Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САРАТОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО»

Кафедра математической
кибернетики и компьютерных наук

ПОСТРОЕНИЕ СИСТЕМЫ АНАЛИЗА ЛОГОВ С ИСПОЛЬЩОВАНИЕМ ТЕХНОЛОГИИ MAPREDUCE

КУРСОВАЯ РАБОТА

Студента 4 курса 411 группы
направления 010501.65 — Прикладная математика и информатика
факультета КНиИТ
Коновалова Никиты Алексеевича

Научный руководитель
доцент, к. ф.-м. н. _____ А. С. Иванова

Заведующий кафедрой
к.ф.-м.н. _____ А. С. Иванов

Саратов 2013

Содержание

ВВЕДЕНИЕ

В настоящее время возникает множество задач обработки больших массивов данных. Как правило, стандартные решения для хранения и обработки таких данных неприменимы в силу того, что они рассчитаны на работу внутри одной вычислительной системы. Современные сервера способны обеспечить высокую производительность вычислений, но какова бы она ни была, одной вычислительной системы будет не досстаточно для работы с большими данными.

Существуют различные технологии получения, хранения и обработки данных, рассчитанные на работу одновременно нескольких вычислительных систем(серверов). Такие технологии называют распределенными.

Важно заметить, что распределенные системы обладают большей отказоустойчивостью, так как хранимые данные могут быть реплицированы, а выполнение вычислений при отказу одного из серверов модет быть передано другому.

Одной из распределенных технологий обработки больших данных является технология MapReduce.

Целью данной работы является построение системы обработки текстовых лог-файлов на основе технологии MapReduce. Задачами можно считать следующие требования к системе:

— система должна иметь возможность обработать сколь угодно большой объем лог-файлов;
— система должна быть масштабируема

1 Технология MapReduce

1.1 Высокоуровневый интерфейс

Технология MapReduce была представлена Google в 2004 году. [1] В докладе описывался программный интерфейс, предоставляемый пользователю, а так же внутрениие механизмы системы.

Предполагается, что все данные, подлежащие обработке можно представить в виде списка пар (ключ, значение). От пользователя требуется реализовать две функции:

— $map(key,\ value) -> list(out\_key,\ intermediate\_value)$
— $reduce(out\_key,\ list(intermediate\_value)) -> out\_value$

Функция *map* предназначена для преобразования сырых данных в данные, пригодные для обработки. На данном этапе одна исходная пара може породить список промежуточных пар. *Map* будет вызвана строго один раз для каждой исходной пары.

На вход функции *reduce* поступают пары вида
$(out\_key,\ list(intermediate\_value))$.

Здесь данные группируются по ключам, полученным на выходе *map*. На данном этапе для каждого ключа доступен список сразу всех значений. На выходе для каждого ключа будет получено итоговое значение.

Данное описание дает лишь высокоуровневое представление о процессе обработки данных. Данная модель никак не указывает на то, откуда получаются исходные данные, как формируются списки промежуточных значений и т.д.

Разбиение обработки на два этапа таким образом, дает возможность работать параллельно нескольким *map* функциям и *reduce* функциям. Важно лишь то, что перед вызовом *reduce* должны завершиться все вызовы *map*, так как на вход *reduce* должен поступить список всех значений для конкретного ключа.

1.2 Пример MapReduce

В оригинальном докладе был приведен пример программы на основе MapReduce. Программа получала набор текстовых документов и выводила для каждого слова количество его вхождений во все документы.

```
map(String input_key, String input_value):
    for each word w in input_value:
        EmitIntermediate(w, "1");


reduce(String output_key, Iterator intermediate_values):
    int result = 0;
    for each v in intermediate_values:
        result += ParseInt(v);
    Emit(AsString(result));
```

Приведенный пример показывает реализацию функций *map* и *reduce* на псевдокоде. Функция *map* принимает документы в качестве своего второго аргумента - значения. Ключ в данном случае не важен. Цикл проходит по всем словам документа и создает промежуточные пары, в которых ключ - это слово из документа, а значение единица, означающая одно вхождение слова-ключа.

На вход *reduce* будут поданы парв, в которых ключ - слово, а значение - список чисел, обозначающих количество вхождений этого слова. В приведенной реализации список значений будет состоять из единиц. Цикл складывает полученные значения и получает окончательное количество входжений слова-ключа.

Данная реализаци не является наиболее эффективной и обладает рядом недостатков. Функция *map* порождает достаточно большое количество промежуточных пар и не проводит никакого анализа относительно самих слов. Например, слово в различных морфологических формах будет воспринято как несколько различных слов. Проблемы такого рода могут быть решены для каждой конкретной задачи и не представляют интереса в данном случае.

Сильной же стороной явлеяется то, что из кода видно, что каждый документ обрабатывается отдельным вызовом *map*. Таким образом, чем больше одновременно исполняется функций *map*, тем больше документов обрабатывается параллельно. Такая система масштабируется простым увеличением количества параллельных процессов.

2 Открытая реализация MaprReduce - Apache Hadoop

Hadoop — проект фонда Apache Software Foundation. Проект Hadoop включает в себя набор библиотек и интерфейсов для организации распределенного зранения и обработки данных. Некоторые из этих библиотек стали самостоятельными проектами Apache, такие как: Zookeeper, Hive, Pig, HBase. Hadoop написан на языке Java и предоставляет программные интерфейсы на этом языке. На данный момент стабильной версией является версия Hadoop 1.1.1, выпущенная в декабре 2012 года.

Так как Hadoop представляет собой достаточно сложную систему компонентов, существует несколько проектов, позволяющих упростить процесс установки, настройки и мониторинга Hadoop кластеров. Наиболее известные из них:

— Apache Ambari
— Cloudera Manager
— Intel Hadoop Distribution

2.1 Копмоненты Apache Hadoop

Основными компонентами Apache Hadoop являются:

— Hadoop MapReduce
— HDFS
— Hadoop Common

2.1.1 Компонент Hadoop MapReduce

Компонент Hadoop MapReduce отвечает непосредственно за исполнение функций *map* и *reduce*, а также всех вспомогательных операций. Схерма работы Hadoop MapReduce представлена на схеме 1.

На приведенной схеме показано, что Hadoop получает данные из локальной файловой системы того сервера, где запущен процесс *Mapping process*. Далее *Mapping process* вызывает функцию *map* на прочитанных данных и генерирует набор промежуточных пар.

Далее $shuffle\ process$ перераспределяет промежуточные пары между серверами так, чтобы пары с одинаковым ключем оказались на одном сервере. Даллее на этих же серверах запускаются процессы *Reducing process*.

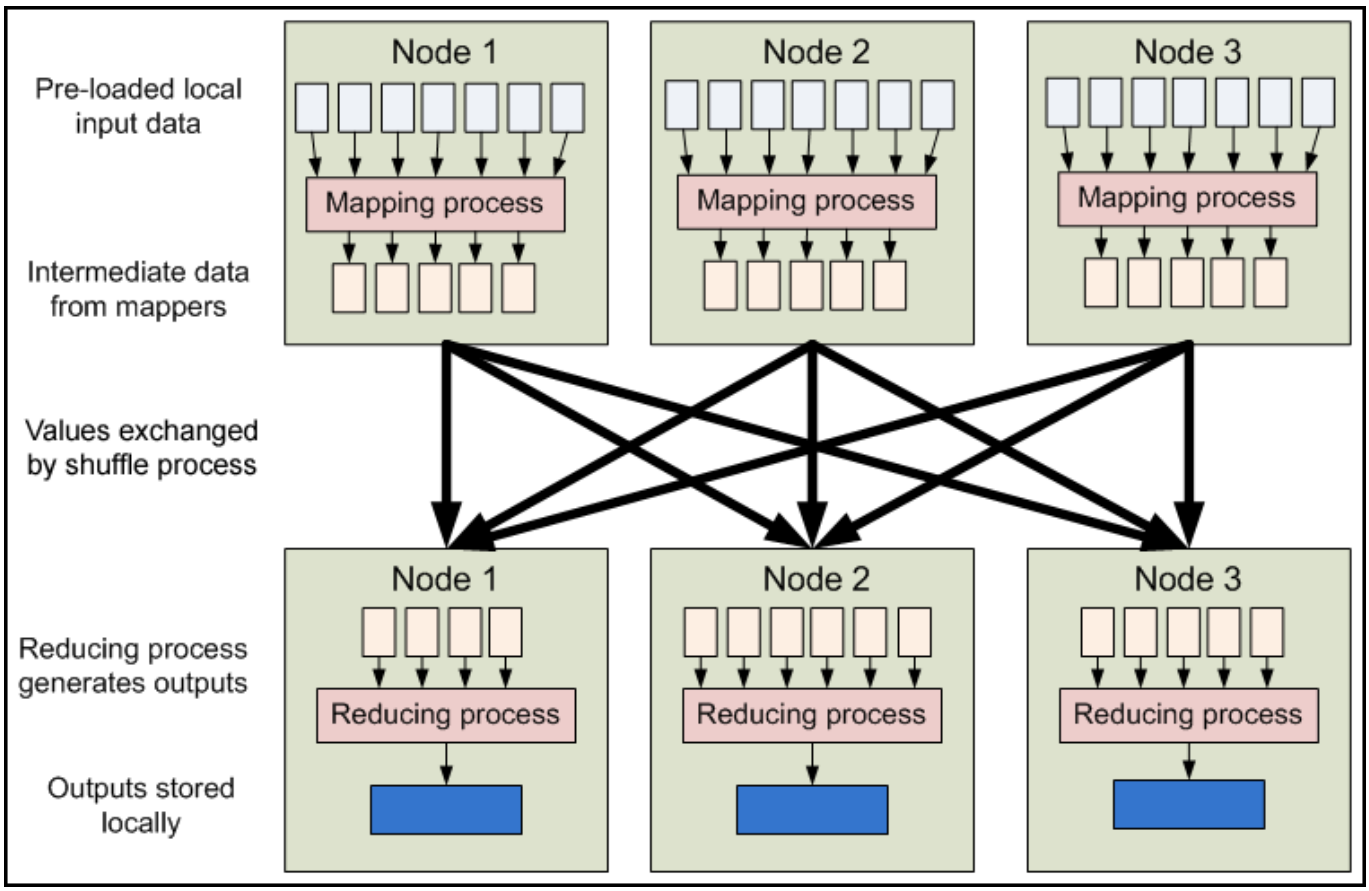

Рисунок 1 – MapReduce на 3-х серверах

Эти процессы вызывают функцию *reduce* на промежуточных парах. По завершении *reduce* полученные данные сохраняются в локальную файловую систему.

2.2 Еще элементы математического текста

Нейрон является составной частью нейронной сети. Он состоит из элементов трех типов: умножителей (синапсов), сумматора и нелинейного преобразователя. Синапсы осуществляют связь между нейронами, умножают входной сигнал на число, характеризующее силу связи (вес синапса). Сумматор выполняет сложение сигналов, поступающих по синаптическим связям от других нейронов, и внешних входных сигналов. Нелинейный преобразователь реализует нелинейную функцию одного аргумента — выхода сумматора. Эта функция называется функцией активации или передаточной функцией. На рисунке 2 приведено строение одного нейрона.

Нейрон в целом реализует скалярную функцию векторного аргумента.

Математическая модель нейрона:

$$s = \sum_{i=1}^{n} w_i x_i + b,$$

$$y = f(s),$$

где w_i — вес синапса; $i = 1, \ldots, n$; b — значение смещения; s — результат суммирования; x_i — i-тый компонент входного вектора (входной сигнал), $i = 1, \ldots, n$; y — выходной сигнал нейрона; n — число входов нейрона; $f(s)$ — нелинейное преобразование (функция активации).

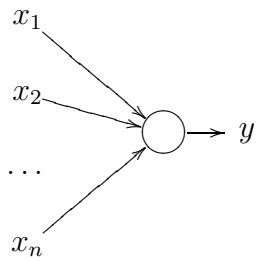

Рисунок 2 – Нейрон

В качестве функции активации нейронов берут обычно одну из следующих:

— пороговая функция активации;
— экспоненциальная сигмоида;
— рациональная сигмоида;
— гиперболический тангенс.

Данные функции активации обладают таким важным свойством как нелинейность. Нелинейность функции активации принципиальна для построения нейронных сетей. Если бы нейроны были линейными элементами, то любая последовательность нейронов также производила бы линейное преобразование и вся нейронная сеть была бы эквивалентна одному нейрону (или одному слою нейронов в случае нескольких выходов). Нелинейность разрушает суперпозицию и приводит к тому, что возможности нейросети существенно выше возможностей отдельных нейронов.

2.3 Снова математический текст

Опишем самую популярную архитектуру — многослойный персептрон с последовательными связями и сигмоидальной функцией активации (Feedfor-

ward Artifitial Neural Network, FANN).

В многослойных нейронных сетях с последовательными связями нейроны делятся на группы с общим входным сигналом — слои. Стандартная сеть состоит из L слоев, пронумерованных слева направо. Каждый слой содержит совокупность нейронов с едиными входными сигналами. Внешние входные сигналы подаются на входы нейронов входного слоя (его часто нумеруют как нулевой), а выходами сети являются выходные сигналы последнего слоя. Кроме входного и выходного слоев в многослойной нейронной сети есть один или несколько скрытых слоев, соединенных последовательно в прямом направлении и не содержащих связей между элементами внутри слоя и обратных связей между слоями. Число нейронов в слое может быть любым и не зависит от количества нейронов в других слоях. Архитектура нейронной сети прямого распространения сигнала приведена на рисунке 3.

На каждый нейрон первого слоя подаются все элементы внешнего входного сигнала. Все выходы нейронов i-го слоя подаются на каждый нейрон слоя $i + 1$.

Нейроны выполняют взвешенное суммирование элементов входных сигналов. К сумме прибавляется смещение нейрона. Над результатом суммирования выполняется нелинейное преобразование — функция активации (передаточная функция). Значение функции активации есть выход нейрона. Приведем схему многослойного персептрона. Нейроны представлены кружками, связи между нейронами — линиями со стрелками.

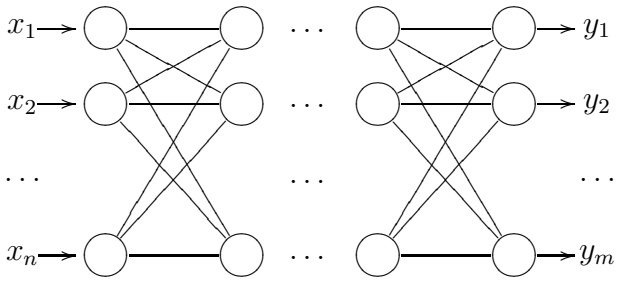

Рисунок 3 – Архитектура многослойной сети прямого распространения

Функционирование сети выполняется в соответствии с формулами:

$$s_j^{[k]} = \sum_{i=1}^{N_{k-1}} w_{ji}^{[k]} y_i^{[k-1]} + b_j^{[k]}, \quad j = 1, \ldots, N_k, \quad k = 1, \ldots, L;$$

$$y_j^{[k]} = f(s_j^{[k]}), \quad j = 1, \ldots, N_k, \quad k = 1, \ldots, L-1,$$
$$y_j^{[L]} = s_j^{[L]},$$

где

- $y_i^{[k-1]}$ — выходной сигнал i-го нейрона $(k-1)$-го слоя;
- $w_{ji}^{[k]}$ — вес связи между j-м нейроном слоя $(k-1)$ и i-м нейроном k-го слоя;
- $b_j^{[k]}$ — значение смещения j-го нейрона k-го слоя;
- $y = f(s)$ — функция активации;
- $y_j^{[k]}$ — выходной сигнал j-го нейрона k-го слоя;
- N_k — число узлов слоя k;
- L — общее число основных слоев;
- $n = N_0$ — размерность входного вектора;
- $m = N_L$ — размерность выходного вектора сети.

На рисунке 4 представлена сеть прямого распространения сигнала с 5 входами, 3 нейронами в скрытом слое и 2 нейронами в выходном слое.

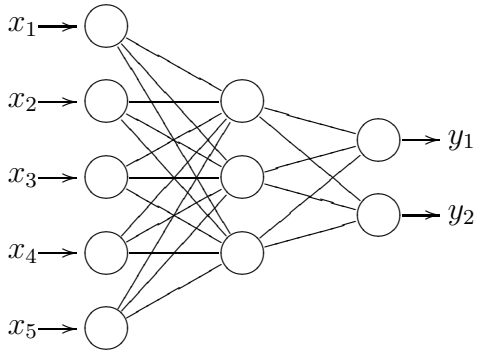

Рисунок 4 – Пример нейронной сети

3 Раздел с подразделами

3.1 Текст с формулами и леммой

Обозначим $[y_0, y_1, \ldots, y_p; f]$ разделенную разность порядка p функции f по узлам $y_0 < y_1 < \ldots < y_p$.

Обозначим $L_p f(x; y_0, y_1, \ldots, y_p)$ интерполяционный полином Ньютона функции f по узлам $y_0, y_1, \ldots, y_p$:

$$L_p f(x; y_0, y_1, \ldots, y_p) = \sum_{j=0}^{p} [y_0, \ldots, y_j; f] \cdot \prod_{i=0}^{j-1} (x - y_i), \quad x - y_{-1} \stackrel{\text{def}}{=} 1 \tag{1}$$

Лемма 1. Если $0 \leqslant x_0 < x_1 < \ldots < x_p \leqslant 1$ и $f \in C[0,1]$ удовлетворяет условиям

1. $f(x) \geqslant 0, \ x \in [0,1]$;
2. $[y_0, \ldots, y_{p+1}; f] \geqslant 0$ для всех $y_i \in [0,1], \ i = 0, \ldots, p+1$,

тогда

$$L_p f(x; x_0, \ldots, x_p) \geqslant 0 \tag{2}$$

для всех $x \in [x_{p-(2k+1)}, x_{p-2k}]$, $k = 0, \ldots, [p/2]$, $x_{-1} \stackrel{\text{def}}{=} -\infty$.

Доказательство. Возьмем $x \in [x_{p-(2k+1)}, x_{p-2k}], \quad k = 0, \ldots, [p/2]$.

Из условия 1 леммы следует, что

$$[x_0, \ldots, x_{p-(2k+1)}, x, x_{p-2k}, \ldots, x_p; f] \geqslant 0,$$

т. е.

$$\Delta_p f(x; x_0, \ldots, x_p) \stackrel{\text{def}}{=} \stackrel{\text{def}}{=} \begin{vmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^p & f(x_0) \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & x_{p-(2k+1)} & x_{p-(2k+1)}^2 & \cdots & x_{p-(2k+1)}^p & f(x_{p-(2k+1)}) \\ 1 & x & x^2 & \cdots & x^p & f(x) \\ 1 & x_{p-2k} & x_{p-2k}^2 & \cdots & x_{p-2k}^p & f(x_{p-2k}) \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & x_p & x_p^2 & \cdots & x_p^p & f(x_p) \end{vmatrix} \geqslant 0 \tag{3}$$

Из равенства

$$\Delta_p f(x; x_0, \ldots, x_p) = (L_p f(x; x_0, \ldots, x_p) - f(x)) \prod_{0 \leqslant i < j \leqslant p} (x_j - x_i).$$

и (3) следует, что

$$L_p f(x; x_0, \ldots, x_p) \geqslant f(x).$$

С учетом условия 1 леммы мы получаем утверждение (2). □

3.2 Название другого подраздела

3.2.1 Более мелкий подраздел

Если разность энергий электронно-дырочных уровней $E_2 - E_1$ близка к энергии предельного оптического фонона $\hbar\Omega_{LO}$, то в разложении волновых функций полного гамильтониана можно ограничиться нулевым приближением для всех состояний, за исключением близких по значению к E_2.

3.2.2 Текст с таблицей

В таблице 1 представлены результаты сокращения словарей неисправностей для схем из каталога ISCAS'89.

Таблица 1 – Результат сокращения словарей неисправностей при помощи масок

1	2	3	4	5	6	7	8
S298	177	1932	341964	61	10797	3,16%	0,61
S344	240	1397	335280	59	14160	4,22%	0,53
S349	243	1474	358182	62	15066	4,21%	0,60
S382	190	12444	2364360	55	10450	0,44%	3,78
S386	274	2002	548548	91	24934	4,55%	1,40
S400	194	13284	2577096	58	11252	0,44%	4,28
S444	191	13440	2567040	60	11460	0,45%	4,26
S510	446	700	312200	70	31220	10,00%	0,63
S526	138	13548	1869624	38	5244	0,28%	2,41
S641	345	5016	1730520	132	45540	2,63%	7,06
S713	343	3979	1364797	131	44933	3,29%	5,61
S820	712	21185	15083720	244	173728	1,15%	126,99
S832	719	21603	15532557	253	181907	1,17%	135,18
S953	326	322	104972	91	29666	28,26%	0,27
S1423	293	750	219750	93	27249	12,40%	0,57
S1488	1359	22230	30210570	384	521856	1,73%	541,69

3.2.3 Текст с кодом программы

Термин «разреженная матрица» впервые был предложен Гарри Марковицем. В 1989 он был награжден премией имени Джона фон Неймана в том числе и за вклад в теорию методов для разреженных матриц.

В большинстве источников, разреженной матрицей называется матрица, в которой мало ненулевых элементов. Это нельзя назвать определением из-за слова «мало». В [3] понятие разреженной матрицы определяется так: «Мы можем называть матрицу разреженной, если применение к ней методов, описываемых в книге, экономит память и/или время». Таким образом, следует дать определение алгоритму для разреженных матриц. Алгоритмом для разреженных матриц будем называть алгоритм, у которого время работы и необходимый объем памяти зависят от количества ненулевых элементов в матрице.

Размерность квадратной матрицы A будем обозначать n, а количество ненулевых элементов в ней $|A|$.

Плотные матрицы обычно хранятся в качестве двумерного массива $n \times n$. Будем обозначать такой массив a. Разреженные матрицы не стоит хранить таким способом из-за слишком большого потребления памяти, которая будет занята в основном нулевыми элементами.

Один из вариантов представления разреженных матриц в памяти компьютера — в виде трех массивов: `column`, `value` и `rowIndex`. Размеры массивов `column` и `value` равны $|A|$. Размер `rowIndex` равен $n + 1$. Ненулевые элементы матрицы A хранятся последовательно по строкам в этих массивах. Элемент `column[i]` содержит номер столбца, в котором содержится `i`-й ненулевой элемент, а `value[i]` — его величину. Массив `rowIndex[i]` содержит в себе индекс первого ненулевого элемента `i`-й строки. Все ненулевые элементы `i`-й строки содержатся в массивах `column` и `value` в элементах с индексами от `rowIndex[i]` по `rowIndex[i + 1]-1`. Для удобства полагают `rowIndex[`n`]`$= |A|$.

Для примера рассмотрим следующую матрицу:

$$\begin{pmatrix} 1 & 0 & 5 & 0 & 0 \\ 0 & 2 & 7 & 4 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 9 & 6 & 0 & 3 & 0 \\ 0 & 0 & 3 & 0 & 5 \end{pmatrix}$$

Массивы `column`, `value` и `rowIndex` для этой матрицы представлены в таблице 2.

Таблица 2 – Массивы `column`, `value` и `rowIndex`

	0	1	2	3	4	5	6	7	8	9	10	11
`column`	0	2	1	2	3	2	0	1	3	2	4	
`value`	1	5	2	7	4	1	9	6	3	3	5	
`rowIndex`	0	2	5	6	9	11						

Неизвестный вектор и вектор правой части хранятся в виде массивов размера n. Массив неизвестного вектора обозначают `x`, а массив правой части — `rhs`.

Рассмотрим пример алгоритма для разреженных матриц. Алгоритм решения СЛАУ, представленной нижнетреугольной матрицей `a`, можно реализовать двумя вложенными циклами по `n`:

```
for(int i = 0; i $<$ n; ++i){
    x[i] = rhs[i];
    for(int j = 0; j $<$ i; ++j)
        x[i] -= a[i][j] * x[j];
    x[i] /= a[i][i];
}
```

Но, если матрица `a` хранится в разреженном виде, то в данном алгоритме можно проходить только по ненулевым элементам `a`:

```
for(int i = 0; i $<$ n; ++i){
    x[i] = rhs[i];
    for(int j = rowIndex[i]; j $<$ rowIndex[i + 1] - 1; ++j)
        x[i] -= value[j] * x[column[j]];
    x[i] /= value[rowIndex[i + 1] - 1];
}
```

В первом случае оценка времени работы будет $O(n^2)$, а во втором $O(|A|)$.

Методы для разреженных матриц основаны на следующих главных принципах [3]:

1. Хранятся только ненулевые элементы матрицы.
2. Выполняются только те преобразования, которые действительно что-то изменяют. В примере не имеет смысла вычитать из `x[i]` значение `x[j]*a[i][j]`, если `a[i][j]` равно нулю.
3. Число «новых элементов», возникающих, например, во время исключения Гаусса, стараются уменьшить путем перестановок строк и столбцов матрицы.

ЗАКЛЮЧЕНИЕ

В настоящей работы приведен пример оформления студенческой работы средствами системы LaTeX.

Показано, как можно оформить документ в соответствии:

— с правилами оформления курсовых и выпускных квалификационных работ, принятых в Саратовском государственном университете в 2012 году;

— с правилами оформления титульного листа отчета о прохождении практики в соответствии со стандартом.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Mapreduce: Simplified data processing on large clusters. http://research.google.com/archive/mapreduce.html.

2 Скворцов, . . Флуктуационные и интерференционные эффекты в мезоскопических системах. — 2008.

3 Эстербю, . Прямые методы для разреженных матиц / . Эстербю, . Златев. — М.: Мир.

ПРИЛОЖЕНИЕ А

Нумеруемые объекты в приложении

Таблица А.1 – Results of pass-fail dictionary reduction with the help of masks

Circuit	Number of modelled faults	Number of test vectors in the test set	The volume of pass-fail dictionary, bit	The volume of found mask	The volume of masked dictionary, bit	% of pass-fail dictionary	CPU running time, min
S298	177	322	56994	30	5310	9,32%	0,07
S344	240	127	30480	29	6960	22,83%	0,04
S349	243	134	32562	35	8505	26,12%	0,05
S382	190	2074	394060	28	5320	1,35%	0,43
S386	274	286	78364	65	17810	22,73%	0,26
S400	194	2214	429516	32	6208	1,45%	0,99
S444	191	2240	427840	30	5730	1,34%	0,98
S526	138	2258	311604	28	3864	1,24%	0,61
S641	345	209	72105	58	20010	27,75%	0,24
S713	343	173	59339	58	19894	33,53%	0,19
S820	712	1115	793880	147	104664	13,18%	9,09
S832	719	1137	817503	151	108569	13,28%	9,20
S953	326	14	4564	13	4238	92,86%	0,01
S1423	293	150	43950	58	16994	38,67%	0,15
S1488	1359	1170	1590030	158	214722	13,50%	26,69

$$F(x) = \int_a^b f(x)\,dx. \tag{A.1}$$

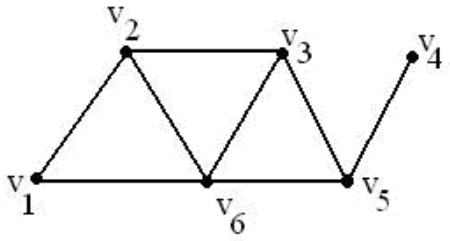

Рисунок А.1 – Подпись к рисунку

Таблица А.2

0	1
1	0

ПРИЛОЖЕНИЕ Б

Листинг программы

Код приложения `task.pl`.

```
use locale;
use encoding "cp866";
{
     print "Имя папки: "; my $folder_name=<>;
     chomp($folder_name);
     my @files = `chcp 866 & attrib $folder_name\\*.pl`;
     if (substr($files[1],0,15) eq 'Не найден путь:') {
          print "Путь не найден. Попробуйте еще.\n";
          redo;
     }
     elsif (substr($files[1],0,15) eq 'Не найден файл:') {
          print "Папка не содержит файлов .txt .\n";
          last;
     }
     else {
          foreach my $file (@files[1 .. $#files]){
               my $file_name = substr($file, 11);
               chomp($file_name);
               open(FH,"<$file_name") or die $!;
               my %hash = ();
               foreach $chunk (<FH>){
                    my @words = $chunk =~ /([\@\%\$][a-zA-Z_0-9]+[\[\{]?)/g;
                    foreach my $word (@words) {
                         $word = "\$".substr($word, 1)
                            if (substr($word, 0, 1) eq '@' &&
                                substr($word, -1) eq '[');
                         $word = "\$".substr($word, 1)."["
                            if (substr($word, 0, 1) eq '@');
                         $word = "\$".substr($word, 1)."{"
                            if (substr($word, 0, 1) eq '%');
                         $hash{$word}++;
                    };
               };
               my @xs = keys %hash;
               print @xs;
               close(FH);
```

```
37                 my $ans = scalar(@xs);
38                 print "$file_name : $ans\n";
39             }
40         }
41 }
```

ПРИЛОЖЕНИЕ В

Многостраничная таблица

Таблица В.1 – ГОСТ DIN ISO — Таблица соответствия стандартов

Стандарт ГОСТ	Наименование	Стандарт DIN	Стандарт ISO
1	2	3	4
ГОСТ 397-79	Шплинты	DIN 94	ISO 1234
ГОСТ 1144-80	Шурупы с полукруглой головкой	DIN 96DIN 7981	ISO 7049
ГОСТ 1145-80	Шурупы с потайной головкой	DIN 97DIN 7982	ISO 7050
ГОСТ 1146-80	Шурупы с полупотайной головкой	DIN 95DIN 7983	ISO 7051
ГОСТ 1476-93	Винты установочные с коническим концом и прямым шлицем классов точности А и В	DIN 553	ISO 7434
ГОСТ 1477-93	Винты установочные с плоским концом и прямым шлицем классов точности А и В	DIN 438DIN 551	ISO 4766ISO 7436
ГОСТ 1478-93	Винты установочные с цилиндрическим концом и прямым шлицем классов точности А и В	DIN 417	ISO 7435
ГОСТ 1481-84	Винты установочные с шестигранной головкой и цилиндрическим концом классов точности А и В	DIN 561	
ГОСТ 1482-84	Винты установочные с квадратной головкой и цилиндрическим концом классов точности А и В	DIN 479	

Продолжение таблицы B.1

1	2	3	4
ГОСТ 1485-84	Винты установочные с квадратной головкой и засверленным концом классов точности А и В	DIN 479	
ГОСТ 1486-84	Винты установочные с квадратной головкой и ступенчатым концом со сферой классов точности А и В	DIN 480	
ГОСТ 1488-84	Винты установочные с квадратной головкой и буртиком классов точности А и В	DIN 478	
ГОСТ 1491-80	Винты с цилиндрической головкой классов точности А и В	DIN 84	ISO 1207
ГОСТ 3032-76	Гайки-барашки	DIN 315	
ГОСТ 3033-79	Болты откидные	DIN 444	
ГОСТ 3057-90	Пружины тарельчатые	DIN 2093	
ГОСТ 3070-88	Канат стальной двойной свивки типа ТК конструкции 6х19 (1+6+12)+1 о.с.	DIN 3060	
ГОСТ 3128-70	Штифты цилиндрические незакаленные	DIN 7DIN 6325	ISO 2338ISO 8734
ГОСТ 3129-70	Штифты конические незакаленные	DIN 1	ISO 2339
ГОСТ 4751-73	Рым-болты	DIN 580	ISO 3266
ГОСТ 5915-70	Гайки шестигранные стальные класса точности В	DIN 555DIN 934	ISO 4032ISO 4033ISO 8673ISO 8674

Продолжение таблицы B.1

1	2	3	4
ГОСТ 5916-70	Гайки шестигранные низкие класса точности В	DIN 439DIN 936	ISO 4035ISO 4036ISO 8675
ГОСТ 5918-73	Гайки шестигранные прорезные и корончатые класса точности В	DIN 935	EN ISO 7035EN ISO 7036EN ISO 7037
ГОСТ 5919-73	Гайки шестигранные прорезные и корончатые низкие класса точности В	DIN 937	EN ISO 7038
ГОСТ 5927-70	Гайки шестигранные класса точности А	DIN 555DIN 934	ISO 4032ISO 4034ISO 8673
ГОСТ 5932-73	Гайки шестигранные прорезные и корончатые класса точности А	DIN 935DIN 937	EN ISO 7035EN ISO 7036EN ISO 7037
ГОСТ 6393-73	Гайки круглые с отверстиями на торце "под ключ"класса точности А	DIN 1816	
ГОСТ 6402-70	Шайбы пружинные	DIN 127	
ГОСТ 6958-78	Шайбы увеличенные. Классы точности А и С	DIN 440DIN 9021	ISO 7094ISO 7093-1ISO 7093-2
ГОСТ 7786-81	Болты с потайной головкой и квадратным подголовком класса точности С	DIN 608	
ГОСТ 7795-70	Болты с шестигранной уменьшенной головкой и направляющим подголовком, класс точности В		