I. A. Khakhaev, E. N. Shapovalov
JSC «Research Institute of software»
A. I. Weintraub
Saint Petersburg Electrotechnical University «LETI»
K. G. Zinoviev
Cosmodrome «Plesetsk»

COMPARATIVE ESTIMATION OF RESOURCES FOR E-LEARNING

The task of comparison of different resources for e-Learning is described. The goal is to select optimized solution for learning process support by e-Learniong system. Some generalized properties which may be taken into consideration at the comparison are discussed. The approach to above mentioned comparison and to visualization of comparison results is also proposed.

Resources for e-Learning, software quality, expert assessment

УДК 004.82, 004.89

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

Метод оценки эффективности анализа данных в распределенной среде

Описывается метод конфигурирования сети акторов для выполнения интеллектуального анализа данных в распределенной среде. Исходными данными для метода являются характеристики набора данных, алгоритма анализа и среды выполнения. В результате применения метода вычисляются количество кластеров сети акторов, количество акторов разных типов, а также определяется размещение акторов в среде выполнения.

Интеллектуальный анализ данных, распределенный анализ, распределенные системы, модель акторов

В последнее время наблюдается существенный рост интереса к технологиям больших данных [1] и Интернета вещей [2]. Общим для них является необходимость обработки данных, получаемых из разных источников в режиме реального времени в распределенной среде. До недавнего времени основными системами, решающими эти задачи, являлись системы, основанные на парадигме Map Reduce [3], такие, как Apache Hadoop [3], Apache Spark [4], Apache Mahout [5] и др. Принципиальным аспектом их работы является централизованная обработка на базе вычислительных кластеров (например, в «облачной» вычислительной среде [6]) заранее собранных данных в едином хранилище. Данный принцип имеет следующие основные недостатки:

– высокие требования к ресурсам (как вычислительным, так и дисковым), которые растут с увеличением объемов собираемой информации;

– большой сетевой трафик, связанный с переносом данных от источников информации к единому хранилищу;

– высокие требования к защите информации, хранящейся в едином хранилище, так как ее гриф секретности повышается по совокупности.

Кроме того, концепция Map Reduce имеет концептуальный недостаток – с ее помощью естественно решаются только задачи, обладающие свойством списочного гомоморфизма [7]. В противном случае или требуется существенная модификация стандартных методов решения задачи, или использование другой концепции выполнения распределенных вычислений.

В противовес централизованной обработке, на базе облачных вычислений (cloud computing), в последнее время все большую популярность набирает технология «туманных вычислений» (fog computing) [8]. Она предполагает перенос части вычислений от центрального кластера в среду передачи данных, «ближе» к источнику информации. Такой подход становится более актуальным в связи с интеллектуализацией любого устройства (наличием у него вычислительного

процессора и памяти, что позволяет обрабатывать информацию непосредственно на нем).

В качестве средства выполнения анализа и обработки данных в распределенной среде могут использоваться акторы [9]. Акторы являются более общим формализмом по сравнению с Map Reduce и более гибким. В отличие от систем, построенных на парадигме Map Reduce, акторы могут произвольно располагаться на узлах распределенной среды. Они взаимодействуют друг с другом посредством обмена промежуточными результатами за счет механизма асинхронного обмена сообщениями. Таким образом они создают сеть акторов [10], совместно решающих общие задачи. Конфигурированием сети акторов можно обеспечить оптимальную обработку и анализ информации.

В данной статье описывается метод оценки эффективности выполнения обработки данных в распределенной среде в зависимости от характеристик среды выполнения, характеристик алгоритма интеллектуального анализа и характеристик обрабатываемых данных.

Описание задачи. Алгоритм интеллектуального анализа данных (ИАД) можно представить как композицию чистых функций [11]:

$$f = f_n \circ f_{n-1} \circ \ldots \circ f_i \circ \ldots f_1 \circ f_0,$$

где f_i – блок алгоритма ИАД, выполняемый как чистая функция:

$$f_i(d, m) = m.$$

Здесь d – набор входных данных, анализируемых блоком; m – модель знаний, изменяемая блоком.

В зависимости от использования набора данных можно выделить блоки:

– обрабатывающие – использующие набор данных:

$$f_i^d(d, m) \neq f_i^d(\text{nil}, m);$$

– вычислительные – не использующие набор данных:

$$f_i^c(d, m) = f_i^c(\text{nil}, m).$$

Таким образом, множество блоков алгоритма ИАД разбивается на 2 подмножества – обрабатывающих и вычислительных блоков:

$$f = f^d \cup f^c = \left\{ f_1^d, \ldots, f_i^d, \ldots, f_s^d \right\} \cup \cup \left\{ f_1^c, \ldots, f_j^c, \ldots, f_h^c \right\}.$$

Для параллельного алгоритма ИАД в его состав вводится блок распараллеливания $f_{i\,\text{parall}}$ [11], который параллельно выполняет декомпозированные блоки блока f_i:

$$f_{i\,\text{parall}} = \text{parall}\left(\left[f_{i.1}, \ldots, f_{i.r}, \ldots, f_{i.k} \right], d, m \right),$$

где $f_i = f_{i.k} \circ \ldots \circ f_{i.r} \circ \ldots \circ f_{i.1}$.

Для параллельного выполнения алгоритма ИАД в распределенной среде могут использоваться акторы [10]. Сеть акторов можно представить как набор акторов, выполняющих различные блоки алгоритма ИАД. При этом сеть акторов может состоять из нескольких систем акторов. Каждую систему акторов можно представить в виде множества

$$A' = \{a_1, a_2, \ldots, a_q, \ldots, a_p\},$$

где a_q – актор, выполняющий некоторый блок алгоритма ИАД f_i, с набором данных d и моделью знаний m:

$$m = a_q(f_i, d, m).$$

В итоге, результатом работы актора является измененная модель знаний m, которую он передает другому актору.

В зависимости от типа блока алгоритма (обрабатывающий или вычислительный), выполняемого актором, можно выделить 2 типа акторов:

– обработчики – акторы, выполняющие обрабатывающие блоки и взаимодействующие с данными:

$$m = a_x^d(f_i, d, m);$$

– вычислители – акторы, выполняющие вычислительные блоки и не взаимодействующие с данными:

$$m = a_y^c(f_i, \text{nil}, m).$$

Таким образом, множество акторов разбивается на 2 подмножества: обработчики и вычислители:

$$A = \{r\} \cup A^d \cup A^c = \{r, p\} \cup \cup \left\{ a_1^d, \ldots, a_x^d, \ldots, a_y^d \right\} \cup \left\{ a^c{}_1, \ldots, a_u^c, \ldots, a_t^c \right\}.$$

Всю сеть акторов можно представить в виде множества систем акторов A'_v, а также специальных акторов, обеспечивающих работу всей сети и входящих в состав одной из систем:

$$A = \{\{r, p\} \cup A'_1, \ldots, A'_v, \ldots, A'_w\},$$

где p – актор, выполняющий последовательную часть алгоритма и блок распараллеливания parall; r – специальный актор – диспетчер сообщений.

Распределенная среда выполнения, в которой должен выполняться алгоритм ИАД, состоит из вычислительных узлов и узлов хранения данных:

$$E = H \cup S = \{s_1, ..., s_f, ..., s_e\} \cup \{h_1, ..., h_k, ..., h_g\},$$
$$e \geq 1,$$

где h_k – вычислительный узел среды; s_f – узел хранения данных среды.

Среда выполнения, в которой выполняется анализ данных, должна иметь хотя бы один узел хранения данных.

На любом узле могут параллельно выполняться несколько потоков t_q:

$$h_k = \{t_1, ..., t_v, ..., t_z\}.$$

В зависимости от структуры можно выделить следующие типы сред:

– сильносвязанную, в которой вычислительные узлы интегрированы высокоскоростными каналами (например, многоядерные или многопроцессорные системы);

– слабосвязанную, в которой вычислительные узлы соединены каналами связи, имеющими ограниченную пропускную способность, вносящую задержку в их взаимодействие (например, сеть вычислительных узлов);

– многоранговую, в которой объединяются несколько сильно- или слабосвязанных сред с разными политиками безопасности, требующих как проверки передаваемой информации, так и ограничивающих передачу этой информации.

Сильносвязанная среда имеет один узел хранения, являющийся вычислительным узлом:

$$E_s = \{h_1 = s_1\}.$$

Слабосвязанная среда имеет несколько вычислительных узлов и/или узлов хранения. При этом если узел хранения один:

$$E_w^c = \{s_1, h_1, h_2, ..., h_k, ..., h_r\},$$

то данные имеют централизованный способ хранения – d_1.

Если узлов хранения несколько:

$$E_w^d = \{s_1, ..., s_f, ..., s_e, h_1, ..., h_k, ..., h_g\},$$

то данные имеют распределенный способ хранения – d_m.

Многоранговая среда включает в себя несколько сильно- или слабосвязанных сред E_i :

$$E_m = \{E_1, E_2, ..., E_i, ..., E_j, ..., E_z\}.$$

Для многоранговой среды может быть задан граф разрешения пересылки данных, соответствующий политике безопасности:

$$S = \{(E_i, E_j) \mid E_i \in E_m, E_j \in E_m\}.$$

Наличие дуги (E_i, E_j) в графе S означает разрешение на пересылку данных между средой E_i и средой E_j.

Для эффективного выполнения анализа данных в распределенной среде должна быть правильно сконфигурирована сеть акторов. Под конфигурацией сети акторов C будем понимать результат отображения сети акторов на распределенную среду выполнения:

$$A \to E \to C = A \to E \to (AxE) =$$
$$= \left\{\left(s_f, a_x^d\right), \left(h_k, a_y^c\right) \mid a_x^d, a_y^c \in A; h_k, s_f \in E\right\}.$$

Учитывая, что на одном узле может выполняться несколько акторов, объединенных в системы, сгруппируем элементы множества C по узлам среды:

$$C = \{(s_f, A_v'), (h_k, A_b') \mid A_v', A_b' \in A; h_k, s_f \in E\}.$$

Конфигурация акторов представлена на рис. 1.

При определении конфигурации сети акторов необходимо решить следующие задачи:

1. Определить характеристики сети акторов:

а) вычислить количество систем акторов – w;

б) для каждой системы акторов вычислить количество акторов-обработчиков – y;

в) для каждой системы акторов вычислить количество акторов-вычислителей – t.

2. Распределить акторы по узлам среды – C.

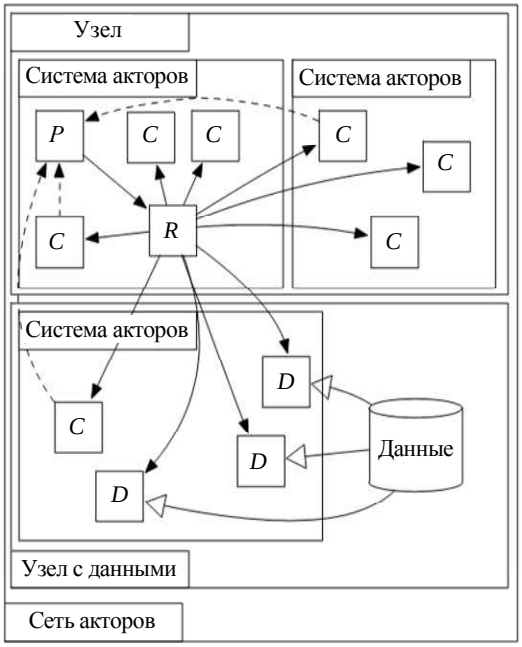

Рис. 1

Метод конфигурирования сети акторов для анализа данных в распределенной среде. Исходными данными для конфигурирования сети акторов являются:

1) характеристики обрабатываемых данных – d;

2) характеристики алгоритма ИАД – f и модели знаний – m;

3) характеристики среды выполнения – E.

К характеристикам обрабатываемых данных относится способ их хранения:

– централизованное хранение (d_1) – все данные хранятся на одном источнике;

– распределенное хранение (d_m) – все данные хранятся на разных (распределенных) источниках. При этом возможны следующие варианты:

– вертикальное распределение – на каждом узле хранится полный набор данных, но атрибуты разные на каждом из узлов;

– горизонтальное распределение – на разных узлах хранятся разные векторы, имеющие одинаковый состав атрибутов.

Вторым параметром данного метода являются характеристики алгоритма ИАД. Алгоритм может характеризоваться следующими свойствами:

– типом распараллеливания (по данным или по задачам);

– количеством обрабатывающих блоков – s;

– количеством вычислительных блоков – h.

Последним параметром является тип среды выполнения:

– сильносвязанная – E_s;

– слабосвязанная – E_w;

– многоранговая – E_m.

Если задана ***сильносвязанная среда*** выполнения с одним узлом хранения $E_s = \{s_1\}$, то сеть акторов будет содержать одну систему $w = 1$.

При одноядерном узле s_0 алгоритм анализа будет выполняться в последовательном виде. В этом случае можно использовать один актор или не использовать акторы совсем.

При многоядерном узле:

$$s_k = \{t_1, ..., t_q, ..., t_p\}$$

рекомендуется создание системы акторов следующей конфигурации:

$$A = \{r,\ p,\ a_1, ..., a_q, ..., a_p\}.$$

При этом все акторы размещаются на единственном узле хранения:

$$C = \left\{\left(s_1, r, p, a_1^d, ..., a_x^d, ..., a_y^d, a_1^c, ..., a_u^c, ..., a_t^c\right) \mid a_x^d,\ a_u^c \in A, s_1 \in E_S; y = s, t = |\, s_1\,| - s\right\}.$$

Общее число акторов не должно превышать количества потоков на узле $|\, s_1\,|$. Количество акторов-обработчиков определяется количеством блоков алгоритма – s, обрабатывающих данные. Количество акторов-вычислителей определяется оставшимся свободным количеством потоков $|\, s_1\,| - s$.

Если задана ***слабосвязанная среда***, то сеть акторов включает системы акторов не меньше количества узлов среды (на каждом узле размещается минимум одна система акторов) $w \geq r + e$.

Если среда выполнения включает только один узел хранения информации (т. е. *централизованное хранение информации*) E_w^c, то для *алгоритма, распараллеленного по данным*, количество акторов-обработчиков в системе акторов, размещенных на узле s_1, не должно превышать количества ядер на нем ($y \leq |\, s_1\,|$), а количество акторов-вычислителей в каждой системе акторов, размещенных на узле h_k, не должно превышать суммарного количества потоков на нем ($t \leq |\, h_k\,|$):

$$C = \left\{\left(s_1, a_1^d, ..., a_x^d, ..., a_y^d\right), \left(h_k, a_1^c, ..., a_u^c, ..., a_t^c\right), \left(h_1, r, p, a_1^c, ..., a_u^c, ..., a_t^c\right) \mid a_x^d,\ a_u^c \in \right.$$
$$\left. \in A, s_1 \in E_w; y = |\, s_1\,|, t \leq |\, h_k\,|\right\}.$$

Если *алгоритм распараллелен по задачам*, то число акторов-обработчиков в системе акторов, размещенных на узле s_1, определяется количеством обрабатывающих блоков алгоритма, использующих наборы данных в параллельной части алгоритма ($y \leq \left|\, f^d \,\right|$). Количество акторов-вычислителей во всех системах акторов, размещенных на узлах, определяется количеством вычислительных блоков в параллельной части алгоритма ($t \leq |\, f^c\,|$):

$$C = \left\{\left(s_1, a_1^d, ..., a_x^d, ..., a_y^d\right), \left(h_1, r, p, a_1^c, ..., a_u^c\right), \left(h_k, a_{u*k}^c, ..., a_t^c\right) \mid a_x^d, a_u^c \in A, h_1, h_k, s_1 \in \right.$$
$$\left. \in E_w; y \leq |\, f^d\,|, t \leq |\, f^c\,|\right\}.$$

Если задана слабосвязанная среда, включающая несколько узлов хранения информации (т. е. *распределенное хранение информации*) E_w^d, то для

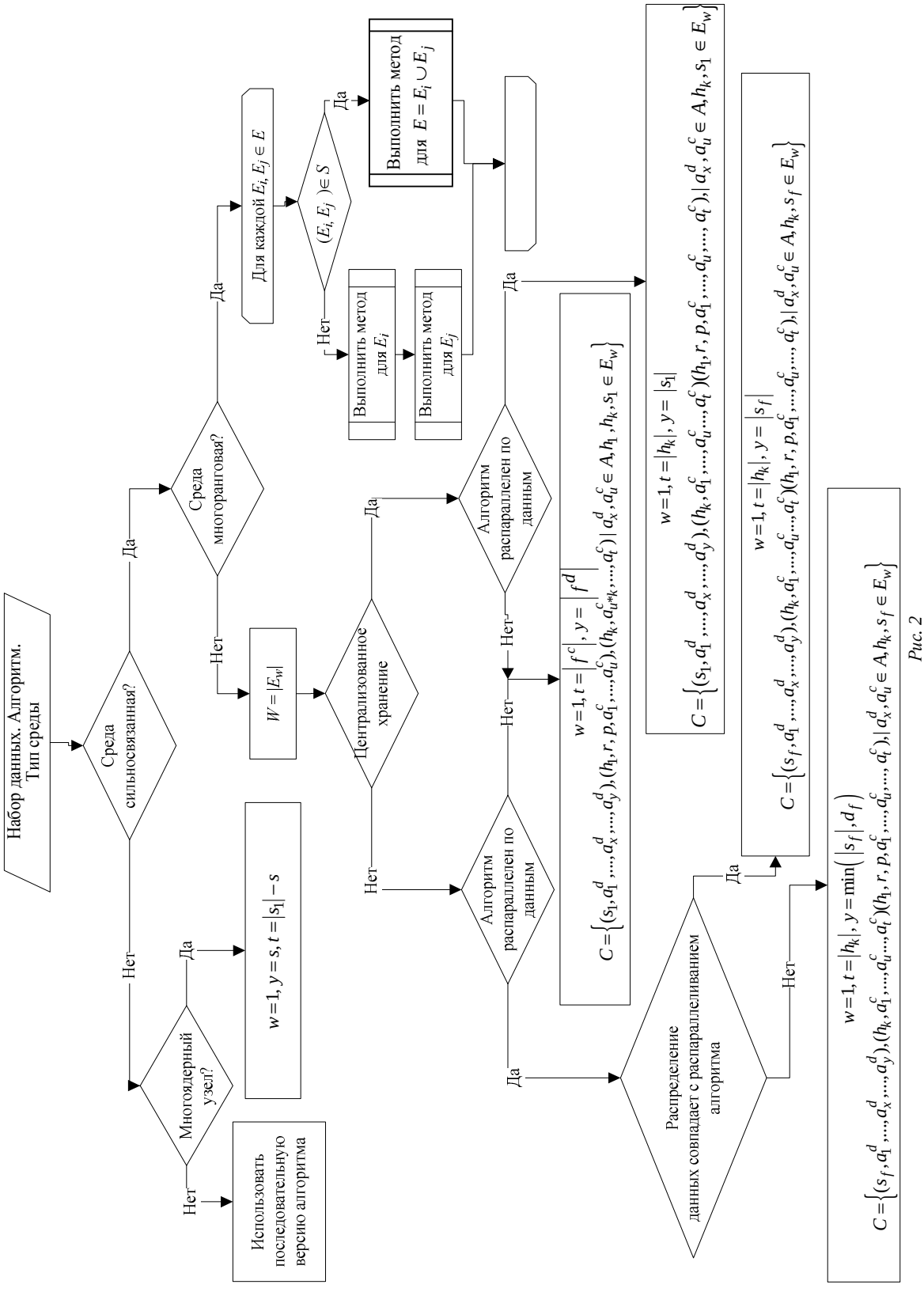
Набор данных. Алгоритм. Тип среды
Среда сильносвязанная?
Да
Нет
Многоядерный узел?
Да
Нет
$w=1, y=s, t=|s_1|-s$
Использовать последовательную версию алгоритма
Среда многоранговая?
Да
Нет
$W=|E_w|$
Для каждой $E_i, E_j \in E$
$(E_i, E_j) \in S$
Да
Нет
Выполнить метод для E_i
Выполнить метод для E_j
Выполнить метод для $E = E_i \cup E_j$
Централизованное хранение
Да
Нет
Алгоритм распараллелен по данным
Да
Нет
$w=1, t=\left|f^c\right|, y=\left|f^d\right|$
$C=\left\{(s_1, a_1^d, ..., a_x^d, ..., a_y^d), (h_1, r, p, a_1^c, ..., a_u^c), (h_k, a_{u^*k}^c, ..., a_t^c) \mid a_x^d, a_u^c \in A, h_1, h_k, s_1 \in E_w\right\}$
Алгоритм распараллелен по данным
Да
Нет
$w=1, t=|h_k|, y=|s_1|$
$C=\left\{(s_1, a_1^d, ..., a_x^d, ..., a_y^d), (h_k, a_1^c, ..., a_u^c ... a_t^c)(h_1, r, p, a_1^c, ..., a_u^c, ..., a_t^c) \mid a_x^d, a_u^c \in A, h_k, s_1 \in E_w\right\}$
Распределение данных совпадает с распараллеливанием алгоритма
Да
Нет
$w=1, t=|h_k|, y=|s_f|$
$C=\left\{(s_f, a_1^d, ..., a_x^d, ..., a_y^d), (h_k, a_1^c, ..., a_u^c ... a_t^c)(h_1, r, p, a_1^c, ..., a_u^c, ..., a_t^c) \mid a_x^d, a_u^c \in A, h_k, s_f \in E_w\right\}$
$w=1, t=|h_k|, y=\min\left(|s_f|, d_f\right)$
$C=\left\{(s_f, a_1^d, ..., a_x^d, ..., a_y^d), (h_k, a_1^c, ..., a_u^c ... a_t^c)(h_1, r, p, a_1^c, ..., a_u^c, ..., a_t^c) \mid a_x^d, a_u^c \in A, h_k, s_f \in E_w\right\}$

Рис. 2

алгоритма, *распараллеленного по данным*, может быть несколько вариантов построения сети акторов:

– если распределение данных совпадает с распараллеливанием алгоритма (т. е. при горизонтальном распределении данных происходит распараллеливание алгоритма по векторам, а при вертикальном распределении данных – распараллеливание по атрибутам), то количество акторов-обработчиков в каждой системе акторов, размещенных на узле s_f, определяется суммарным количеством потоков на нем ($y \le |s_f|$), а количество акторов-вычислителей в каждой системе акторов, размещенных на узле h_k, определяется количеством потоков на нем ($t \le |h_k|$):

$$C = \left\{ \left(s_f, a_1^d, ..., a_x^d, ..., a_y^d \right), \left(h_1, r, p, a_1^c, ..., a_u^c, ..., a_t^c \right), \left(h_k, a_1^c, ..., a_u^c, ..., a_t^c \right) | \, a_x^d, a_u^c \in A; h_k, s_f \in E_w; y \le |s_f|, t \le |h_k| \right\};$$

– если распределение данных и распараллеливание алгоритма не совпадают, то количество акторов-обработчиков в каждой системе акторов, размещенных на узле s_f, определяется минимальным из количества потоков на узле s_f и количеством атрибутов (векторов), хранящихся на узле s_f ($y = \min(|s_f|, d_f)$), а количество акторов-вычислителей в каждой системе акторов, размещенных на узле h_k, определяется количеством потоков на нем ($t \le |h_k|$):

$$C = \left\{ \left(s_f, a_1^d, ..., a_x^d, ..., a_y^d \right), \left(h_1, r, p, a_1^c, ..., a_u^c, ..., a_t^c \right), \left(h_k, a_1^c, ..., a_u^c, ..., a_t^c \right) | \, a_x^d, a_u^c \in A; h_k, s_f \in E_w; y \le \min\left(|s_f|, d_f \right), t \le |h_k| \right\}.$$

Если алгоритм *распараллелен по задачам*, то, как и в случае с централизованным хранением, количество акторов и конфигурация сети акторов определяются блоками алгоритма ИАД:

$$C = \left\{ \left(s_1, a_1^d, ..., a_x^d, ..., a_y^d \right), \left(h_1, r, p, a_1^c, ..., a_u^c \right), \left(h_k, a_{u*k}^c, ..., a_t^c \right) | \, a_x^d, a_u^c \in A, h_1, h_k, s_1 \in E_w; y \le |f^d|, t \le |f^c| \right\}.$$

Если задана ***многоранговая среда***, то для каждой отдельной среды применяются правила, соответствующие ее типу. Политика безопасности S должна учитываться при размещении акторов между средами:

– если для двух сред E_i и E_j в политике безопасности S существует разрешение $(E_i, E_j) \in S$, то их можно рассматривать как единую среду $E = E_i \cup E_j$;

– если для двух сред E_i и E_j в политике безопасности S отсутствует разрешение $(E_i, E_j) \notin S$, то каждую из них нужно рассматривать в отдельности или разделять акторы-обработчики и акторы-вычислители по разным средам.

Описанный метод представлен в виде блок-схемы на рис. 2.

В статье описан метод конфигурирования сети акторов для распределенного анализа данных на основе оценки исходных условий выполнения алгоритма:

– характеристик набора данных;

– характеристик среды выполнения;

– характеристик алгоритма анализа.

В результате использования метода определяются основные характеристики сети акторов для их эффективного выполнения в распределенной среде:

1) количество кластеров акторов;

2) для каждого кластера акторов количество акторов, взаимодействующих с данными;

3) для каждого кластера акторов количество акторов, не взаимодействующих с данными;

4) размещение акторов в среде выполнения.

Данные характеристики позволяют однозначно построить сеть акторов для выполнения заданного параллельного алгоритма ИАД, в заданной распределенной среде, для анализа определенного набора данных.

СПИСОК ЛИТЕРАТУРЫ

1. Laney D. 3-D Data Management: Controlling Data Volume, Velocity and Variety // META Group Research Note, Stamford. META Group Inc. 6 Febr. 2001.

2. Ashton K. That 'Internet of Things' Thing. In the real world, things matter more than ideas. RFID J. URL: http://www.rfidjournal.com/articles/pdf?4986 (18.10.2016).

3. Dean J., Ghemawat S. MapReduce: Simplified data processing on large clusters // Proc. of Operating Systems Design and Implementation. San Francisco, CA., 2008. Vol. 51. P. 107–113.

4. Machine learning in apache spark / X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu,

J. Freeman, D. Tsai, M. Amde, S. Owen, D. Xin // Microtome Publishing. 2015. Vol. 17. P. 1–7.

5. Ingersoll G. Introducing apache mahout. Scalable, commercial friendly machine learning for building intelligent applications. URL: http://www.ibm.com/developerworks/library/j-mahout/j-mahout-pdf.pdf (18.10.2016).

6. Mell P., Grance T. The NIST Definition of Cloud Computing. Recommendations of the National Institute of Standards and Technology / Computer Security Division Information Technology Laboratory National Institute of Standards and Technology. Gaithersburg, 2011.

7. Gorlatch S. Extracting and implementing list homomorphisms in parallel program development // Science of Computer Programming. 2015. Vol. 33, iss. 1. P. 1–27.

8. Fog computing and its role in the internet of things / F. Bonomi, R. Milito, J. Zhu, S. Addepalli // Proc. of MCC, Helsinki, Finland, 17 Aug. 2012. P. 13–16.

9. Hewitt C., Bishop P., Steiger R. A universal modular ACTOR formalism for artificial intelligence // 3rd Intern. joint conf. on Artificial intelligence, Stanford, California, 1973. P. 235–245.

10. Kholod I., Petuhov I., Kapustin N. Creation of data mining cloud service on the actor model // NEW2AN/ruSMART.LNCS. 2015. Vol. 9247. P. 585–598.

11. Kholod I., Petuhov I. Creation of Data Mining Algorithms as Functional Expression for Parallel and Distributed Execution. In: Malyshkin, V. (eds.) // Parallel Computing Technologies. LNCS. 2015. Vol. 9251. P. 62–68.

I. I. Holod, I. V. Petuhov
Saint Petersburg Electrotechnical University «LETI»

METHOD FOR ESTIMATION OF DATA ANALYSIS EFFICIENCY IN A DISTRIBUTED ENVIRONMENT

Describes the method of configuring a network of actors to perform data mining in a distributed environment. Source data for the method are the characteristics of the data set, the algorithm analysis and runtime. Results of applying the method are the number of clusters, networks of actors, the number of different types of actors, as well as the placement of actors is determined at runtime.

Data mining, distributed analysis, distributed systems, actors model

УДК 004.622

Е. К. Лепилкина, С. И. Якушкин
Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В. И. Ульянова (Ленина)

Создание инфраструктуры сбора данных о статических характеристиках программ для задачи подбора оптимизационных опций для компиляторов на базе LLVM

Рассматривается вопрос о способе сбора и хранения статических характеристик отдельных базовых блоков программ до и после выполнения различных оптимизаций для компиляторов, разработанных на основе системы LLVM. Также рассматривается возможность дальнейшего доступа к этим данным для анализа эффективности работы компиляторов.

Статические характеристики, компилятор, система LLVM, Elasticsearch, оптимизационные проходы компилятора

Современные компиляторы представляют собой достаточно сложные системы с множеством опций, позволяющих получать высокопроизводительный машинный код. Все существующие компиляторы предоставляют большие возможности по оптимизации программ, однако зачастую подобрать опции компиляции достаточно сложно, так как необходимы данные о компилируемой программе, причем после работы большинства проходов эти сведения могут сильно измениться