ДИНАМИЧЕСКОЕ РАСПРЕДЕЛЕНИЕ ОБЪЕКТОВ ИМИТАЦИОННОЙ МОДЕЛИ, ОСНОВАННОЕ НА ЗНАНИЯХ

Александр Миков, Елена Замятина, Константин Осмехин

Abstract: В докладе представлена подсистема балансировки системы имитации Triad.Net,. приводится архитектура системы балансировки, обсуждаются особенности реализации статической, динамической автоматической балансировки и предлагается применять динамическую управляемую балансировку для равномерного распределения объектов имитационной модели по узлам вычислительной системы. В докладе рассматриваются также лингвистические средства языка Triad для описания алгоритмов балансировки.

Keywords: Распределённые вычисления, распределённое имитационное моделирование, статическая балансировка, динамическая балансировка, экспертные системы.

ACM Classification Keywords: I.6 Simulation and Modeling I.6.8 Types of Simulation - Distributed : I.2 Artificial Intelligence I.2.5 Programming Languages and Software - Expert system tools and techniques

Введение

В настоящее время актуальным является использование высокопроизводительных вычислительных средств для решения всё более усложняющихся задач. Такая необходимость возникает и в имитационном моделировании. С одной стороны это вызвано увеличением объёмов вычислений и желанием оптимизировать время проведения имитационных экспериментов[1,2,3]. Объекты имитационной модели распределяются по вычислительным узлам (кластера, локальной или глобальной сети) и выполняются параллельно, обычно взаимодействуя друг с другом. С другой стороны, использование распределённых имитационных вычислений диктуется необходимостью объединить уже готовые имитационные модели[1] или необходимостью коллективной работы удалённых пользователей для проведения совместных имитационных экспериментов.

Гетерогенность вычислительной среды и гетерогенность имитационной модели являются причиной того, что нагрузка вычислительных узлов и линий связи в ходе имитационного эксперимента может стать несбалансированной. Гетерогенность вычислительной среды объясняется разной производительностью вычислительных узлов и разной пропускной способностью линий связи, а гетерогенность имитационной модели тем, что некоторые объекты могут длительное время находиться в ожидании, в то время, как другие запускают одно событие за другим, всё время выполняя вычисления. Кроме того, некоторые объекты могут интенсивно обмениваться информацией, а другие – весьма редко.

Дисбаланс нагрузки приводит к тому, что выигрыш от использования распределённых вычислительных ресурсов сводится к нулю.

По этой причине возникает необходимость в разработке алгоритмов и программных средств, которые стараются сохранить баланс нагрузки на компьютерах. Эти справедливо как для распределённых имитационных экспериментов[2,3], так и для распределённых вычислений в общем случае.

Алгоритм балансировки должен быть оптимален для имитационной модели любой структуры, с любым механизмом продвижения времени. Это является нелёгкой задачей. Усилия большого числа

исследователей были направлены на разработку подобных алгоритмов [2,3]. Однако это были алгоритмы, разработанные для определённого класса задач, или алгоритмы, которые в значительной степени затрагивали код программ[2]. Из вышесказанного следует, что необходим новый подход к балансировке загрузки. Авторы [5]предлагают использовать динамическую балансировку, основанную на знаниях. В докладе обсуждается постановка задачи балансировки при выполнении имитационного эксперимента, краткий обзор алгоритмов балансировки (в том числе и алгоритмов, применяемых в распределённых системах имитации), архитектура программных средств балансировки. Далее авторы представляют архитектуру подсистемы балансировки в разрабатываемой распределённой системе имитации Triad.Net и предлагают языковые средства Triad для описания алгоритмов балансировки.

Балансировка загрузки вычислительных узлов во время имитационного эксперимента

Чаще всего в параллельном дискретно-событийном имитационном моделировании (PDES-Parallel Discreet Event Simulation) компоненты моделируемой системы представляют собой логические процессы ($LP_i, i=1\div n$), которые могут функционировать параллельно. Логические процессы распределяются между вычислительными узлами (кластера или сети), взаимодействие процессов осуществляется путём посылки сообщений от одного процесса другому. Во время выполнения имитационного эксперимента возникает конфликт между сбалансированным распределением объектов (логических процессов) по вычислительным узлам и низкой скоростью обменов сообщениями между процессами по линиям связи (линии связи перегружены или имеют низкую пропускную способность). Если логические процессы распределены между процессорами таким образом, что издержки на коммуникацию между ними сведены к нулю, то некоторые процессоры (компьютеры) могут простаивать, в то время как остальные будут перегружены. В другом случае, «хорошо сбалансированная» система потребует больших затрат на коммуникацию. Следовательно, стратегия балансировки должна быть таковой, чтобы процессоры (компьютеры) были загружены достаточно равномерно, но и коммуникационная среда не должна быть перегружена.

Следует различать *статическую* и *динамическую* балансировки. Статическая балансировка выполняется до начала имитационного прогона. При распределении логических процессов по процессорам используется опыт предыдущих имитационных прогонов (именно так происходит предварительное размещение процессов по компьютерам в SPEEDES[2]). В Triad.Net при размещении объектов имитационной модели по узлам ВС используют ещё и структурные особенности имитационной модели: объекты и подобъекты имитационной модели (а модель является иерархической) стараются расположить на одном вычислительном узле. Кроме того, выявляют клики в графе имитационной модели и они также отображаются на один вычислительный узел для того, чтобы сократить время на пересылку сообщений между узлами ВС.

Однако предварительное размещение логических процессов по процессорам (компьютерам) не всегда эффективно.

Это объясняется тем, что:

- Модель во время имитационного прогона может измениться (планирование новых событий, появление новых процессов, завершение работы процессов).
- Может измениться и вычислительная среда, в которой происходит моделирование (выход из строя компьютера или процессора).
- Компьютер (или процессор), на котором выполняется имитационная модель, занят ещё и другими вычислениями, вследствие этого доля работ не связанных с имитационной моделью, может возрасти.

Так или иначе, очень часто выигрыша от предварительного распределения логических процессов по компьютерам с целью выполнения параллельной обработки часто не наблюдают.

Динамическая балансировка предусматривает перераспределение вычислительной нагрузки на узлы *во время имитационного прогона*. В динамической балансировке можно выделить следующие этапы: *оценку* загрузки вычислительных узлов; *инициацию балансировки* загрузки, *принятие решений* о балансировке; *перемещение* объектов с одного вычислительного узла на другой.

Решение о переносе логических процессов (объектов имитационной модели) с одного узла на другой принимается на основании собранных во время имитационного прогона данных (этап оценки загрузки). Эта информация хранится в базе данных и, как правило, содержит данные двух типов:

- об имитационной модели (частота обменов между логическими процессами, количество объектов модели, располагающихся на одном вычислительном узле, продолжительность выполнения того или иного логического процесса и т.д.);
- о состоянии вычислительной системы, на которой выполняется вычислительный эксперимент (данные о загрузке вычислительного узла, о его простоях, о фоновой загрузке, о загрузке линий связи, о промежутке времени, потраченном на посылку сообщения и т.д.).

Кроме того, важно владеть информацией о том, каким образом выполняется обмен информацией между процессами, т.е. топологию обменов (или модель коммуникаций между объектами).

Оценку загрузки процессоров можно оценить аналитически (на основе знаний о поведении модели) или на основании измерений. Большинство современных машин снабжено счетчиками времени (с точностью до микросекунд), которые могут быть использованы для измерения времени выполнения каждой задачи. Сбор данных выполняется специальным программным обеспечением.

Далее следует определить, существует ли дисбаланс и принять решение о проведении перераспределения нагрузки (очень часто проводимая балансировка может привести к отсутствию выигрыша от перераспределения нагрузки).

Дисбаланс загрузки можно определить:

- *Синхронно.* Все вычислительные узлы прерывают работу в определенные моменты синхронизации и определяют дисбаланс загрузки путем сравнения загрузки отдельного процессора с общей средней загрузкой.
- *Асинхронно.* Каждый вычислительный узел хранит историю своей загрузки. В этом случае момент синхронизации для определения степени дисбаланса отсутствует. Вычислением объема дисбаланса занимается фоновый процесс, работающий параллельно с приложением.

Принятие решения о балансировке может быть выполнено:

- *Централизованно*–специальный компьютер собирает глобальную информацию о состоянии всей вычислительной системы и принимает решение о перемещении задач для каждого из вычислительных узлов.
- *Распределенно*–каждый вычислительный узел выполняет собственный алгоритм балансировки, перемещение объектов выполняется только с загруженного вычислительного узла на соседние

Последним этапом является этап перемещения объектов, при этом следует обеспечивать целостность состояния объекта.

Итак, можно сделать вывод о большом количестве алгоритмов и стратегий, используемых при выполнении этапов балансировки нагрузки. Тем не менее, общими являются обозначенные выше этапы балансировки и архитектура программного обеспечения, отвечающего за балансировку нагрузки

вычислительных узлов. Перечислим компоненты этого программного обеспечения: программные средства, обеспечивающие оценку состояния распределённой имитационной модели и вычислительной среды (подсистема анализа); управляющая программа, принимающая решение о моменте проведения балансировки, и о том, какие логические процессы следует переместить с одного процессора на другой; программные средства, реализующие перемещение объекта с процессора на процессор, подсистема визуализации, отображающая распределение компонентов имитационного моделирования по вычислительным узлам, коммуникационную среду, изменение состояния имитационной модели и вычислительной среды; базу данных, которая хранит информацию о компонентах имитационной модели и о вычислительной среде.

Теперь рассмотрим представление имитационной модели в Triad.Net и принципы построения подсистемы балансировки в Triad.Net.

Имитационная модель в Triad

Имитационная модель в Triad [4] представляет собой совокупность объектов, которые действуют по определённым сценариям и обмениваются информацией друг с другом и может быть представлена тройкой: μ = {Str, Rout, Mes}.{Str), (Rout), (Mes) – это слой структур , рутин и сообщений соответственно.

Слой структур предназначен для описания моделируемых объектов и связей между ними, слой рутин представляет собой набор алгоритмов поведения моделируемых объектов, а слой сообщений даёт возможность описывать сообщения сложной структуры. Моделируемые объекты часто имеют иерархическую структуру. Имитационная модель 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 полюса 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.

Рутина представлена множествами событий (E), состояний Q, моментов времени. 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 $e_i \in E$.

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. Исследователь имеет 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.

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, логика поведения могут быть изменены динамически во время имитационного прогона, а это ещё раз подтверждает необходимость применения алгоритма динамической балансировки.

Подсистема балансировки нагрузки в Triad.Net

Подсистема балансировки вычислительной нагрузки предназначена для оптимального размещения объектов имитационной модели по узлам ВС с целью повышения производительности системы имитации.

Задача балансировки ставится как задача отображения неизоморфных связных графов, B: TM $\rightarrow$ NG , где TM – множество графов моделей, NG – множество графов – конфигураций компьютерной сети. Граф G $\in$ NG , G = {C, Ed}, определяется множеством вычислительных узлов C и множеством ребер Ed, обозначающих линии связи. Можно рассматривать NG как супергграф, содержащий все возможные (допустимые) графы G в качестве подграфов. Граф M $\in$ TM , M = {U, V, W} , задает имитационную модель.

Рассматриваются три разновидности задачи балансировки[5]: статическая Bs, динамическая (автоматическая) Ba и динамическая (управляемая) Bc.

В алгоритме статической балансировке Bs наилучшим результатом считается отыскание подграфа G $\subset$ NG , изоморфного графу – модели M. Однако такой подграф существует далеко не всегда, поэтому предлагается метод отыскания в некотором смысле «близкого» подграфа.

В алгоритме автоматической динамической балансировки Ba графы G и M рассматриваются как нагруженные. Вершинам первого графа приписывается параметр – производительность, а его ребрам – скорость передачи данных. Во втором графе вершины характеризуются временной сложностью вычислений, а ребра – интенсивностью потоков сообщений (выходных событий).

Веса вершин и ребер графа NG (и, значит, любых его подграфов) считаются известными. Соответствующие параметры графа M должны определяться во время имитационного прогона. В соответствии с некоторым алгоритмом происходит определение «узких» мест ВС и имитационной модели и выполняется перенос объектов на менее загруженные узлы, не прерывая процесса моделирования. Алгоритм автоматической балансировки можно описать на языке Triad, пример приведён ниже.

На основании собранной статистики, используя генетические алгоритмы, для последующих имитационных прогонов конкретной модели можно найти лучшее распределение объектов по вершинам графа G. Автоматическая динамическая балансировка использует «прошлое» процесса имитационного моделирования для планирования его будущего выполнения. Однако выполнение конкретной модели может не соответствовать этому предсказанию. Действительно, опыт применения множества алгоритмов автоматической динамической балансировки не даёт заметного выигрыша в производительности. Именно автор модели может знать, как модель может вести себя в том или ином случае: например, автор модели знает, что поток заявок на конкретном устройстве будет наиболее интенсивным через 600 единиц модельного времени. Другой пример: частота обменов между двумя конкретными узлами будет наиболее интенсивной примерно через 300 единиц модельного времени. Таким образом, становится ясным, что для эффективной балансировки необходимы специальные программные и языковые средства, которые позволили бы управлять процессом балансировки.

В Triad.Net предложен алгоритм динамической балансировки, основанной на знаниях.

Управляемая динамическая балансировка использует экспертный компонент и нестандартные информационные процедуры. В экспертном компоненте сосредоточены правила оптимизации, которые

формулирует автор для данной модели (или класса моделей). Нестандартные информационные процедуры также разрабатываются автором модели и предназначены для вычисления моментов (или условий) применения правил.

Архитектура подсистемы управляемой балансировки

Итак, подсистема управляемой балансировки включает следующие компоненты:

- *Экспертный компонент*, содержащий базу знаний с правилами для оптимального размещения объектов имитационной модели по вычислительным узлам, редактор правил для модификации этих правил, механизм вывода, компонент объяснения и т.д.
- *Подсистему анализа имитационной модели и вычислительной системы*, на которой выполняется имитационный эксперимент. Подсистема анализа представлена информационными процедурами для сбора информации о поведении объектов имитационной модели (они определяют частоту обменов между объектами, частоту выполнения тех или иных событий и т.д.) и системных информационные процедур, вычисляющих загрузку узлов ВС, пропускную способность линий связи и т.д.
- *Визуализатор модели и вычислительной системы.* Визуализатор отображает статистические данные в виде графиков и диаграмм, причём пользователь может самостоятельно выбрать представление информации, используя те или иные информационные процедуры. Кроме того, визуализатор отображает распределение объектов имитационной модели на вычислительные узлы ВС.
- *Миграционную подсистему,* выполняющую перенос объектов имитационной модели с одного узла ВС на другие.

Как уже упоминалось ранее специальные объекты Triad.Net – информационные процедуры - ведут наблюдение за элементами модели, а именно, отслеживают изменение локальных переменных рутин, фиксируют наступление того или иного события и приход или отсылку сообщения с полюсов вершины. Информационные процедуры могут анализировать и сопоставлять данные от разных объектов модели во время имитационного прогона. Стандартные информационные процедуры входят в подсистему анализа Triad.Net. Кроме того, пользователь имеет возможность написать на языке Triad нестандартную информационную процедуру.

Подсистема балансировки использует информационные процедуры для визуализации хода моделирования, для наблюдения за характеристиками модели. Как только характеристики модели примут критическое значение, они передают сигнал экспертной системе, которая на основании правил выполняет операции над графом G –графом структуры модели, отображённой на граф M – граф ВС.

Правила преобразований могут быть такими:

- *если на i-ом вычислительном узле находится больше 2-х активных объектов, а j-й узел свободен, то перенести один из объектов на j-й свободный узел;*
- *если два объекта V_i и Vj в сети обмениваются данными и в сети есть линия связи Edk с большей пропускной способностью, то отобразить дугу wij на эту линию связи;*
- *если два объекта (V_i и V_j) в сети интенсивно обмениваются данными и в сети есть незанятый узел V_i, то переместить их на этот узел и т.д.*

Таким образом, правила сводятся к выполнению операций над графом G. Правила представляют собой продукции вида «if …then…else…».и могут быть описаны на языке Triad. На языке Triad может быть описан и алгоритм автоматической (неуправляемой) балансировки.

Языковые конструкции языка Triad для описания алгоритма балансировки

Приведём фрагмент описания алгоритма автоматической балансировки, выполненный на языке Triad. Этот алгоритм был использован в распределённой системе имитации SPEEDES[3]. Следуя этому алгоритму, при возникновении дисбаланса нагрузки выбирают наиболее загруженный узел. Далее возникает необходимость в выборе одного из объектов имитационной модели, находящихся на этом узле и переносе его на другой, менее загруженный узел. Выбор объекта выполняется случайным образом.

Описание имитационной модели начинается с ключевого слова *model* языка Triad и завершается ключевым словом *endmod*. Описание модели включает описания слоя структур имитационной модели (*structure …endstr*), описание слоя рутин (*routine … endrout*(поведение объектов модели)) и слоя сообщений *message…endmsg* (сообщения сложной структуры). Пусть структура имитационной модели G представляет собой граф с вершинами A,B,C,D,E,F (полный граф с вершинами A,B,C,D и двумя присоединёнными к вершине D вершинами E и F. Вершины также соединены ребом лруг с другом).

model Mod1;

var graph G; *structure* S *def*…G1:=compl(A,B,C,D)+node(E)+Node(F)+edge(E<>F)+edge(D<>E)+ edge(D<>F) …endstr

…G:=S; (* описали структуру имитационной модели, compl – графовая константа «полный граф»*)

Для описания ВС, с помощью которой выполняется имитационный эксперимент, также следует использовать слой структур языка Triad.

….*Var graph* M; (* описание ВС, которая представляет собой сеть топологии звезда, в сети 3 компьютера и в центре – компьютер О *) *structure* S1 def … G2:=*star*(3)(O,P,R,S) …*endstr*; M :=S1: …(*структура ВС*).

Для описания алгоритма статической балансировки используем процедуру, в которой пользователь может определить предварительное размещение объектов имитационной модели по вычислительным узлам. Это процедура должна быть выполнена до начала имитационного эксперимента, поэтому ссылку на процедуру следует разместить в части *conditions of simulation (условия моделирования)*. Эта часть программного кода реализуется в первую очередь. Процедура статической балансировки может выглядеть следующим образом:

procedure Static_Load_Balancing (*in ref node* G1, *ref node* V,W,X,Y,Z) *def* X:=G1; Y:=V; Z:=W;*endproc*

Описание автоматической балансировки выполняется процедурой Automated_Load_Balancing. Алгоритм, описываемый этой процедурой, выбирает случайным образом один из объектов имитационной модели, размещённый на вычислительном узле, который указывается в качестве входного параметра процедуры, и выполняет перенос этого объекта на другой вычислительный узел. Этот узел также передаётся в качестве входного параметра. Узлы определяются системными информационными процедурами как наиболее и наименее загруженные. Пусть это будут узлы M.P и M.R.

procedure Automated_Load_Balancing (in ref node X,Y) def

set U *of node* (G.A,G.B,G.C,G.D); *ref node* Q; Q:= *random*(U); Migrate(Q) *from* (X) *to* (Y)

endproc

Ссылку на процедуру Automated_Load_Balancing также размещают в условиях моделирования (в основной части после вызовов информационных процедур).

Conditions of simulation Running …

Initial … Static_Load_Balancing(compl(G.A,G.B,G.C,G.D,G),G.E,G.F,M.P,M.R,M.S) …*endi*

….(*вызов информационных процедур для сбора статистики*)…

(*вызов процедуры автоматической балансировки*)

Automated_Load_Balancing (M.P,M.R); *processing* ….*endproc*;

Endcond

Заключение

Итак, в статье описана архитектура подсистемы балансировки и языковые конструкции для описания алгоритмов балансировки. Языковые конструкции дают возможность пользователям распределённой системы имитации Triad.Net написать свой алгоритм балансировки, который наиболее адекватен поведению имитационной модели, над которой они проводят имитационный эксперимент. Подсистема балансировки наряду со статической и автоматической балансировками включает также программные средства для проведения управляемой балансировки, основанной на правилах. Правила задаёт пользователь, знающий особенности поведения конкретной имитационной модели

Библиографический список

1. [Fujimoto, 2003] 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. pp. 124-134
2. [Wilson, 1998] 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, pp.590-596
3. [Zheng, 2005] 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, 2005,165p. Доступно на сайте: http://charm.cs.uiuc.edu/
4. [Mikov, 1995] 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.
5. [Миков, 2005] Миков А.И.., Замятина Е.Б., Осмехин К.А. Метод динамической балансировки процессов имитационного моделирования. В кн. «Материалы Всероссийской научно-технической конференции «Методы и средства обработки информации МСО-2005». М.: Изд-во МГУ, 2005, стр.472-478.

Сведения об авторах

Александр Миков – АНО «Институт компьютинга», директор; Россия, г. Краснодар, ул. Аксайская, 40/1-28; e-mail: alexander_mikov@mail.ru

Елена Замятина – Пермский государственный университет, доцент кафедры математического обеспечения вычислительных систем, Россия, г. Пермь, 614017, ул. Тургенева, д. 33, к. 40; e-mail: e_zamyatina@mail.ru

Константин Осмехин – Пермский государственный университет, аспирант кафедры математического обеспечения вычислительных систем, Россия, г. Пермь, 614017, ул. Гашкова, 28-12; e-mail: kosmehin@lukoilperm.ru