Вестник Тюменского государственного университета.
Физико-математическое моделирование. Нефть, газ, энергетика. 2015. Том 1. № 3(3) 130-139

МАТЕМАТИЧЕСКОЕ МОДЕПИРОВАНИЕ, ЧИСПЕННЫЕ МЕТОДЫ И КОМППЕКСЫ ПРОГРАММ. ИНФОРМАЦИОННЫЕ ТЕХНОПОГИИ

© С. С. САМБОРЕЦКИЙ, И. Г. ЗАХАРОВА
Тюменский государственный университет
sword92@gmail.com, izharova@yandex.ru

УДК 004.94

ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ПАРАЛЛЕЛЬНОГО АЛГОРИТМА ИТЕРАЦИОННОГО СОПРЯЖЕНИЯ СЕКТОРНЫХ МОДЕЛЕЙ КРУПНЫХ МЕСТОРОЖДЕНИЙ

SOFTWARE IMPLEMENTATION OF PARALLEL ALGORITHM OF ITERATIVE CONJUGATION FOR SECTOR MODELS OF LARGE-SCALE DEPOSITS

В данной статье рассматривается программная реализация компьютерного моделирования крупных нефтегазовых месторождений. Разработана архитектура распределенной вычислительной системы, обеспечивающей сопряжение секторных моделей в ходе итерационного процесса на каждом шаге работы симуляторов. Проанализированы технологии программной реализации системы. Обоснована эффективность технологии WCF для решения данной задачи. Спроектирован протокол (интерфейс и класс), обеспечивающий передачу данных между узлами для алгоритма обработки данных на отдельных вычислительных узлах, а также классы, отвечающие за выполнение общего алгоритма. Приведены две возможные стратегии балансировки нагрузки на вычислительную систему, обоснованы их области применения.

The paper deals with the software implementation of computational modeling of large-scale oil and gas deposits. The architecture of the distributed system providing conjugation of sector models during iterative process on each step of operation of simulators is developed. The technologies of software implementation of the system are analyzed. The efficiency of the WCF technology for the solution of this task is proved.

The protocol (interface and class) providing data transfer between the nodes for the algorithm of data handling on separate computing nodes, and also the classes which are responsible for the execution of the general algorithm are designed. Two possible strategies of load balancing of the computing system are given; the application fields are demonstrated.

КЛЮЧЕВЫЕ СЛОВА. Секторное моделирование, программный комплекс, нефтегазовые месторождения, распределенные вычисления, параллельное программирование, балансировка нагрузки.

KEY WORDS. Sector modeling, software system, oil and gas deposits, distributed computing, parallel programming, load balancing.

Введение

В работах [2, 3, 4, 5] представлен алгоритм компьютерного моделирования нефтегазовых месторождений, предназначенный для параллельного сопряжения секторных моделей методом Шварца. В них было показано, что распараллеливание алгоритма сопряжения может дать существенный выигрыш в производительности по сравнению с последовательным алгоритмом. Это определило необходимость проектирования архитектуры распределенной вычислительной системы и разработки программного обеспечения, решающего данную задачу. Исходные предпосылки данной работы представлены в [4]. Дополнительным подтверждением актуальности создания системы служит успешное использование аналогичного подхода в других предметных областях [1].

В настоящее время существуют различные инструменты, позволяющие создавать распределенные системы. К их числу относятся, в первую очередь, технологии DCOM [8], CORBA [12], Java RMI [9], Windows Communication Foundation (далее WCF) [11]. В нашем случае при выборе конкретной технологии основной предпосылкой являлось то, насколько эффективно с точки зрения программной реализации и функционирования самой системы будет осуществляться обмен сообщениями. Это связано с тем, что для алгоритма сопряжения секторных моделей решение вопроса обмена данными является принципиальным. Технология WCF обладает существенным преимуществом, поскольку имеет в своем арсенале различные готовые решения для осуществления передачи информации между узлами системы. Для программной реализации распределенных вычислений требуется только выбрать подходящий *протокол* передачи данных и сформировать *контракты* данных (соответствующие понятия поясняются ниже). По сравнению с гибридной технологией распараллеливания OpenMP + MPI [7] это существенно упрощает разработку. Недостатком можно назвать медленную скорость соединения, поскольку узлы передают друг другу большие объемы данных, в отличие от случая использования MPI, однако в данной задаче это можно считать несущественным.

Особенности используемой технологии

В технологии WCF в качестве базовых выступают следующие понятия: «служба», «клиент», «контракт службы», «контракт данных» [10]. Их содержание имеет свою специфику, определяемую сущностью WCF.

В основе технологии лежит принцип связи с помощью обмена сообщениями, и любые объекты, моделируемые в виде сообщений (например, HTTP-запрос или сообщение очереди сообщений, MSMQ), можно представить единым об-

разом в модели программирования. Это обеспечивает универсальный интерфейс API для разных транспортных механизмов.

В модели различаются *клиенты*, являющиеся приложениями, которые инициируют связь, и *службы*, являющиеся приложениями, которые ожидают связи клиентов с ними и отвечают им. Одно приложение может являться как клиентом, так и службой. И именно последний подход лег в основу проектирования приложения.

Одним из обязательных элементов стека связи является протокол передачи данных. Сообщения можно отправлять через интрасети или через Интернет с помощью общих транспортов, таких как HTTP и TCP. Имеются различия в производительности передачи данных при использовании разных транспортных протоколов [10]. Для данной задачи представляется логичным использование TCP, поскольку скорость обработки и передачи данных существенно выше, чем при использовании HTTP.

Контракт службы — это модуль, объединяющий в себе несколько операций. Контракт данных — хранящееся в метаданных описание типов данных, используемых службой. Оно позволяет другим объектам работать со службой.

Проектирование протокола для взаимодействия является первоочередной задачей при построении систем, работающих на технологии WCF. Протокол в программе делится на две составляющие: интерфейс, определяющий набор вызываемых функций (в терминологии WCF — определяющий контракт службы), и класс, их реализующий. Интерфейс нужен клиентским приложениям, чтобы осуществлять соединение и не обращаться к конкретной программной реализации. Класс же реализуется и функционирует в пределах одной службы.

Специфика применения WCF при программировании состоит в том, что классам, интерфейсам и методам присваиваются атрибуты. Для определения контракта службы интерфейсу присваивается атрибут ServiceContractAttribute. Это существенно облегчает программирование с использованием данной технологии, в отличие от MPI, где на программиста накладывается больший объем работы.

Программная реализация системы

Были созданы *интерфейс IServiceSectorModelling* и *класс ServiceSectorModelling*, содержащие методы, описанные в таблице 1.

Для передачи данных в системе были разработаны классы, показанные в таблице 2. Как и в случае с контрактом службы, данные классы помечаются атрибутом DataContractAttribute.

Для выполнения алгоритма сопряжения моделей [2] были созданы классы, обеспечивающие эффективное представление разнородных данных (параметры и значения для выполнения сопряжения, информация о секторных моделях, расписание заданий для узлов, информация о конфигурации узлов) и достаточно простую реализацию локальных и глобальных алгоритмов функционирования системы.

Класс Initializer. Служит точкой входа для проекта, создает все необходимые объекты и контролирует выполнение алгоритма. В классе реализованы следующие методы:

Таблица 1

Методы интерфейса IServiceSectorModelling

Метод	Назначение
GetLeaderInfo	Передача информации от лидера о своем адресе и параметров сопряжения
GetTask	Передача задания на моделирование
GetConjugation	Передача задания на сопряжение
ModellingResult	Передача результата моделирования лидеру
ConjResult	Передача результата сопряжения лидеру
GetCurrentCommonState	Текущий статус выполнения алгоритма
GetCommonResult	Получение итогового результата (работает только при положительном/отрицательном результате выполнения алгоритма)
Launch	Запуск алгоритма, параметр — проект

Таблица 2

Классы для передачи данных между службами

Классы — контракты данных	Назначение
ConjugationProject	Класс, инкапсулирующий информацию о проекте. Под проектом подразумевается набор моделей и сопутствующей информации для выполнения алгоритма сопряжения
Node	Класс, хранящий информацию об узле
NodeTask	Родительский класс, обозначающий задание для узла, хранит общую информацию для обоих типов задания (например, временной шаг)
ModellingTask и ConjTask	Дочерние от NodeTask классы, содержащие информацию о запуске симулятора и проведении сопряжения
ModellingResult и ConjResult	Классы для передачи информации о результате выполнения задания

1) StartCalculation — метод запуска настройки системы для последующего расчета секторных моделей и их сопряжения. Псевдокод показан в листинге 1.

```
TakeProject(); // Сохранение параметров проекта
MakeObjects(); // Создание объектов из полученных параметров проекта
commonState = CommonState.InProcess; /* Смена общего состояния системы
*/
MakeLeaderCurrentNode();  /*  Назначение  текущей  службы  лидером
и  рассылка  остальным  службам  информации  о  лидере  и  параметрах
алгоритма */
scheduler.MakeSchedule(); // Создание расписания
DistributeTasks(); // Рассылка заданий свободным узлам
```

Листинг 1. Псевдокод метода StartCalculation

2) DistributeTasks — метод рассылки заданий свободным узлам, который вызывается при запуске настройки системы и при получении сообщения об успешном выполнении задания от одной из служб. Псевдокод показан в листинге 2.

```
int  modelsCount,  conjsCount,  nodesCount;  /*  Количество  оставшихся
моделей, заданий на сопряжение и свободных узлов*/
while (nodesCount > 0) // Пока имеются свободные узлы
{
if (modelsCount > 0){ //Имеются необработанные модели
     node = TakeUnusedNode(); // Свободный узел
/* Задание на моделирование */
task = scheduler.TakeModelForNode(node);
     /* Отправка задания, смена статуса узла */
TakeTask(task, node);
     modelsCount--;
     nodesCount--;
} else if (conjsCount > 0) { // Имеются задания на сопряжение
node = TakeUnusedNode();
     task = scheduler.TakeConjForNode(node);
     TakeTask(task, node);
     conjsCount --;
     nodesCount--;
} else break;
}
```

Листинг 2. Псевдокод метода DistributeTasks

3) TakeTask — метод передачи задания конкретному узлу на основе удаленного вызова процедур. Обеспечивает создание объекта, отвечающего за соединение со службой, при этом параметрами являются протокол, тип соединения и адрес. Затем следует вызов метода выполнения задания.

4) TakeModelResult — метод получения результатов моделирования (в частности, расчета секторных моделей). Псевдокод показан в листинге 3.

```
if (Result == Success)
{
     Node.Status = None; // Смена статуса узла
     /*Отметка в очереди на сопряжение */
scheduler.MarkConjs();
CheckCalculation(); // Проверка хода алгоритма
} else {
     throw new Exception(); // Исключительная ситуация
}
```

Листинг 3. Псевдокод метода TakeModelResult

5) TakeConjResult — метод получения результата сопряжения. Псевдокод показан в листинге 4.

```
if (Result == Success) // Невязка не превысила допустимое значение
{
     Node.Status = None; // Смена статуса узла
CheckCalculation();
} else { // Превысила
/* Добавление в очередь на повторное моделирование */
scheduler.MarkRestart();
CheckCalculation();
}
```

Листинг 4. Псевдокод метода TakeConjResult

6) CheckCalculation — проверка хода выполнения алгоритма по значениям набора контрольных параметров. Псевдокод показан в листинге 5.

```
/* Если очереди на данном шаге алгоритма пусты, то счетчик текущего
шага увеличивается на единицу. */
if (Queues.Count == 0) currentStep++;
/* Если предыдущий шаг был не последний, то запускается метод рассылки
заданий свободным узлам. */
if (currentStep < stepsCount) DistributeTasks();
else {
     /* Смена общего состояния системы на «Успех»*/
     commonState = Success;
     /* Формирование протоколов */
     logmaker.MakeLogs();
}
```

Листинг 5. Псевдокод метода CheckCalculation

Класс Scheduler. Используется для формирования очередей заданий на моделирование и сопряжение по узлам системы. Класс инкапсулирует следующие данные: два объекта для очередей заданий, которые представляют собой двумерные массивы очередей для моделирования и сопряжения (первое измерение — номер узла, второе — номер шага моделирования), и один объект — матрица смежности для моделей с расширенным функционалом. Такая структура данных должна решать две задачи:

— хранить информацию о смежности моделей;

— определять готовность моделей к проведению сопряжения.

1) MakeSchedule — метод для запуска построения расписаний, реализация которого опирается на методы, представленные ниже (пп. 2-3).

2) MakeModelsQueue — метод для построения расписания на моделирование (расчет моделей). При выполнении вызываются конструкторы для создания массивов нужной длины и т. п., определяется количество моделей в указанной директории. Если фактическое количество не совпадает с заявленным в проекте — происходит аварийное завершение алгоритма. Иначе производится добавление заданий в очереди к узлам. Принцип распределения заданий зависит от выбранной стратегии: прямой (задания передаются независимо от конфигурации узла системы) или оптимальной (конфигурация узла учитывается при назначении заданий).

3) MakeConjsQueue — метод для построения расписания на сопряжение. Нужен для вызова конструкторов всех необходимых объектов, т. к. задания на сопряжение возникают в ходе выполнения программы.

4) AdjacencyMark — добавление завершенной модели в матрицу смежности, т.е. смена состояния в ячейках матрицы.

5) AddConj — добавление задания на сопряжение после появления двух посчитанных смежных моделей. Появившееся задание добавляется в очередь узлу с наименьшим количеством заданий на текущий момент.

Класс SimLauncher. Класс для запуска моделирования, осуществляющегося гидродинамическим симулятором. В общем случае их может быть несколько, в предельном случае каждой модели может соответствовать свой симулятор. На вход симулятору подается модель, которая представляет собой структурированный текстовый файл или набор текстовых файлов. Данный класс нужен для того, чтобы настроить модель для передачи ее симулятору. Для этого проверяется информация по указанному в проекте шагу моделирования.

Класс Conjugator. Класс для проведения сопряжения двух смежных секторных моделей. Укажем, что сопряжение имеет смысл проводить в том случае, если на данном шаге одна из невязок (параметрами сопряжения являются давление и поток [3]) превысила допустимое значение. В этом случае производится сопряжение по ячейкам на границах смежных областей, а также повторный расчет моделей.

Класс InfoCollector. Назначение этого класса — сбор процессом-лидером информации об имеющихся узлах в локальной сети. Данный класс создавался исходя из общего случая: у узлов нет постоянных адресов, и для поддержания актуальной информации о системе необходимы широковещательные запросы об имеющихся узлах. Также данный класс необходим для проведения балансировки нагрузки: во время оповещения узлы также передают информацию о своей конфигурации (тип процессора, количество ядер, объем оперативной памяти, скорость приема-передачи данных).

Класс *LoadBalancer*. Назначение данного класса — балансировка нагрузки. Класс Scheduler при формировании заданий опирается на информацию класса InfoCollertor и создает расписание по стратегии, представленной в классе LoadBalancer.

Предположения о балансировке нагрузки

Для тестирования работы распределенной системы могут быть использованы две стратегии планирования нагрузки: прямая и оптимальная.

Под прямой стратегией понимается равномерное распределение заданий узлам независимо от их конфигурации. Таким образом, каждый узел должен будет обработать M/N моделей, где M — количество моделей, N — количество узлов. Такой подход допустим в ситуации, когда вычислительные узлы имеют одинаковую производительность, и модели мало различимы по трудоемкости. Однако данный случай можно считать идеальным, поскольку на практике такая ситуация, как правило, не встречается. Наиболее вероятна другая характеристика узлов и моделей: первые могут иметь разную производительность (другими словами, вычислительная система имеет гетерогенную структуру), а вторые — существенно различаться по трудоемкости вычислений, что может выражаться как в особенностях реального физического объекта, отраженного в модели, так и в размерах сетки. В такой ситуации нужна более оптимальная стратегия для уменьшения времени вычислений.

В основе оптимальной стратегии лежит следующая идея. Формируется структура данных на основе двух массивов или списков, отсортированных по специфическим приоритетам и связанных с вычислительными узлами и заданиями. Узлы должны быть отсортированы по производительности, а модели — по количеству ячеек или какому-либо другому параметру, характеризующему трудоемкость вычислений. Исходя из этих приоритетов, задания распределяются между узлами с возможностью варьирования способа распределения. Можно предложить схему, по которой оба ряда делятся на равные группы, и задания равномерно распределяются внутри соответствующей. Количество подгрупп может быть задано пользователем, однако представляется необходимым провести эксперименты для определения наиболее эффективного количества подгрупп.

Подобный подход представлен в распараллеливании циклов технологии OpenMP [6]. В стандарте директивы *#pragma omp parallel for* имеется опция *schedule*, которая управляет распределением работы между нитями в конструкции распределения работы цикла. Опция задает, каким образом итерации цикла распределяются между нитями; определяет вид алгоритма планирования и, если необходимо, его числовой параметр, (обычно, размер блока пространства итераций). Представленной выше идее соответствует тип *guided* — динамическое распределение итераций, при котором размер «порции» итераций на нить уменьшается с некоторого начального значения до величины chunk (по умолчанию chunk=1, размер блока) пропорционально количеству еще не распределенных итераций, деленному на количество нитей, выполняющих цикл. Размер первоначально выделяемого блока зависит от реализации. В ряде случаев такое распределение позволяет эффективнее разделить работу и динамически сбалансировать загрузку нитей.

Заключение

В данной работе представлена программная реализация распределенной вычислительной системы для параллельного алгоритма итерационного сопряжения секторных моделей. Поскольку для функционирования системы с использованием реальных данных наиболее принципиальным является обеспечение эффективного обмена сообщениями, было проведено тестирование соответствующих подсистем (передачи сообщений, составления расписания заданий, документирования выполнения алгоритма) на базе 4 рабочих станций, объединенных в локальную сеть, показавшее преимущество предложенной архитектуры в сравнении с MPI и ранее использованными подходами [3].

СПИСОК ЛИТЕРАТУРЫ

1. Белоносов М. А. Организация параллельных вычислений для моделирования сейсмических волн с использованием аддитивного метода Шварца / М. А. Белоносов // Вычислительные методы и программирование: Новые вычислительные технологии. 2012. Т. 13. № 1. С. 525-535.
2. Костюченко С. В. Алгоритм параллельного моделирования разработки гигантских нефтегазовых месторождений с сопряжением секторных моделей / С. В. Костюченко // Материалы V научно-практической конференции «Суперкомпьютерные

технологии в нефтегазовой отрасли. Математические методы, программное и аппаратное обеспечение». М. 2015.

3. Костюченко С. В. Технология моделирования крупных месторождений системами сопряженных секторных моделей. Часть 2. Метод итерационного сопряжения секторных моделей / С. В. Костюченко // Нефтяное хозяйство. 2012. № 4. С. 96-100.
4. Самборецкий С. С. О распределенной вычислительной системе для компьютерного моделирования нефтегазовых месторождений на основе итерационного сопряжения секторных моделей / С. С. Самборецкий // Вестник тюменского государственного университета. Физико-математическое моделирование. Нефть, газ, энергетика. 2015. Т. 1. № 2(2). С. 204-213
5. Самборецкий С. С. Проектирование и разработка распределенной системы для итерационного сопряжения секторных гидродинамических моделей / С. С. Самборецкий // Материалы конференции «Суперкомпьютерные дни в России». М. 2015.
6. Dagum L. OpenMP: an industry standard API for shared-memory programming / L. Dagum, R. Menon //Computational Science & Engineering, IEEE. 1998. Т. 5. No 1. Pp. 46-55.
7. He Y., Ding C. H. Q. MPI and OpenMP paradigms on cluster of SMP architectures: the vacancy tracking algorithm for multi-dimensional array transposition / Y. He, C. H. Q. Ding // Supercomputing, ACM/IEEE 2002 Conference. IEEE. 2002. P. 6.
8. Horstmann M. DCOM architecture / M. Horstmann, M. Kirtland //Microsoft Corporation, July. 1997.
9. Maassen J. Efficient Java RMI for parallel programming / J. Maassen //ACM Transactions on Programming Languages and Systems. 2001. Т. 23. No 6. Pp. 747-775.
10. Resnick S. Essential windows communication foundation: for. net framework 3.5 / S. Resnick, R. Crane, C. Bowen. Addison-Wesley Professional, 2008.
11. Smith J. Inside microsoft windows communication foundation / J. Smith. Redmond: Microsoft Press, 2007. Pp. 89-96.
12. Vinoski S. CORBA: integrating diverse applications within distributed heterogeneous environments / S. Vinoski // Communications Magazine, IEEE. 1997. Т. 35. No 2. Pp. 46-55.

REFERENCES

1. Belonosov M. A. Organizatsija paralel'nuh vuchislenij dlja modelirovanija sejsmicheskih voln s ispol'zovanijem additivnogo metoda Shvartsa [The Organization of Parallel Computings for Simulation of Seismic Waves with Use of an Additive Method of Schwartz] // Vuchislitel'nuje metodu I programmirovanije: novuje vuchislitel'nuje technologii [Computing Methods and Programming: New Computing Technologies]. 2012. T. 13. No. 1. Pp. 525-535. (In Russian)
2. Kostyuchenko S. V. Algaritm parallel'nogo modelirovanija razrabotki gigantskih neftegazovuh mestorozhdenij s soprjazhenijem sektornuh modelej [Algorithm of Parallel Simulation of Development of Large Oil and Gas Fields with Conjugation of Sector Models] // Materialu V nauchno-prakticheskoj konferentsii "Superkomp'yuternuje technologii v nefte-gazovoj otrasli. Matematicheskije metodu, programnoe I apparatnoje obespechenije" [Materials of V Scientific and Practical Conference "Supercomputer Technologies in Oil and Gas Branch. Mathematical Methods, Program and Hardware"]. M. 2015. (In Russian)

3. Kostyuchenko S. V. Technologija modelirovanija krupnuh mestirizhdenij sistemami soprjazhennuh sektornuh modelei. Chast' 2. Metod iterazionnogo soprjazhenija sektornuh modelei [Technology of simulation of large-scale deposits by systems of the conjugate sector models. Part 2. A method of iterative conjugation of sector models] // Neftjanoe hozjajstvo [Oil Economy]. 2012. No 4. Pp. 96-100. (In Russian)
4. Samboretskiy S. S. O raspredelennoj vuchislitel'noj sisteme dlja komp'yuternogo modelirovanija nefte-gazovuh mestorozhdenij na osnove iteratsionnogo soprjazhenija sektornuh modolej [On the Distributed Computing System for Computer Model Operation of Oil and Gas Fields on the Basis of Iterative Conjugation of Sector Models] // Vestnik Tjumenskogo gosudarstvennogo universiteta. Fiziko-matematicheskoje modelirovanije. Neft', gaz, energetika [Tyumen State University Herald. Physical and mathematical simulation. Oil, gas, power engineering]. 2015. T. 1. No 2(2). Pp. 204-213. (In Russian)
5. Samboretsky S. S. Proektirovanije I razrabotka raspredelitel'noj sistemu dlja iteratsionnogo soprjazhenija sektornuh modolej [Design and Development of Distributed System for Iterative Conjugation of Sector Hydrodynamic Models] // Materialu konferentsii "Superkomp'yuternuje dni v Rossii" [Materials of the "Supercomputer Days in Russia" conference]. M. 2015. (In Russian)
6. Dagum L., Menon R. OpenMP: an industry standard API for shared-memory programming //Computational Science & Engineering, IEEE. 1998. T. 5. No 1. Pp. 46-55.
7. He Y., Ding C. H. Q. MPI and OpenMP paradigms on cluster of SMP architectures: the vacancy tracking algorithm for multi-dimensional array transposition // Supercomputing, ACM/IEEE 2002 Conference. IEEE, 2002. P. 6.
8. Horstmann M., Kirtland M. DCOM architecture // Microsoft Corporation, July. 1997.
9. Maassen J. Efficient Java RMI for parallel programming // ACM Transactions on Programming Languages and Systems. 2001. T. 23. No 6. Pp. 747-775.
10. Resnick S., Crane R., Bowen C. Essential windows communication foundation: for. net framework 3.5. — Addison-Wesley Professional, 2008.
11. Smith J. Inside microsoft windows communication foundation. Redmond: Microsoft Press, 2007. Pp. 89-96.
12. Vinoski S. CORBA: integrating diverse applications within distributed heterogeneous environments // Communications Magazine, IEEE. 1997. T. 35. No 2. Pp. 46-55.

Авторы публикации

Самборецкий Станислав Сергеевич — аспирант Тюменского государственного университета

Захарова Ирина Гелиевна — доктор педагогических наук, профессор, заведующая кафедрой программного обеспечения Тюменского государственного университета

Authors of the publication

Stanislav S. Samborertskiy — Postgraduate at the Tyumen State University

Irina G. Zakharova — Dr. Sci. (Ped.), Head of the Department of Software, Tyumen State University