УДК: 20.53.19, 28.23.13

И. И. Холод
Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В. И. Ульянова (Ленина)

Возможности выполнения алгоритмов интеллектуального анализа в распределенных системах

Рассматриваются возможности выполнения алгоритмов интеллектуального анализа данных в моделях распределенных вычислений: многопоточности; акторов; map reduce. Для анализа возможностей используются методы алгебры процессов.

Распределенный анализ данных, параллельный анализ данных, интеллектуальный анализ данных, распределенные системы

Выполнение алгоритмов интеллектуального анализа данных (ИАД) требует больших вычислительных ресурсов. Это связано как с высокой математической сложностью алгоритмов, требующих сложных вычислений, так и с большим объемом данных, обрабатываемых алгоритмами, для получения максимальной пользы. Неудивительно, что последнее время выполнение такого рода алгоритмов рассматривается в контексте параллельных и распределенных вычислений. В области Data Mining (англоязычный термин ИАД) выделяются отдельные направления [1]:

– parallel data mining (PDM) предполагает выполнение алгоритмов в сильносвязанных системах, таких, как системы с разделяемой памятью (shared-memory systems – SMP), машины с распределенной памятью (distributed-memory machines – DMM) и т. п.;

– distributed data mining (DDM) предполагает выполнение алгоритмов в слабосвязанных системах, таких, как кластеры, географически распределенные системы с объединением через Интернет/Интранет.

В параллельных/распределенных вычислениях можно выделить 3 основные модели выполнения:

1. *Многопоточная модель* выполнения параллельной программы, которая предполагает параллельное выполнение отдельных ветвей алгоритмов (программ) в разных потоках, имеющих доступ к общей памяти [2].

2. *Модель акторов*, предполагающая выполнение отдельных программ на разных узлах вычислительной сети, взаимодействие между которыми осуществляется посредством обмена сообщениями [3].

3. *Модель map reduce*, предполагающая обработку данных на разных узлах с явным выделением функции отображения (map), которая выполняет предварительную обработку входных данных, и функции свертки (reduce), объединяющей предварительно обработанные данные [4].

Для анализа возможности выполнения алгоритма ИАД на данных моделях опишем его как процесс используя элементы алгебры процессов. Алгоритм ИАД можно представить как процесс, выполняющий:

– внутренние операции (анализ данных, вычисления и др.), не взаимодействующие с внешними по отношению к процессу (алгоритму) объектами;

– внешние операции, взаимодействующие с внешними по отношению к процессу (алгоритму) объектами – анализируемыми данными и моделью знаний.

С точки зрения исследования процессов, внутренние операции не вызывают интереса. Рассмотрим внешние операции:

– чтение данных (read data) – чтение определенного вектора из набора данных, хранящихся в памяти;

– чтение модели (read model) – чтение модели знаний, хранящейся в памяти;

– запись модели (write model) – изменение модели знаний (изменение ее элементов, добавление или удаление существующих), хранящейся в памяти.

Таким образом, алфавит процесса построения модели знаний алгоритмом ИАД будет представлен множеством:

$$\alpha\text{DMA} = \{\text{read\_data, write\_model, read\_model}\} \cup \cup\, \alpha\text{DMA'},$$

где αDMA' – алфавит внутренних операций процесса (алгоритма ИАД).

Сам процесс параллельного выполнения алгоритма ИАД представляет собой параллельную композицию отдельных процессов – ветвей алгоритма B_i:

$$\text{DMA} = B_1 \mid \ldots \mid B_i \mid \ldots B_n. \qquad (1)$$

Данные операции могут выполняться в произвольном порядке, в том числе и итерационно. Графически параллельный алгоритм ИАД можно представить в виде рис. 1.

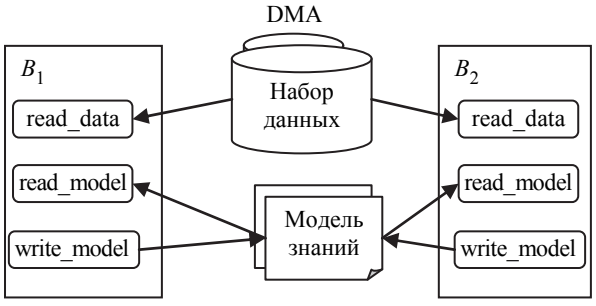

Рис. 1

В статье [5] предложена классификация алгоритмов ИАД с точки зрения их работы относительно параллельных ветвей, модели и данных. Она включает в себя следующие типы выполнения параллельных алгоритмов ИАД:

– один источник данных и одна модель (SDSM) – в этом случае параллельные ветви алгоритма работают с одним источником данных и одной моделью;

– много источников и одна модель (MDSM) – каждая ветвь алгоритма работает с отдельным источником, но с одной моделью;

– один источник и много моделей (SDMM) – каждая ветвь работает с общим источником данных, но строит каждая свою модель, которые впоследствии объединяются в одну;

– много источников и много моделей (MDMM) – каждая ветвь работает со своим источником данных и строит свою модель, которые впоследствии объединяются в одну.

Процессы анализа данных и построения модели для каждого из перечисленных типов имеют одинаковый алфавит, однако соответствующие действия выполняются по-разному.

Рассмотрим возможности реализации алгоритмов ИАД для трех перечисленных моделей параллельного выполнения, ответив для каждой из них на следующие вопросы:

1. Ограничения, накладываемые моделью программирования на параллельную форму алгоритма ИАД.

2. Выбор типа параллельных алгоритмов ИАД в зависимости от модели параллельного выполнения.

В модели многопоточности каждая параллельно выполняющаяся ветвь алгоритма работает в отдельном потоке, который взаимодействует с общей памятью. Взаимодействие заключается в выполнении операций записи и чтения памяти. Таким образом, алфавит процесса, реализующего модель многопоточности, представляет собой следующее множество:

$$\alpha\text{MT} = \{\text{write, read}\} \cup \alpha\text{MT}',$$

где αMT′ – множество внутренних операций процесса, выполняющегося в потоке.

Сам процесс многопоточного выполнения будет представлять собой параллельную композицию потоков T_i:

$$\text{MT} = T_1 \mid \ldots \mid T_i \mid \ldots T_n. \qquad (2)$$

Данная модель в общем случае не накладывает ограничений на порядок выполнения операций чтения и записи в память (рис. 2, *а*). Однако она порождает проблемы, связанные с синхронизацией доступа к общей памяти: блокировки, гонки и др. Для решения этих проблем используются различные механизмы (семафоры, критические секции и др.), которые могут накладывать некоторые ограничения на порядок выполнения операций записи и чтения.

Структурно процессы DMA (1) и MT (2) соответствуют друг другу, поэтому при использовании данной модели выполнения параллельных вычислений для алгоритмов ИАД ветви алгоритма B_i будут выполняться в отдельных потоках T_i

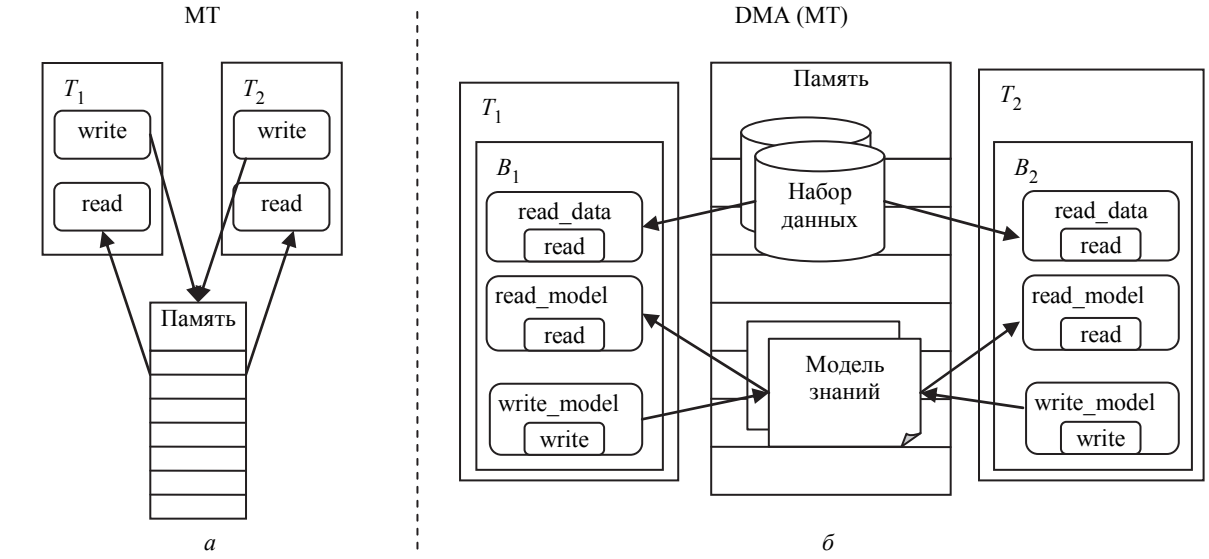

Рис. 2

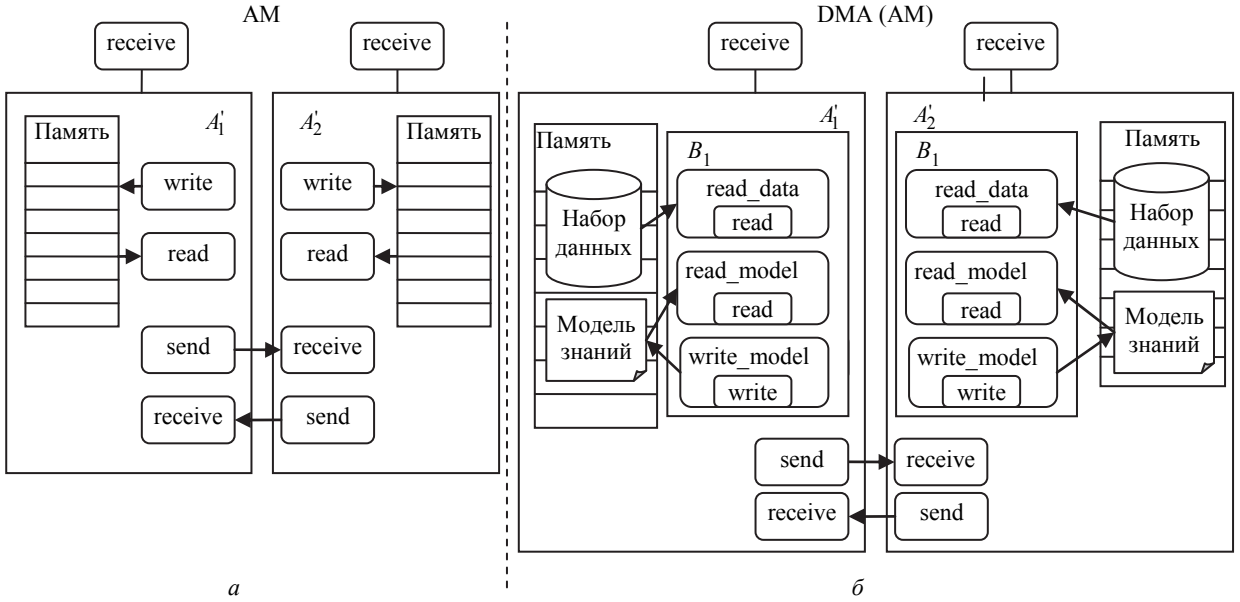

Рис. 3

(рис. 2, *б*). Размещая анализируемые данные и строящуюся модель знаний в разделяемой памяти, операции read_data, write_model, read_model (для работы с данными и моделью) процесса DMA могут использовать операции read и write.

Многопоточная модель не использует дополнительных операций и не накладывает ограничений на порядок выполнения алгоритма. В связи с этим в данной модели может быть реализован любой тип выполнения параллельных алгоритмов ИАД. При этом выбор типа SD* или MD* должен определяться наличием одного или нескольких источников данных, а выбор *SM или *MM – особенностями алгоритма. Если алгоритм строит модели знаний, которые в дальнейшем могут быть объединены без дополнительных затрат, то предпочтительнее использовать тип *MM, так как в этом случае не возникает проблем, присущих многопоточной модели при одновременной работе с одними и теми же данными. В противном случае лучше использовать тип *SM, обеспечивая синхронизацию изменения общей модели знаний средствами, предоставляемыми реализацией многопоточной модели.

Модель акторов представляет каждый процесс как актор, который, получив сообщение, выполняет некоторые операции, в том числе чтение и запись в локальную память, а также отправку сообщений. Алфавит процесса, реализующего модель акторов, включает в себя следующие операции:

$\alpha AM = \{write, read, send, receive\} \cup \alpha AM'$,

где $\alpha AM'$ – множество внутренних операций процесса, выполняющегося в акторе.

Процесс АМ также можно представить как параллельную композицию процессов – акторов A_i:

$$AM = A_1 \mid \ldots \mid A_i \mid \ldots \mid A_n. \quad (3)$$

При этом сам процесс работы актора можно представить следующим образом:

$$A_i = \text{receive} \rightarrow A_i', \quad (4)$$

т. е. процесс A_i получает сообщение, а затем начинает выполняться. При этом порядок выполнения операций в процессе A_i' может быть произвольным (рис. 3, *а*).

Как и в случае с многопоточностью, структурно процессы DMA (1) и АМ (3) соответствуют друг другу, поэтому при использовании модели акторов каждая ветвь алгоритма B_i может выполняться в отдельном акторе A_i. Однако процесс выполнения акторов имеет две особенности:

1) содержит дополнительные операции: send, receive;

2) выполнение актора инициируется получением сообщения (4).

В связи с этим для реализации алгоритма ИАД на модели акторов возможны 2 подхода:

1. Изменение алгоритма в явном виде и включение в него операций send и receive для обмена информацией (моделью знаний) между параллельными ветвями алгоритмов.

2. Декомпозиция алгоритма на блоки, которые будут выполняться между операциями receive и send (рис. 3, *б*).

Реализация первого подхода приводит к расширению алфавита процесса DMA:

$$\alpha DMA = \{read\_data, write\_model, read\_model, send\_model, receive\_model\} \cup \alpha DMA'.$$

Расширение алфавита позволяет осуществлять взаимодействие между ветвями алгоритма ИАД, для чего требуется их использование в явном виде, т. е. изменение алгоритма.

Реализация второго подхода измененяет выполнение актора, задавая строгий порядок выполнения операций:

$$A_i = \text{receive} \rightarrow A_i', \rightarrow \text{send}.$$

При этом в рамках процесса A_i будет выполняться отдельная ветвь (блок) алгоритма B_i. Данный подход требует от разработчика параллельного алгоритма ИАД декомпозиции на блоки, внутри которых не требовался бы обмен промежуточными результатами (моделями знаний).

В обоих подходах возможна реализация вариантов с *децентрализованным управлением*, когда все акторы (ветви алгоритма) равноправны и сообщения рассылаются всем, и с *централизованным управлением*, когда выделяется актор (ветвь алгоритма) диспетчер, который получает сообщения от всех, выполняет их обработку и рассылает сообщения всем.

При выборе типа выполнения параллельных алгоритмов ИАД необходимо учитывать, что данная модель применяется в системах с разделенной памятью, т. е. каждый актор работает с собственной памятью. В связи с этим наиболее естественным является использование типа MDMM, при котором каждый актор работает с собственным источником данных и строит собственную модель. При необходимости обмена промежуточными результатами выполняется рассылка модели знаний. Однако в случаях, когда имеется один источник данных, модель акторов может быть применена с централизованным управлением. При этом на диспетчера ложится функция распределения данных из источника между остальными акторами.

Модель параллельного выполнения map reduce можно представить как процесс, в рамках которого выполняются следующие операции:

– read – чтение данных из источника;

– map – выполнение пользовательской функции – предварительной обработки исходных данных;

– shuffle – перераспределение данных между имеющимися узлами обработки;

– sorted – сортировка обработанных функцией map данных;

– reduce – выполнение пользовательской функции – свертки предварительно обработанных данных функцией map;

– write – запись обработанных данных.

Таким образом, алфавит процесса выполнения программы в модели map reduce представляется следующим множеством:

$$\alpha MR = \{read, map, shuffle, sorted, reduce, write\}.$$

Модель map reduce выделяет узел, выполняющий диспетчеризацию обработки (master node), и узлы, выполняющие обработку данных (worker node) (рис. 4, *а*). Учитывая это, а также то, что модель жестко задает последовательность выполнения операций, процесс можно представить в терминах алгебры процессов следующим образом:

$$MR = MN\ |WN_1|...|WN_i|...|WN_n|, \quad (5)$$

где MN – процесс узла-диспетчера:

$$MN = read \rightarrow shuffle \rightarrow sorted \rightarrow write; \quad (6)$$

WN_i – процесс узла–обработчика:

$$WN_i = (map) + (reduce). \quad (7)$$

Структурно процесс DMA (1) не соответствует процессу MR (5), так как у последнего процессы неравнозначны и выделяется процесс-диспетчер MN. Однако данный процесс является служебным и не используется для реализации пользовательских функций, поэтому процесс для выполнения алгоритмов ИАД можно представить следующим образом:

$$MR' = MR \,/\, MN = WN_1\ |...|\ WN_i\ |...|\ WN_n. \quad (8)$$

Таким образом, процесс MR' (8) приведен в соответствие с процессом DMA (1), и в этом случае процессы-обработчики WN_i могут использоваться для выполнения параллельных ветвей алгоритма ИАД (рис. 4, *б*). При этом особенностями модели map reduce являются:

1. Строгий порядок работы с данными (6).

2. Явное разделение процесса обработки данных на функции map и reduce (7).

3. Реализация пользовательских вычислений в операциях map и reduce.

Перечисленные особенности накладывают следующие ограничения на использование модели map reduce для параллельного выполнения алгоритмов ИАД:

1. Параллельные ветви алгоритма ИАД должны выполнять один проход по набору данных, так как строгий порядок выполнения (6) не предполагает итерационной обработки.

2. В параллельных ветвях алгоритма должна выполняться функция обработки данных, обладающая свойством списочного гомоморфизма [6], так как операции отображения (map) и свертки (reduce) корректно работают только для таких функций.

3. Алгоритм ИАД должен быть явно декомпозирован на блоки, выполняющие функции map и reduce.

Данные ограничения приводят к необходимости существенного изменения алгоритмов ИАД для их реализации в модели map reduce.

В модели map reduce наличие узла-диспетчера MN и выполнение им функций чтения данных приводит к необходимости работать только с одним источником данных. При этом параллельная обработка данных на узлах-обработчиках WN_i приводит к появлению на каждом узле промежуточных результатов и фактически – к построению собственной модели знаний, которые в

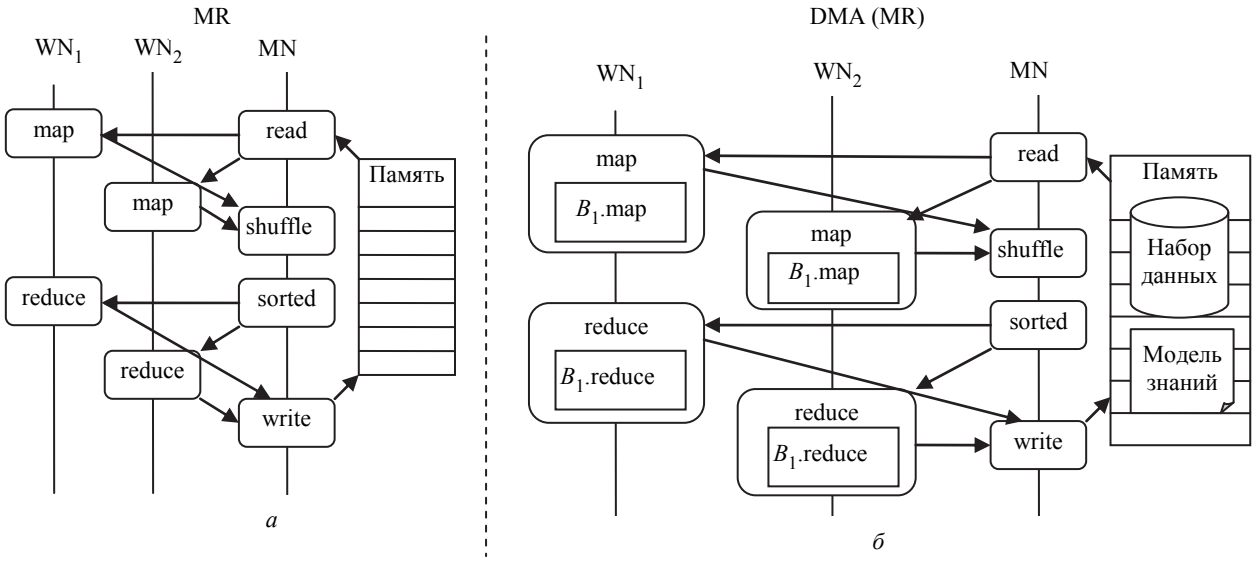

Рис. 4

итоге объединяются в одну узлом-диспетчером. Таким образом, модель map reduce позволяет реализовывать только один тип выполнения параллельных алгоритмов ИАД – SDMM.

Проведенный анализ возможностей реализации параллельных алгоритмов ИАД на разных моделях выполнения параллельных/распределенных вычислений показал следующее:

1. Многопоточная модель не накладывает никаких ограничений на реализацию алгоритма ИАД. В нем может быть реализован любой тип выполнения параллельного алгоритма ИАД. Однако в случае модели *SM необходимо использовать механизмы синхронизации для обеспечения корректного доступа к модели знаний, хранящейся в общей памяти.

2. Модель акторов также может быть использована для реализации параллельных алгоритмов ИАД, однако на реализацию таких алгоритмов накладывается ограничение, связанное или с изменением алгоритма для использования операций приема и отправки сообщений, или с декомпозицией на блоки, не требующие обмена промежуточными результатами. На данной модели могут быть реализованы 2 типа выполнения параллельных алгоритмов ИАД: MDMM или SDMM (с использованием диспетчера).

3. Модель map reduce накладывает самые большие ограничения на параллельную реализацию алгоритмов ИАД. Такие алгоритмы должны быть декомпозированы на блоки, не выполняющие итерационную обработку данных, отдельно реализующие функции отображения (map) и свертки (reduce), а также использующие для обработки данных только функции, обладающие свойством списочного гомоморфизма.

Окончательный выбор модели реализации может быть сделан в зависимости от особенностей алгоритма ИАД, а также от условий его выполнения (например, от системы, в которой он будет выполняться с разделяемой или разделенной памятью).

Исследования, результаты которых описаны в данной статье, выполнены в СПбГЭТУ при финансовой поддержке Министерства образования и науки Российской Федерации в рамках договора № 02.G25.31.0058 от 12.02.2013 г.

СПИСОК ЛИТЕРАТУРЫ

1. Zaki M. J., Ho C.-T. Large-Scale Parallel Data Mining. Berlin Heidelberg: Springer-Verlag, 2000. P. 1–23.

2. Таненбаум Э., ван Стеен М. Распределенные системы. Принципы и парадигмы. СПб.: Питер, 2003. 877 с.

3. Clinger W. Foundations of Actor Semantics / Massachusetts Inst. of Technology. Massachusetts, 1981. 178 p.

4. Dean J., Ghemawat S. MapReduce: Simplified Data Processing on Large Clusters // Proc. of the USENIX Symp. on Operating SystemsDesign & Implementation (OSDI), Calif, USA, San Francisco, 2004. P. 137–147.

5. Холод И. И. Способы параллелизации алгоритмов интеллектуального анализа // Изв. СПбГЭТУ «ЛЭТИ». 2012. № 9. С. 39–47.

6. Gorlatch S. Extracting and implementing list homomorphisms in parallel. // Program development Science of Computer Programming. Vol. 33, № 1. P. 1–27.

I. I. Kholod

Saint-Petersburg state electrotechnical university «LETI»

THE ABILITY TO PERFORM DATA MINING ALGORITHMS IN DISTRIBUTED SYSTEMS

This article discusses the possibility of the data mining algorithms in the distributed models: multi-threading, actors model and MapReduce. For research is applied the methods of process algebra.

Distributed data mining, parallel data mining, data mining, distributed systems