Проектирование и разработка распределенной системы для итерационного сопряжения секторных гидродинамических моделей*

С.С. Самборецкий

Тюменский государственный университет

В данной работе приведена архитектура программного комплекса, представляющего собой распределенную вычислительную систему для гидродинамического моделирования нефтегазового месторождения. В основе моделирования лежит итерационный алгоритм параллельного сопряжения секторных моделей методом Шварца. В первоначальном алгоритме были распараллелены только основные этапы. Для достижения большей степени параллелизма алгоритм был модифицирован таким образом, что его программная реализация должна представлять собой распределенную систему. Рассматриваются способы решения задачи отказоустойчивости в данной распределенной системе, подход к балансировке нагрузки и инструментарий для организации взаимодействия между объектами данной системы.

1. Введение

В наши дни особо актуально моделирование крупных нефтегазовых месторождений на высокопроизводительных кластерах. Процесс моделирования представляет из себя восстановление информации об изменении параметров месторождения в динамике, согласно истории работы скважин.

Компьютерное моделирование для больших областей на базе гидродинамических симуляторов [1] требует больших вычислительных затрат, и одним из возможных методов ускорения вычислений может быть распараллеливание по подобластям, для чего к исходной области применяются методы декомпозиции [2]. Применение этих методов подразумевает, что в процессе моделирования для сохранения целостности объекта необходимо на каждом временном шаге проводить операцию сопряжения. В случае нефтегазовых месторождений параметрами сопряжения являются пластовое давление, поток флюидов и насыщенность [3].

Среди подходов к моделированию на основе декомпозиции области можно выделить два, в определенном смысле противоположных. Первый, в данный момент активно использующийся [4], – посекционное моделирование. Гидродинамический симулятор самостоятельно определяет границы областей, по которым производится распараллеливание. Недостатки такого подхода: во-первых, невозможность более детального моделирования по отдельным областям в силу использования общей сетки (имеющаяся информация по разным областям месторождения неоднородна, разные области могут иметь разную детализацию, что невозможно учесть в одной модели); во-вторых, ограничения на размерность модели, которые в данный момент до конца не преодолены; в-третьих, сопряжение секций модели – операция ресурсозатратная, т.к. требуется передавать между областями большое количество информации из узлов сетки.

Второй подход, легший в основу данного исследования – моделирование путем сопряжения секторных моделей методом Шварца [3]. Полномасштабные модели применяются с целью совершенствования систем разработки залежей нефти и газа, секторные модели – для решения задач на отдельных участках залежей. Данный подход позволяет обойти недостатки предыдущего: нет необходимости вводить ограничения на размерность всей системы секторных моделей; разные области могут иметь разную степень детализации, сетки моделей могут иметь разную размерность; передача информации для сопряжения требует меньших затрат, т.к. сопряжение производится по ячейкам, находящимся на границах смежных секторных моделей.

В контексте рассматриваемой проблемы, а именно повышение производительности программ для компьютерного моделирования, наибольший интерес представляют исследования,

* Доклад выполнен при финансовой поддержке ООО «Тюменский нефтяной научный центр»

связанные с методом Шварца, особенностями его применения, в первую очередь, возможности его распараллеливания.

2. Проектирование и разработка

2.1 Параллельный алгоритм сопряжения секторных моделей

В настоящее время глубоко изучены различные методы декомпозиции области и их параллельные реализации в общем случае [7], и, в частности, для решения задачи сопряжения секторных моделей. Рассмотрены различные варианты метода Шварца (метод Якоби-Шварца, аддитивный метод Шварца, возможные оптимизации), метод Крылова, двухуровневые методы, показаны различия в применении. Представленные результаты сравнения методов позволяют предположить, что итерационный метод Шварца, взятый за основу для решения данной задачи, может оказаться не единственным эффективным методом. Тем не менее, в работах С.В. Костюченко [3,5,6] показано, как данный подход был реализован для многоядерной рабочей станции, и подтверждена его применимость для данной задачи: чем больше область перекрытия секторных моделей, тем быстрее сходится итерационный процесс. В дальнейшем алгоритм вычислений был модифицирован [8]: вместо последовательного сопряжения (в котором этапы моделирования распараллелены, но выполняются строго последовательно с ожиданием), используется параллельное сопряжение.

Такая модификация требует пересмотра структуры вычислительной системы. Данная система должна учитывать гетерогенность секторных моделей и, соответственно, используемого аппаратного обеспечения, поскольку дальнейшим шагом её развития является создание программного комплекса для высокопроизводительной распределенной системы. При этом, программный комплекс должен удовлетворять следующим требованиям: парадигма объектно-ориентированного программирования, свободное программное обеспечение, средства поддержания вычислений в распределенной системе.

Таким образом, разработка приложения для узлов системы (далее – *приложение*) осуществляется в парадигме объектно-ориентированного программирования. Этот подход позволяет проектировать в едином ключе функционирование всех узлов вычислительной системы.

2.2 Описание структуры системы

С точки зрения общего алгоритма функционирования приложения в архитектуре выделено два вида ролей процессов.

1) Управляющий. Роль координирования работы системы, доступна только одному процессу. Основными функциями этой роли являются:

a) формирование заданий на моделирование для подчиненных процессов (задание представляет собой набор данных: путь к модели, используемый симулятор, параметры моделирования для симулятора);

b) балансировка нагрузки между узлами в соответствии с их производительностью (управляющий процесс владеет информацией о производительности всех узлов системы, и, исходя из этой информации, задание на моделирование сопоставляется соответствующему узлу);

c) формирование заданий на вычисление невязок между моделями (в промежуточных временных точках моделирования необходимо производить сопряжение смежных секторных моделей, для этого используется процедура пересчета невязок на границах; само задание представляет собой набор данных: пути к двум смежным моделям и дополнительные параметры моделирования);

d) контроль хода выполнения алгоритма, обработка частичных сбоев (управляющий процесс с определенной периодичностью проверяет функционирование остальных узлов системы, отправляя эхо-запросы, также получает ответы об успешном или неуспешном выполнении задания, исходя из чего алгоритм продолжает работать или производится обработка исключительной ситуации, например, повторная отправка задания другому процессу в случае выхода из строя того, которому оно изначально отправлялось).

2) Подчиненный. Роль, в которой процесс выполняет задания, полученные от процесса с ролью управляющего. Данная роль назначена всем процессам в системе. Основные функции:

a) сбор информации об узле, на котором запущен процесс, для дальнейшей отправки управляющему (необходимая информация – численные характеристики аппаратного обеспечения узла: тип процессора, количество ядер и частота, разрядность, объем оперативной и постоянной памяти);

b) выполнение заданий на моделирование (процесс получает задание, запускает симулятор с заданными параметрами, отслеживает работу симулятора, по результату работы отправляет ответ о выполнении задания управляющему);

c) сопряжение двух смежных секторных моделей (процесс получает задание и производит пересчет невязок на границах областей; в случае невыхода невязок за пределы допустимого значения, управляющий процесс получает ответ об успешном сопряжении, иначе отправляется запрос о повторном моделировании);

d) механизм выбора управляющего процесса (в случае выхода из строя узла, на котором работает процесс с ролью управляющего, остальные процессы должны определить, который из оставшихся будет обладать ролью управляющего на время восстановления основного, для этого должен быть реализован протокол выбора).

При проектировании приложения за основу была взята работа [9], в которой подробно описана архитектура системы, используемой для задач декомпозиции области. Количество требуемых классов меньше, чем в представленной работе, поскольку в данном комплексе не требуется учитывать вычисления гидродинамических симуляторов.

Для реализации функционала соответствующих ролей процессов были созданы два класса: Manager и Worker, реализующие методы работы управляющего и подчиненного соответственно. Сам процесс, запущенный на узле, представлен классом Process, который в зависимости от роли, может использовать объекты вышеуказанных классов.

2.3 Взаимодействие в системе с помощью передачи сообщений

В данной реализации системы взаимодействие между вычислительными узлами организовано следующим образом: управляющий процесс передает сообщения, которые содержат в себе задания двух видов – проведение вычислений на симуляторе и пересчет невязок для двух смежных моделей. В первом случае в сообщении содержится: путь к модели, используемый симулятор, дополнительные параметры. Во втором: две модели и параметры сопряжения. Исполнители в ответ посылают сообщения с результатами выполненной работы.

Таким образом, можно выделить сообщения следующих видов:

1) сообщение о состоянии процесса;
2) сообщение с информацией об узле;
3) сообщение с заданием для симулятора;
4) сообщение с заданием на сопряжение;
5) ответы на задания и сообщение о состоянии.

Для каждого сообщения создается свой класс, являющийся наследником от родительского класса Message.

2.4 Организация отказоустойчивости

Любая распределенная система должна быть устойчивой к частичным отказам [10], т.е., система продолжает функционировать после частичных отказов, незначительно снижая при этом общую производительность. Возникает необходимость рассмотреть инструментарий для обеспечения данного требования.

Для решения подобного рода задач могут быть использованы гибридные решения, в частности две общедоступные библиотеки OpenMP [11] и MPI [12]. OpenMP служит для распараллеливания вычислений на устройствах с общей памятью, MPI – для организации взаимодействия процессов с системах с распределенной памятью.

Данный подход к решению имеет ряд существенных недостатков, важным среди которых является невозможность обеспечения дальнейшей работы приложения после частичных отказов. MPI предполагает создание процессов на заданном программистом количестве узлов. Но данный стандарт не предполагает системы отслеживания работы запущенных процессов, и в случае выхода из строя одного из них, вся система незамедлительно прекращает свою работу. Другим значительным недостатком является низкоуровневость данного подхода: взаимодействие между процессами происходит путем передачи сообщений, как и в любой распределенной системе, но в случае MPI программист должен включать в алгоритм каждую пересылку сообщений, их содержимое, что усложняет решение исходной задачи.

Для отказоустойчивости необходимо обеспечить следующее поведение системы: в случае прекращения работы одного из процессов с ролью подчиненного процесс с ролью управляющего должен либо дождаться восстановления работы процесса, либо отправить задание повторно на другой свободный процесс; в случае прекращения работы процесса с ролью управляющего остальные процессы должны выбрать замену среди оставшихся на время восстановления основного управляющего. Для этого в распределенных системах используются протоколы выбора (election protocols), и при этом процессы должны соединяться друг с другом по сети.

Решением может послужить применение программного обеспечения промежуточного уровня [13]. Этот вид программного обеспечения позволяет учесть все необходимые требования к распределенным системам, в частности требование к отказоустойчивости. При отказе одного из процессов остается возможность регулировать систему. Наиболее распространенными пакетами для распределенных систем являются пакеты Ice [14], Globus Toolkit [15], NumGRID [16]. Выбор одного из данных пакетов позволит обойти недостатки подхода OpenMP+MPI.

Для дальнейшей разработки был выбран Globus Toolkit как основной инструментарий. В данной системе уже реализован менеджер управления заданиями на выполнение, и, фактически, работа программиста заключается в использовании API для взаимодействия с ним. Однако, имеет смысл также рассмотреть технологию WCF [21], разработанную Microsoft. Данная технология не является программным обеспечением промежуточного уровня, однако она позволяет ускорить процесс разработки сетевого взаимодействия между процессами и предоставляет возможность отделить клиентское приложение, необходимое для формирования задания на запуск алгоритма, от приложения, производящего вычисления. В конечном итоге система будет иметь следующую структуру: клиентское приложение с графическим пользовательским интерфейсом и сетевая служба (фоновый процесс), выполняющая алгоритм и взаимодействующая с другими службами в сети.

Таблица 1. Время работы алгоритма.

Количество ячеек	Алгоритм последовательного сопряжения для многоядерной рабочей станции		Алгоритм параллельного сопряжения для распределенной системы		
	k	t	k	t	S
10^2	40	0,9	42	0,7	0,2
10^3	86	1,22	83	1,07	0,15
10^4	152	1,8	154	1,52	0,28

(k – количество итераций, t – время (час), S - ускорение)

Технология WCF не подразумевает наличие менеджера заданий, однако такой подход позволяет спроектировать и реализовать только необходимые функции, в отличие от Globus Toolkit, предоставляющего избыточный функционал. Сложность применения данного подхода ещё несколько лет назад заключалась в отсутствии реализации для Unix-подобных операционных систем. В данный момент существует проект Mono, с недавнего времени финансируемый Microsoft, который позволяет компилировать и запускать приложения, написанные на языке C#, на Unix-подобных системах. Недавно технология WCF была частично перенесена на Mono, что позволяет также разрабатывать систему на данной платформе.

2.5 Существующие методы балансировки и идея подхода к данной задаче

Выбор промежуточного ПО не решает вопроса распределения нагрузки в гетерогенных системах. При том, что данный вопрос важен с точки зрения общей производительности. В статье [17] показана невозможность универсального решения задачи балансировки нагрузки. Действительно, несмотря на наличие большого количества разработанных методов балансировки, окончательно данная проблема решена не была, и появление новых, более сложных систем предполагает разработку новых методов балансировки. Также при балансировке стоит учитывать, что чаще для вычислений используются гетерогенные распределенные системы, т.е. такие системы, узлы которых могут иметь разную производительность.

Подходов к решению задачи балансировки нагрузки на вычислительную систему имеется несколько. Одни авторы предлагают фреймворк для разработки приложений для распределенных систем, например, [18]. Другие авторы предлагают свои реализации балансировщиков нагрузки для кластеров [19].

Авторами [20] предложен метод декомпозиции области и балансировки нагрузки на вычислительные процессы с использованием кривых Пеано. Как показано в данной статье, данный метод позволяет снизить время выполнения вычислений и дает хорошие показатели балансировки. Основная идея метода состоит в упорядочивании многомерных данных согласно кривым Пеано и последующем делении на области в одномерном порядке.

Вопрос применимости готовых решений к данной задаче остается открытым. Представленные выше балансировщики должны каким-то образом учитывать отправляемые процессам задания. В данный момент можно предположить, что механизм балансировки нагрузки необходимо разрабатывать самостоятельно. В основе работы балансировщика должна быть функция оценки производительности вычислений на известных узлах системы. Балансировщик находит максимальное значение функции для модели на разных узлах и, исходя из этого, формируется таблица соответствия задания узлу системы.

3. Выводы

Изложенные принципы легли в основу прототипа программного комплекса, предназначенного для моделирования крупного месторождения с помощью сопряжения секторных моделей. В качестве языка программирования использован C++, гидродинамический симулятор – Schlumberger Eclipse, поддержка распределенных вычислений основана на использовании Globus Toolkit. Были проведены вычислительные эксперименты на двух смежных секторных моделях небольшого размера в 10^4 ячеек, показавшие перспективность предложенной архитектуры вычислительной системы в плане повышения производительности.

Однако, требуют дальнейшего исследования вопросы балансировки нагрузки, для решения которых необходимо перейти к работе с реальными объемами данных, примером которых могут служить секторные модели для Самотлорского месторождения. В связи с этим, дальнейшая разработка предполагается на языке C# с применением технологии WCF, будет применен гидродинамический симулятор РН-КИН.

Литература

1. Кудряшов И. Ю., Максимов Д. Ю. Моделирование задач многофазной многокомпонентной фильтрации на многопроцессорных вычислительных комплексах //Препринты Института прикладной математики им. МВ Келдыша РАН. – 2009. – №. 0. – С. 68-25.
2. Барышников А. В. и др. Итерационное сопряжение как метод рационального подхода к моделированию гигантских пластовых систем // Нефтяное хозяйство. - 2011. - №20. - С. 3.
3. Костюченко С.В., Аржиловский А.В., Бикбулатова Т.Г. Технология моделирования крупных месторождений системами сопряженных секторных моделей. Часть 1. Анализ проблемной ситуации // Нефтяное хозяйство. - 2011. - №11. - С. 52-55.

4. Дзюба В.И., Литвиненко Ю.В., Богачев К.Ю., Миргасимов А.Р., Семенко А.Е., Хачатурова Е.А., Эйдинов Д.А. Технология посекционного моделирования для построения моделей гигантских месторождений. // Материалы Российской технической нефтегазовой конференции и выставки SPE по разведке и добыче. Москва, 2012.
5. Костюченко С.В. Технология моделирования крупных месторождений системами сопряженных секторных моделей. Часть 2. Метод итерационного сопряжения секторных моделей // Нефтяное хозяйство. - 2012. - №4. - С. 96-100.
6. Костюченко С.В., Толстолыткин Д.В., Чупров А.А., Шинкарев М.Б. Технология моделирования крупных месторождений системой сопряженных секторных моделей. Часть 3. Апробация технологии на примере моделей пластов $АВ_{1-5}$ Самотлорского месторождения // Нефтяное хозяйство. – 2013. – №8. – С. 78-81.
7. Dolean V., Jolivet P., Nataf F. An Introduction to Domain Decomposition Methods: algorithms, theory and parallel implementation. – 2015.
8. Костюченко С.В. и др. Алгоритм параллельного моделирования разработки гигантских нефтегазовых месторождений с сопряжением секторных моделей // Материалы V научно-практической конференции «Суперкомпьютерные технологии в нефтегазовой отрасли. Математические методы, программное и аппаратное обеспечение». Москва, 2015.
9. Копысов С. П., Красноперов И. В., Рычков В. Н. Объектно-ориентированный метод декомпозиции области //Вычислительные методы и программирование. – 2003. – Т. 4. – С. 1.
10. Таненбаум Э., ван Стеен М. Распределенные системы. Принципы и парадигмы. – СПб: Питер, 2003. – 877 с.: ил.
11. Dagum L., Menon R. OpenMP: an industry standard API for shared-memory programming //Computational Science & Engineering, IEEE. – 1998. – Т. 5. – №. 1. – С. 46-55.
12. Gropp W. et al. A high-performance, portable implementation of the MPI message passing interface standard //Parallel computing. – 1996. – Т. 22. – №. 6. – С. 789-828.
13. Bakken D. Middleware //Encyclopedia of Distributed Computing. – 2001. – Т. 11.
14. Сухорослов О. В. Промежуточное программное обеспечение Ice //Проблемы вычислений в распределенной среде/Под ред. АП Афанасьева. Труды ИСА РАН. – 2007. – Т. 32. – С. 33-67.
15. Foster I. Globus toolkit version 4: Software for service-oriented systems //Network and parallel computing. – Springer Berlin Heidelberg, 2005. – С. 2-13.
16. Fougere D. et al. NumGrid middleware: MPI support for computational grids //Parallel Computing Technologies. – Springer Berlin Heidelberg, 2005. – С. 313-320.
17. Devine K. D. et al. New challenges in dynamic load balancing //Applied Numerical Mathematics. – 2005. – Т. 52. – №. 2. – С. 133-152.
18. Barker K. et al. A load balancing framework for adaptive and asynchronous applications //Parallel and Distributed Systems, IEEE Transactions on. – 2004. – Т. 15. – №. 2. – С. 183-192.
19. Clarke D., Lastovetsky A., Rychkov V. Dynamic load balancing of parallel computational iterative routines on highly heterogeneous HPC platforms //Parallel Processing Letters. – 2011. – Т. 21. – №. 02. – С. 195-217.
20. Aluru S., Sevilgen F. Parallel domain decomposition and load balancing using space-filling curves //High-Performance Computing, 1997. Proceedings. Fourth International Conference on. – IEEE, 1997. – С. 230-235.
21. Mikulski M. A., Szkodny T. Remote control and monitoring of AX-12 robotic arm based on windows communication foundation //Man-Machine Interactions 2. – Springer Berlin Heidelberg, 2011. – С. 77-83.