка процессоров объектами имитационной модели и сторонними программами, пропускная способность линий связи, объем памяти и т. д.;

характеристики выполнения имитационной модели, а именно; частота обменов между объектами, частота выполнения события и т. д.;

особенности распределения объектов имитационной модели по узлам ВС;

топология вычислительной системы.

Правила определены для различных классов задач. И поскольку только разработчик модели знает характерные особенности своей модели (например, он знает, что в определенный момент времени два объекта будут интенсивно взаимодействовать друг с другом, посылая сообщения), он может откорректировать правила с помощью специального редактора правил.

На кафедре математического обеспечения ВС ведутся работы по созданию распределенной системы имитации Triad.Net. Как уже упоминалось выше, распределенный имитационный эксперимент требует во время исполнения восстановления баланса нагрузки на вычислительных узлах. Рассмотрим более подробно архитектуру подсистемы управляемой балансировки Triad.Net.

Архитектура подсистемы балансировки

Подсистема балансировки в Triad.Net включает в себя следующие компоненты:

компонент мониторинга вычислительной системы, на которой выполняется распределенный имитационный эксперимент;

компонент мониторинга имитационной модели;

базу знаний, которая включает правила балансировки;

редактор правил;

визуализатор процесса балансировки;

модуль принятия решений;

базу данных, включающую информацию о модели и о вычислительной системе.

Именно на основании этой информации модуль принятия решений выбирает соответствующие правила (из базы знаний), в результате применения которых перераспределяются ресурсы вычислительной системы. Информация в базе данных является результатом работы компонентов мониторинга вычислительной системы и имитацион-

ной модели. Эта информация может быть использована для проведения статической балансировки.

Правила в базе знаний могут быть такими:

– если на *i*-м вычислительном узле находится больше двух активных объектов, а *j*-й узел свободен, то перенести один из объектов на *j*-й свободный узел;

– если два объекта (*Vi* и *Vj*) в сети интенсивно обмениваются данными и в сети есть незанятый узел *Vi*, то переместить их на этот узел, и т. д.

Правила представляют собой продукции вида «if …then…else…» и могут быть описаны на языке Triad.

Визуализатор процесса балансировки дает возможность наглядно представить, каким образом объекты имитационной модели отображены на вычислительный узлы, а по окончании эксперимента анимировать процесс балансировки. Работа визуализатора также связана с компонентами мониторинга.

Рассмотрим подробнее *компонент мониторинга* имитационной модели, особенности реализации программных средств, выполняющих сбор, обработку и накопление информации о модели, но прежде - особенности представления модели имитации в системе Triad.Net.

Описание имитационной модели

Описание имитационной модели в Triad состоит из трех слоёв: слоя структур (STR), слоя рутин (ROUT) и слоя сообщений (MES). Таким образом, модель в системе Triad можно определить как M={STR,ROUT,MES}.

Слой структур представляет собой совокупность объектов, взаимодействующих друг с другом посредством посылки сообщений. Каждый объект имеет полюса (входные P_{in} и выходные P_{out}), которые служат соответственно для приёма и передачи сообщений. Слой структур можно представить графом. В качестве вершин графа следует рассматривать отдельные объекты. Дуги графа определяют связи между объектами.

Объекты действуют по определённому алгоритму поведения, который описывают с помощью рутины (rout) (routine - в переводе с англ. «подпрограмма», алгоритм). Рутина представляет собой последовательность событий e_i ,планирующих друг друга ($e_i \in E$, $i=1…n$, E – множество событий; множество событий рутины является частично упорядоченным в модельном времени). Выполнение события сопровождается изменением состояния q_k объекта. Состояние объекта определяется значениями переменных var_j рутины ($var_j \in Var$, $j=1…m$, Var – множество переменных рутины). Таким образом, система имитации является событийно-ориентированной. Рутина так же, как и объект, имеет входные (Pr_{in} и выходные Pr_{out}) полюса. Входные полюса служат соответственно для приёма сообщений, выходные полюса – для их передачи. В множестве событий рутины выделено входное событие e_{in}. Все входные полюса рутины обрабатываются входным событием. Обработка выходных полюсов осуществляется остальными событиями рутины. Для передачи сообщения служит специальный оператор *out* (*out* <сообщение> *through* <имя полюса>). Совокупность рутин определяет слой рутин ROUT.

Слой сообщений (MES) предназначен для описания сообщений сложной структуры. Система Triad реализована таким образом, что пользователь может описать только один из слоев модели. Так, если возникает необходимость в исследовании структурных особенностей модели, то можно описать только слой структур.

Необходимо отметить особенность имитационных моделей в Triad: модель не является статической. В Triad определены операции над моделями в каждом из трёх слоёв [4]. Это операции в слое структуре: добавление и удаление вершины, добавление и удаление полюсов, добавление и удаление дуг, рёбер, объединение графов (модель представлена в виде графа), пересечение графов и т.д. В слое рутин – это добавление и

удаление событий из графа событий. В слое сообщений – добавление и удаление типов и т. д. Т. е., структура имитационной модели и логика поведения могут быть изменены динамически во время имитационного прогона, а это ещё раз подтверждает необходимость применения алгоритма динамической балансировки.

Информационные процедуры

Алгоритмом имитации назовём объекты, функционирующие по определённым сценариям, и синхронизирующий их алгоритм.

Для сбора, обработки и анализа имитационных моделей в системе Triad.Net существуют специальные объекты – информационные процедуры и условия моделирования. Информационные процедуры и условия моделирования реализуют алгоритм исследования.

Информационные процедуры ведут наблюдение только за теми элементами модели (событиями, переменными, входными и выходными полюсами), которые указаны пользователем. Если в какой-нибудь момент времени имитационного эксперимента пользователь решит, что следует установить наблюдение за другими элементами или выполнять иную обработку собираемой информации, он может сделать соответствующие указания, подключив к модели другой набор информационных процедур. Информационные процедуры являются *единственным средством системы* для одновременного доступа к элементам модели, принадлежащим разным объектам. Именно с помощью информационных процедур пользователь может взаимодействовать с объектами модели во время имитации.

Условия моделирования анализируют результат работы информационных процедур и определяют, выполнены ли условия завершения моделирования.

Система имитации Triad.Net располагает языковыми средствами для описания алгоритмов работы информационных процедур. В описание информационных процедур входят: описание начальной части (выполняется до начала имитационного эксперимента), описание заключительной части (эта часть выполняется по окончании эксперимента и служит для окончательной обработки интегральных характеристик, например, подсчет среднего значения), описания основной части информационных процедур, настроечных параметров и параметров интерфейса. При изменении значения переменной, за которой ведётся наблюдение, при выполнении события, указанного пользователем, или после прихода (передачи) сообщения на входной полюс происходит подключение информационной процедуры (основной части) к конкретному элементу модели (посредством параметров интерфейса) и данные обрабатываются по заданному в информационной процедуре алгоритму.

Таким образом, подсистема анализа модели обеспечивает получение информации по заранее сформулированному запросу, а не ограничивает пользователя строго регламентированным набором собираемых данных. Такой подход к сбору информации позволяет избежать избыточности собранной информации или того, что она окажется недостаточной.

Информационные процедуры представляют собой программные средства для мониторинга имитационной модели.

Компонент мониторинга включает стандартные информационные процедуры и функции, например: count_event(*e*) (*t*1,*t*2) – определяет, сколько раз событие *e* было выполнено за промежуток времени (*t*1,*t*2); count_state(var)(*t*1,*t*2)- определяет количество смен состояния за промежуток времени (*t*1,*t*2); count_in_signal(p) (*t*1,*t*2) – определяет количество приходов сообщений на входной полюс *p*; count_out_signal(p)(*t*1,*t*2) – определяет, сколько сообщений было отправлено с выходного полюса *p*; time_event(*e*) – фиксирует моменты времени наступления некоторого события, time_state(Val)(Var) –

фиксирует моменты времени, когда переменная Var принимает определенное значение, interval_event(*e*1,*e*2) – определяет интервал времени между двумя событиями, interval_signal (*p*1,*p*2) – определяет интервал времени между приходами сообщения на полюсы *p*1,*p*2.

Пользователь может расширить возможности компонента мониторинга, написать самостоятельно информационные процедуры, воспользовавшись средствами языка Triad.

Заключение

В статье рассмотрена архитектура подсистемы управляемой балансировки. Балансировка выполняется по правилам, которые учитывают особенности конкретной модели. Правила определяют, какой объект имитационной модели и когда следует перенести с одного вычислительного узла на другой. Эти правила задает пользователь, которому известны особенности разработанной им модели. Правила действуют на основании информации из базы данных. Сбор информации осуществляется компонентом мониторинга, который включает стандартные информационные процедуры системы Triad.Net и информационные процедуры, написанные пользователем на языке Triad.

Литература

1. **Fujimoto R. M.** Distributed Simulation Systems. In Proceedings of the 2003 Winter Simulation Conference S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, eds. P. 124–134
2. **Wilson L. F. and Wei Shen.** Experiments In Load Migration And Dynamic Load Balancing In Speedes. Proceedings of the 1998 Winter Simulation Conference. D. J. Medeiros, E. F. Watson, J. S. Carson and M. S. Manivannan, eds. P. 590–596
3. **Zheng G**. Achieving High Performance on Extremely Large Parallel Machines: Performance Prediction and Load Balancing; in Ph.D. Thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, 2005. 165 p. Доступно на сайте: http://charm.cs.uiuc.edu/
4. **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. P. 15–20.
5. **Миков А. И., Замятина Е. Б., Осмехин К. А.** Метод динамической балансировки процессов имитационного моделирования//Материалы Всероссийской научно-технической конференции «Методы и средства обработки информации МСО-2005». М.: Изд-во МГУ, 2005. Стр. 472–478.