ISBN 5-94052-066-0 ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ. ПЕРЕСЛАВЛЬ-ЗАЛЕССКИЙ, 2004

УДК 519.68

В. А. Васенин, С. А. Афонин, А. С. Козицын, А. С. Шундеев

Поиск в сверхбольших хранилищах данных и высокопроизводительные системы с массовым параллелизмом

Аннотация. Одной из важных задач на современном этапе развития информационных технологий является задача релевантного поиска в больших объемах разноструктурированной информации. В работе рассматривается новая система тематического поиска и поиска с учетом структуры документа. Система была разработана с использованием массово-параллельных вычислительных комплексов на платформе «СКИФ» и программных систем, поддерживающих автоматическое динамическое распараллеливание. В работе описаны алгоритмы работы системы, приведены результаты тестирования распараллеливания и сформулированы основные выводы.

Ключевые слова и фразы: тематический поиск, структура документа, динамическое распараллеливание.

1. Введение

Одной из стратегически важных для общества на современном этапе развития информационных технологий является задача оперативного и релевантного поиска научной информации в больших и сверхбольших (крупные корпоративные системы или Интернет в целом) объемах объективно разноструктурированной информации. От решения этой задачи во многом зависит эффективность решений, принимаемых на уровне корпораций, государственных ведомств, а значит, и результаты деятельности отдельных отраслей национального хозяйственного комплекса, темпы развития экономики. Постановка и значимость данной задачи, условия и результаты ее решения прямо зависят не только от адекватности используемых математических моделей, но и от возможности средств вычислительной техники, средств телекоммуникаций и сетевых технологий. С учетом этого обстоятельства они эволюционировали столь же быстрыми, а с появлением сетей пакетной коммуникации и Интернет, — сверхвысокими темпами.

Работа выполнялась при частичной поддержке суперкомпьютерной программы «СКИФ» Союзного государства.

В данной публикации рассмотрим задачу в ее сегодняшней постановке, с учетом сложившихся и перспективных подходов к ее решению, в том числе, — с изменением современных высокопроизводительных систем с массовым параллелизмом. Часть результатов, изложенных ниже, получены в ходе работ по российско-белорусской программе «СКИФ» [1]. В настоящее время наиболее распространенным является подход к организации поиска, основанный на ключевых словах. Он широко используется на таких известных поисковых серверах, как Yandex, Google, Rambler. Широкое применение этого метода объясняется простотой реализации и высокой скоростью работы, поскольку он позволяет использовать заранее составленные индексы по словам или их нормальным формам. Однако во многих ситуациях такой поиск не дает желаемого результата. Либо интересующий нас документ не находится, либо находится такое количество «лишних» документов, что отыскать среди них нужный не представляется возможным. Допустим, нам требуются тексты по титановым сплавам. Введя слова «титановый» «сплав» мы не найдем многих документов, поскольку в профессиональных статьях часто вместо слов «титановый сплав» используется просто слово «титан». Введя же только слово «титан» мы получим огромное количество документов по всем темам, начиная от древних богов, и кончая рекламными проспектами фирм и организаций, имеющих в своем названии или названии товара слово «титан».

Для более эффективного поиска можно использовать методы тематического поиска и поиска с использованием структуры документов, а также комбинации этих методов. Идея последовательного взаимодополнения этих методов заключается в том, чтобы на первом шаге отсечь большую часть ненужных документов быстрыми методами поиска, например, поиском по ключевым словам или тематическим поиском, а на следующем шаге на оставшемся множестве провести поиск более точными, но менее быстрыми методами, например, поиском с учетом структуры документа.

Метод тематического поиска заключается в выделении документов, попавших в определенный пользователем кластер или рубрику. Множество таких кластеров или рубрик могут быть определены заранее при настройке и обучении системы или описываются пользователем в процессе работы. В настоящий момент широко используется только фиксированная рубрикация, и при этом, как правило,

разбиение документов на рубрики происходит вручную. Такой подход позволяет получить хорошее качество рубрикации на небольших объемах текстов, что, однако, совершенно неприменимо при работе с объемами документов в Интернет, поскольку потребует слишком много людских ресурсов.

Использование методов поиска с учетом структуры документа (поиск в полуструктурированных данных) [2] связано с развитием таких стандартов разметки научно-технических документации как MathML и DocBook. Подобная семантическая разметка позволяет расширять соответствующие языки запросов новыми конструкциями. В результате возрастает релевантность (качество) поиска.

В первой части настоящей работы представлен метод тематической рубрикации входящего потока документов. Предполагается, что на входе имеется достаточно «широкий» одноуровневый классификатор предметной области, заданный обучающими выборками. В качестве результата требуется распределить документы входящего потока по рубрикам в соответствии с их тематикой. Вторая часть работы посвящена проблеме выбора оптимальных планов выполнения запросов к полуструктурированным базам данных. Делается вывод о необходимости использования высокопроизводительных параллельных систем при приближенном вычислении оптимальных планов.

Для эффективной реализации, как первичного тематического поиска, так и поиска на втором этапе с учетом структуры документов использовались массово-параллельные вычислительные комплексы на платформе «СКИФ» (в ИПС РАН и НИИ механики МГУ им. М. В. Ломоносова) и программные системы, поддерживающие автоматическое динамическое распараллеливание в их эволюции от T/Grace системы [3] к T-системе с открытой архитектурой — Open TS в ее сегодняшней реализации.

2. Алгоритм рубрикации

Алгоритм рубрикации содержит два основных этапа. На первом этапе происходит подготовка исходных данных. Производится:

- морфологический анализ обучающих выборок;
- анализ терминов;
- построение статистических портретов каждой рубрики;
- построение дерева статистических портретов.

Этот этап — подготовительный и запускается только один раз перед началом работы алгоритма рубрикации. На втором этапе производится:

- морфологический анализ входящего потока текстов;
- анализ терминов входящего потока;
- осуществляется привязка текстов входного потока к заданным рубрикам на основе статистических портретов, т. е. рубрикация текстов.

Заметим, что для первого, подготовительного этапа скорость обработки не является критичной, поскольку он запускается только один раз для обучения системы. На втором, основном, этапе наибольший объем вычислений требуется для непосредственной рубрикации текстов. И, следовательно, именно этот процесс требует максимального ускорения, в том числе методами динамического распараллеливания. С учетом изложенных обстоятельств, далее коротко остановимся на принципах работы модулей тематического поиска, уделив более подробное внимание ресурсоемкому алгоритму рубрикации текстов.

2.1. Морфологический анализ. Блок морфологического анализа преобразует входной поток текстов в векторное представление, которое можно в дальнейшем использовать в различных математических моделях обработки текста. Существует и практически реализовано несколько вариантов подобного преобразования. В настоящей работе используется модель представления без учета порядка. Каждому документу сопоставляется вектор, размерность которого равна количеству известных системе лемм (леммой называют нормальную форму слова или словоформы, например, для существительного это именительный падеж, единственное число, для прилагательных мужской род, именительный падеж, единственное число). В качестве координаты берется число, показывающее, сколько раз данная лемма встретилась в анализируемом документе. Вместо широко распространенного метода поиска по таблицам основ и окончаний используется метод простого поиска по таблице словоформ. Целесообразность подобного подхода объясняется тем, что увеличение быстродействия в данном случае является более приоритетной задачей, чем минимизация требований к оперативной памяти на головном узле. Таким образом, сопоставление ключа леммы по словоформе производится поиском по отсортированному заранее массиву описания словоформ следующего вида: «ключ леммы» «словоформа».

2.2. Анализ терминов. Термин — это устойчивое множество слов, находящихся одно от другого в тексте на расстоянии меньше заданного. В реализованном алгоритме рассматривались пары слов с минимальным расстоянием равным единице. Анализ терминов позволяет улучшить точность описания документа при построении статистических портретов рубрик. Например, лемма «теория» встречается во множестве документов самых разных направлений. Лемма «граф» тоже может встречаться в исторических и литературных текстах. Однако, термин, состоящий из пары лемм, «теория граф» характеризует вполне определенное направление математики.

Анализ терминов осуществляется на основе файла описания терминов, в котором записаны соответствия ключа термина паре ключей лемм. Файл описания терминов строится автоматически на основе статистической информации о встречаемости пар слов в некоторой обучающей выборке.

2.3. Построение статистических портретов рубрик. Статистические портреты рубрики служат для описания вероятностных характеристик рубрики. Каждый статистический портрет начинается с заголовка, содержащего название рубрики, номер, априорную вероятность принадлежности произвольного документа данной рубрике и количество документов, заданных в обучающей выборке для описания рубрики. После этого идет описание лемм и терминов с указанием ключа, вероятности вхождения в документ рубрики и во внешний документ, встречаемость в рубрике и количество документов в рубрике, содержащих эту лемму или термин.

Статистические портреты элементов общего рубрикатора строятся на основе алгоритмов вероятностной рубрикации. Основная сложность, связанная с этим методом, состоит в выборе корректной обучающей выборки, включающей как документы из предметной области, так и вне ее. Поскольку предполагается, что общий рубрикатор является достаточно широким, то для оценки частоты встречаемости слов вне предметной области можно взять частотности слов в русском языке. Таким образом, обучающая выборка общего рубрикатора может содержать документы этой предметной области.

2.4. Построение дерева статистических портретов. Дерево статистических портретов рубрик, как и анализ терминов, требуется для улучшения качества рубрикации. Его построение осуществляется следующим образом.

Построенное на предыдущем шаге множество портретов рубрик преобразуется во множество точек многомерного пространства, координатами которых являются частоты встречаемости лемм и терминов. Затем каждая точка рассматривается как материальная точка единичной массы. В полученной системе рассматривается гравитационный или «псевдогравитационный» закон взаимодействия, который приводит к некоторому движению точек с постепенным сближением, слиянием друг с другом, и, в конечном счете, объединением в одну точку, находящуюся в центре масс начальной системы. Считается, что точки не обладают инерционностью. Это позволяет получить сходимость без введения параметров вязкости.

Процесс слияния точек может рассматриваться как дерево, кластеризующее множество статистических портретов рубрик. Близкие по смыслу рубрики, обладающие похожими статистическими портретами, будут объединяться раньше, далекие позже. Результаты кластеризации статистических портретов вместе с самими статистическими портретами используются как входные данные для рубрикации текстов.

2.5. Рубрикация текстов. Рубрикация текстов осуществляется вероятностным методом на основе формулы условной вероятности [4]. Изначально считается, что документ принадлежит данной рубрике с определенной априорной вероятностью, а затем каждая лемма, и каждый термин, встретившийся в документе, меняют эту вероятность согласно формуле условной вероятности.

Один из недостатков этого метода заключается в том, что приходится использовать гипотезу о независимости вхождений слов в документе, в том числе, повторных вхождений слов в один и тот же документ. Поскольку в реальном документе слова взаимозависимы, необходимо использовать различные способы компенсации взаимных зависимостей. В данном случае используется логарифмический учет повторной встречаемости слова в документе. Таким образом, если лемма или термин встретились в документе n раз, то они будут учтены с весом $1 + \log(n)$.

Второй серьезной проблемой данного метода является слияние близких рубрик в общем рубрикаторе. Если имеются две близких рубрики, например, «физика» и «математика», а также большое множество других рубрик, например, «психология», «история», «философия», непохожих на первые две, то система при поступлении документа не сможет отнести его только к физике или только к математике. Этот факт объясняется тем, что леммы и термины, которые могут позволить разделить физику и математику, будут «подавлены» множеством лемм и терминов, выделяющих физику и математику из основной массы рубрик. Данная проблема решается при помощи построенного ранее дерева статистических портретов рубрик.

Определение рубрики документа происходит спуском по дереву рубрик, начиная с его корня. Каждый узел дерева считается рубрикой, статистический портрет которого является объединением статистических портретов всех его листьев. На каждом шаге происходит выбор из нескольких равноценных, «равноудаленных» рубрик. Те рубрики, которые признаются подходящими для данного документа, помечаются и рассматриваются на следующем шаге. Процесс повторяется до тех пор, пока не достигнет листьев, то есть не будет осуществлена привязка к конкретным рубрикам, или пока количество выбраных узлов на очередном шаге не станет равным нулю. Это обстоятельство будет означать отсутствие подходящей рубрики для данного документа. Описанный подход позволяет избежать описанной выше проблемы слияния близких рубрик.

Заметим, что документ может пройти сразу по нескольким ветвям и попасть сразу в несколько рубрик. Например, текст о Ломоносове может попасть одновременно и в рубрику «история», и в рубрику «математика».

Из вычислительных проблем следует отметить следующие: наличие нулевых вероятностей встречаемости лемм в статистических портретах рубрик и малые значения участвующих в вычислении величин.

Для устранения нулевых и единичных вероятностей определяется некоторая малая вероятность d. После этого значения меньше d заменяются на равные d, а значения больше $(1 - d)$ заменяются на равные $(1 - d)$. Практически это означает, что даже если термин ни разу не встретился вне рубрики для данной обучающей выборки, он, все-таки, может встретиться в каком-нибудь документе, не принадлежащем рубрике, с вероятностью, равной d.

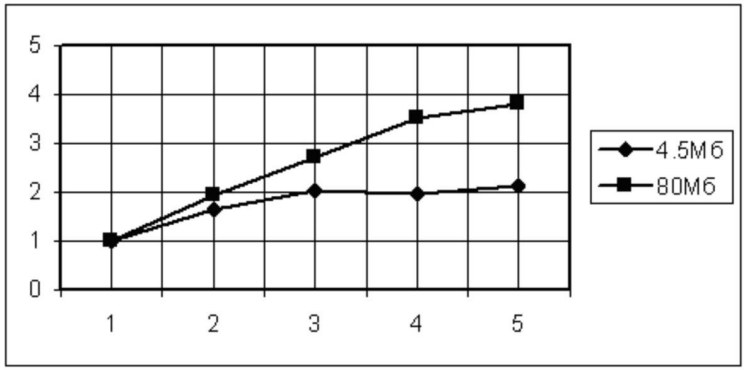

Рис. 1. Сравнение роста коэффициентов ускорения в зависимости от размера входного потока

Для устранения проблем с малыми величинами используется следующее свойство формулы условной вероятности. Если все вероятности принадлежности документа рубрикам умножить на одно и тоже ненулевое число, то результат формулы не изменится. Это позволяет при вычислениях на каждом шаге проводить нормализацию вероятностей для избежания малых величин.

В настоящий момент реализовано два алгоритма оценки принадлежности документа рубрике: оптимистичный, при котором считается, что документ принадлежит хотя бы одной рубрике, и пессимистичный, при которой с достаточно высокой степенью вероятности считается, что документ не принадлежит ни к одной из указанных рубрик.

В оптимистичном алгоритме считается, что вероятность встречаемости леммы вне рубрики j равна $P_j * (\frac{q_j}{d_j})$, а вероятность встречаемости леммы вне рубрики j равна $\Sigma(P_i * (\frac{q_i}{d_i}))$ по всем $i \neq j$.

В пессимистичном алгоритме считается, что вероятность встречаемости леммы вне рубрики j равна $(\frac{q_j}{d_j})$, а вероятность встречаемости леммы вне рубрики j равна $(\frac{\Sigma q_i}{\Sigma d_i})$ по всем $i \neq j$.

Здесь P_i — вероятность принадлежности документа i-ой рубрике, q_i — количество документов в i-ой рубрике, содержащих лемму, d_i — общее количество документов в i-ой рубрике.

4.5 Мб.					
количество узлов	1	2	4	8	12
узел 0	122	74	56	34	46
узел 1		98	48	22	40
узел 2			54	36	44
узел 3			36	32	18
узел 4				16	8
узел 5				38	2
узел 6				20	52
узел 7				26	2
узел 8					0
узел 9					0
узел 10					0
узел 11					0
80 Мб.					
узел 0	3051	2542	1178	697	702
узел 1		1781	1302	808	812
узел 2			1168	474	686
узел 3			1264	828	774
узел 4				808	570
узел 5				672	340
узел 6				764	174
узел 7				368	80
узел 8					158
узел 9					160
узел 10					184
узел 11					782

Таблица 1. Результаты загрузки узлов при входных потоках 4.5 Мб. и 80 Мб.

2.6. Результаты распараллеливания тематического поиска. В качестве гранулы для распараллеливания данной задачи выбрана функция вычисления вероятности принадлежности документа всем рубрикам. Это значит, что после морфологического анализа на вход функции подается векторное представление текста и статистические портреты кластеров, а на выходе ожидается вектор вероятностей принадлежности данного документа каждому из кластеров.

Заметим, что из-за неоднородности входящего потока текстов, как по объему, так и по сложности, времена вычисления гранул могут сильно отличаться. Это приводит к тому, что при использовании статического распараллеливания и распределении работы равномерно по количеству будет наблюдаться простой узлов, получивших «простые» задания и чрезмерная загрузка узлов, получивших «сложные» задания. Поэтому, для более эффективной работы в данной задаче использовались методы динамического распараллеливания.

Реализация была выполнена с использованием языка ТС и тестировалась на высокопроизводительном вычислительном комплексе семейства «СКИФ», установленном в Институте программных систем РАН в г. Переславле-Залесском.

В Таблице 1 и на Рис. 1 представлены результаты тестирования для двух потоков документов: небольшой поток 4.5 Мб и рабочий поток 80 Мб. Из приведенных ниже таблиц видно, что распараллеливание по узлам проходит достаточно эффективно, однако добиться линейного ускорения производительности не удается. Это объясняется, во-первых, необходимостью первичной загрузки входных параметров, что ярко видно на небольшом потоке документов, во-вторых, необходимостью пересылать большие объемы данных при отправке гранулы на вычисляющий узел. Вместе с тем заметим, что даже при подобных потерях коэффициент ускорения на большом потоке достаточно высок.

3. Поиск с использованием структуры данных

Отсутствие должной степени релевантности и эффективности поиска в больших и сверхбольших хранилищах данных в значительной степени связано со сложностью реализации алгоритмов и моделей, которые позволяли бы учитывать структуры анализируемых данных. Разработка теоретической базы, положенной в основу таких моделей, началась еще с 70-х годов прошлого столетия [5]. Однако, несмотря на достаточно большое количество публикаций по этой тематике, результаты, которые используются на практике, нельзя назвать исчерпывающими. Одна из причин связана с недостатками соответствующих математических решений, особенно для данных со слабой структурой, а другая — с отсутствием инструментальных средств, в первую

очередь, программных, которые позволяли бы эффективно отображать эти модели на платформы вычислительных систем достаточной производительности. Теоретические и практические результаты, которые обсуждаются далее, направлены на решение этих задач.

3.1. Структура данных. На сегодняшний день для представления полуструктурированных данных стандартом de facto является так называемая OEM-модель (Object Exchange Model) [6]. В соответствии с этой моделью, данные представляются в виде совокупности *объектов*, которые могут быть как атомарными, так и составными. Совокупность объектов представляется в виде ориентированного графа с помеченными ребрами, называемым базой данных. Вершины графа соответствуют объектам, а ребра представляют отношения между объектами. Листовым (по аналогии с терминологией деревьев, вершина графа называется листовой, если у нее нет исходящих ребер) вершинам может соответствовать некоторая информация, являющаяся значением атрибута. Приведем формальное определение.

Полуструктурированным документом называется помеченный ориентированный граф $\mathcal{B} = \langle V, \Sigma, E, R\rangle$, где V — множество вершин графа, Σ — множество меток ребер, $E \subseteq V \times \Sigma \times V$ — множество Σ-помеченных ребер, $R \subseteq V$ — корневые вершины базы данных. В дальнейшем будет использоваться следующее обозначение. Любая пара вершин $(x, y) \in V \times V$ графа $\mathcal{B}$ порождает регулярный язык $L_{(x,y)}$, состоящий из всех слов, приписанных путям из x в y.

Встречается несколько вариаций данного определения, отличительные аспекты которых в основном сводятся к следующему:

- множество Σ может быть бесконечным, например, Σ может содержать все возможные значения вершин;
- база данных может рассматриваться либо как один граф, либо как совокупность нескольких графов (совокупность документов);
- в некоторых случаях вместо произвольных ориентированных графов рассматривают упрощенные структуры, например, деревья.

Представление данных при помощи помеченного ориентированного графа описанного вида является *самоопределяющим*, то есть содержит не только сами данные, но и их структуру.

Введем понятие *база полуструктурированных данных*. В литературе встречается два подхода к определению базы полуструктурированных данных: «химический» и «всемирной паутины». В химическом подходе база данных представляется в виде набора несвязанных между собой документов (молекул). Подход всемирной паутины представляет базу данных как один большой документ. В данной работе мы будем придерживаться «химического» подхода. Во-первых, во многих предметных областях деление данных на отдельные фрагменты задано a priori (например, деление на документы в информационной системе). Во-вторых, многие из рассматриваемых далее задач естественным образом формулируются для совокупности документов. При другом подходе решение этих задач требует использования методов декомпозиции графа. Тем не менее, когда это не вызывает противоречий, мы будем пользоваться термином *база данных* вместо документа.

OEM-модель, обладая гибкостью представления различных данных, не может быть напрямую применена для представления XML-документов. Основными различиями между XML и полуструктурированными данными являются отсутствие порядка в OEM-модели и наличие атрибутов у элементов XML. Возможными способами представления XML-документов являются:

- отказ от порядка дочерних элементов и различия между атрибутами и дочерними элементами;
- введение отношения порядка между исходящими ребрами;
- введение дополнительных меток в алфавит.

3.2. Язык запросов. Языки запросов к базам полуструктурированных данных проектируются с учетом характерной особенности полуструктурированных данных — отсутствием единой схемы. Семантически одинаковые фрагменты базы данных могут иметь различную структуру. Язык запросов должен содержать специальные механизмы, обеспечивающие получение «осмысленного» результата даже при несоответствии структуры данных в различных фрагментах базы данных.

В основе языка запросов должна лежать (образующая алгебру) совокупность базовых операций над объектами. В настоящее время предложено несколько вариантов таких операций, учитывающих

нечеткую структуру данных. Мы будем рассматривать только одну операцию, характерную для языков запросов к полуструктурированным данным, и соответствующую поиску при нечеткой структуре данных. Вопросы построения новых объектов, преобразования и изменения данных в языках запросов мы затрагивать не будем.

Задача поиска в базе полуструктурированных данных состоит в нахождении вершин графа-базы данных, связанных определенными условиями.

Во многих языках запросов [7–9] к полуструктурированным данным в качестве базового механизма используются регулярные путевые выражения [10]. Основная идея, лежащая в основе этого метода построения запросов, состоит в том, что при графовом представлении как сами данные, так и их структура описывается путями между вершинами, а регулярные выражения позволяют достаточно простыми средствами описывать широкий класс объектов, и дают возможность сформулировать запрос, не зная точной структуры данных.

Конъюнктивным регулярным путевым запросом (т. е., CRP-запросом) к базе полуструктурированных данных называется ориентированный помеченный граф $Q = \langle X, \mathfrak{L}_R(\Sigma), E_q, R_q \rangle$, где X — множество вершин запроса, $\mathfrak{L}_R(\Sigma)$ обозначает множество регулярных языков над алфавитом, Σ, E_q — множество помеченных ребер, R_q — корневые вершины запроса.

Выбор регулярных языков в качестве метода определения допустимых путей в базе данных вызван тем, что:

- для регулярных языков проблемы вложения и пересечения разрешимы;
- регулярные языки имеют простую строковую нотацию — регулярные выражения;
- любые две вершины помеченного графа порождают регулярный язык (множество слов, получающихся путем выписывания меток всех путей между этими вершинами).

Далее мы будем рассматривать только регулярные путевые запросы, которые для краткости будем называть просто запросами.

Инъективное отображение $\mu : X \to V$ вершин запроса Q в вершины базы данных $\mathcal{B}$ *удовлетворяет запросу*, если:

- $\forall x \in R_q \quad \mu(x) \in R$ (корневые вершины запроса отображаются в корневые вершины базы);

- $\forall x, y \in X : (x, R, y) \in E_q \implies L_{(\mu(x),\mu(y))} \cap L(R) \neq \varnothing$ (при этом образы $\mu(x)$ и $\mu(y)$ назовем согласованными).

Вычисление CRP-запросов сводится к следующим двум вариатам:

- для заданной базы данных $\mathcal{B}$ и запроса Q определить, существует ли отображение вершин Q в вершины $\mathcal{B}$, удовлетворяющее запросу Q;
- для заданной базы данных $\mathcal{B}$ и запроса Q найти все такие отображения.

3.3. Вычисление CRP-запросов. Стратегия вычисления запроса (что очевидно) состоит в последовательном исчерпывающем поиске допустимых образов для вершин $x_1, \ldots, x_n$ запроса Q. Поиск начинается с корня базы данных, для которого вычисляется множество достижимых относительно R_1 вершин. Для каждого допустимого образа x_1 рекурсивно производится поиск допустимых образов x_2 и так далее до тех пор, пока не будут рассмотрены все вершины. Эта стратегия соответствует применению алгоритма Ульмана [5] с «переопределенным» отношением инцидентности. Очевидным недостатком этой стратегии является потенциальная возможность быстрого разрастания дерева перебора. Это связано с тем, что для некоторых регулярных выражений число достижимых вершин может быть велико. Проверка того, являются ли две вершины базы данных смежными относительно заданного регулярного выражения, требует полиномиального (относительно размера базы данных и регулярного выражения) времени и требует, вообще говоря, обхода всей базы данных.

Остановимся подробнее на важности правильного (оптимального) выбора порядка обхода вершин при вычислении запроса (проблема создания оптимального плана выполнения запроса). Стандартное решение выглядит следующим образом. На каждом этапе работы алгоритма рассматривается текущая вершина запроса $x \in X$ и допустимое значение μ_x, которое должна принимать искомая функция μ на вершине x. Для x рассматривается множество исходящих из нее вершин Y_o. Далее накладываются ограничения на множество допустимых значений каждой вершины $y \in Y_o$ (μ_y может быть допустимым значением для y, только если μ_x и μ_y согласованы друг с другом). На следующих этапах работы алгоритма просматриваются вершины из Y_o. Данный план выполнения запроса назовем *прямым обходом*.

Естественными модификациями рассмотренного подхода является использование на каждом шаге работы алгоритма вместо множества Y_o либо множество входящих вершин Y_i — *обратный обход*, либо объединение $Y_i \cup Y_o$ — *комбинированный обход*. Данные обходы можно считать «стандартными» планами выполнения запроса. Основным недостатком стандартных планов является недостаточное использование внутренней структуры запроса. Для создания приемлемых планов необходимо использовать как точные, так и эвристические подходы к анализу графовой структуры запроса и строения входящих в него регулярных языков. При этом на каждом шаге работы алгоритма рассматривается подмножество $Y' \subseteq Y_i \cup Y_o$, выбранное в соответствии с результатами анализа.

3.4. Результаты распараллеливания поиска с использованием структуры данных. Программа поиска с учетом структуры документов [11] была реализована на языке ТС и протестирована на высокопроизводительном вычислительном комплексе семейства «СКИФ», установленном в Институте программных систем РАН в г. Переславле-Залесском. На одном из тестов использовался псевдослучайный помеченный граф со 100 вершинами и 6 типами меток. Вероятности появления меток были определены как: a 0.3, b 0.2, c 0.2, d 0.1, e 0.1, f 0.1. Минимальная степень вершины графа была 8, а максимальная — 10. В качестве запроса использовалась «цепочка», изображенная на Рис. 2.

При вычислении запроса быои применены стандартные планы (D — прямой обход, R — обратный обход и DR — комбинированный обход, а также разработанные «вручную» планы $P_1, \ldots, P_6$). Тестирование производилось на 1, 2, 4 и 8 вычислительных узлах. Результаты приведены в Таблице 2. В качестве основных выводов сформулируем следующие. Для построения эффективного с точки зрения временных затрат плана выполнения запроса необходимо детально (подробно) анализировать его структуру. Стандартные планы D, R и DR могут не совпадать с оптимальным планом и даже использоваться в качестве его удовлетворительного приближения. Применение эвристических соображений, соответствующих планам P_3 и P_5, повысило производительность более чем на два порядка.

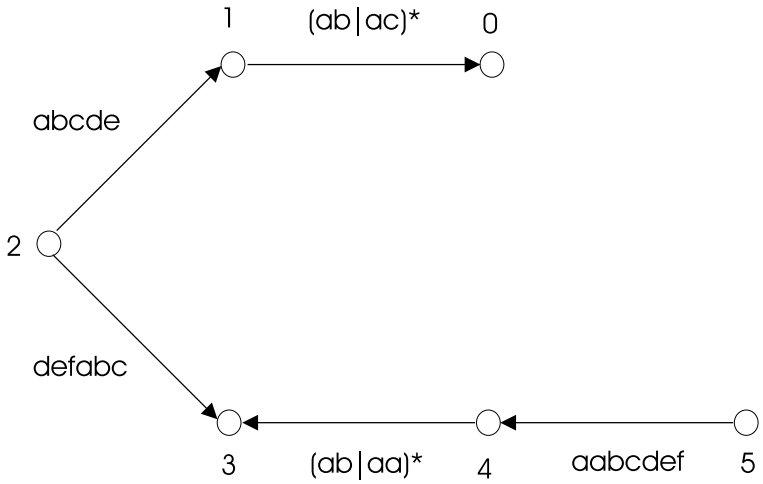

Рис. 2. Тестовый запрос для оценки эффективности эвристик

	D	R	DR	P_1	P_2	P_3	P_4	P_5	P_6
$N = 1$	1069	1565	1237	21.8	77.1	5.8	60.4	6.4	1080.2
$N = 2$	696	1535	805	20.0	51.8	5.2	38.2	5.1	569.4
$N = 4$	429	1569	519	19.9	44.3	4.8	26.0	4.5	302.7
$N = 8$	348	1572	416	24.2	52.1	7.9	36.1	8.1	213.1

Таблица 2. Среднее время вычисления (сек) тестовых запросов при различных планах.

4. Заключение

По результатам тестирования модулей тематического поиска и поиска с учетом структуры документов на массово-параллельных вычислительных комплексах «СКИФ», при реализации которых эффективно использовалось автоматическое динамическое распараллеливание, можно сделать следующие выводы общего характера:

- алгоритмы, которые используют динамическое распараллеливание, могут эффективно применятся для рубрикации на входных потоках документов большого объема. Для малых

потоков (менее 10 мегабайт) накладные расходы на распараллеливание оказываются слишком высоки, и использование параллельных алгоритмов нецелесообразно;

- наиболее критичным ресурсом при работе алгоритма рубрикации оказывается скорость передачи данных между узлами кластера. Следовательно, перспективным является направление исследований в области уменьшения объемов передаваемой на узлы информации, а также ее повторное использование;
- Создание точных оптимальных планов в задаче поиска с учетом структуры документов на сегодняшний день является нерешенной задачей. На практике приходится использовать приближения оптимальных планов. В связи с этим возрастает важность использования систем автоматического динамического распараллеливания, которые позволяют сгладить (по временным затратам) погрешность подобного приближения.

Список литературы

[1] Абрамов С. М., Васенин В. А., Мамчиц Е. Е., Роганов В. А., Слепухин А. Ф. *Динамическое распараллеливание программ на базе параллельной редукции графов. Архитектура программного обеспечения новой версии T-системы* // Труды Всероссийской научной конференции «Высокопроизводительные вычисления и их приложения». — Черноголовка, 2002, с. 261–265. ↑1

[2] Афонин С. А., Козицын А. С. *Поиск текстовых документов с учетом их логической структуры* // Ломоносовские чтения. Секция механики. Тезисы докладов, Апрель 2003. ↑1

[3] Васенин В. А., Роганов В. А. *GRACE: распределенные приложения в Internet* // Открытые системы. — Т. **5–6**, 2001, с. 29–33. ↑1

[4] Боровков А. А. Теория вероятностей: Учеб. пособие. — М.: Наука, 1986. ↑2.5

[5] Ullmann J. R. *An algorithm for subgraph isomorphism* // Journal of the ACM. — Т. **23**, № 1, January 1976, с. 31–42. ↑3, 3.3

[6] Quass D., Widom J., Goldman R., Haas K., Luo Q., McHugh J., Nestorov S., Rajaraman A., Rivero H., Abiteboul S., Ullman J. R., Wiener J. L. *LORE: A Lightweight Object Repository for semistructured data* // Proceedings of the 1996 ACM-SIGMOD International Conference on Management of Data. — Montreal, Quebec, Canada, June 1996, с. 549. ↑3.1

[7] Quass D., Rajaraman A., Sagiv Y., Ullman J., Widom J. *Querying semistructured heterogeneous information* // Deductive and Object-Oriented Databases (DOOD '95), Number 1013 in LNCS: Springer, 1995, с. 319–344. ↑3.2

[8] Deutsch A., Fernandez M., Florescu D., Levy A., Suciu D. *A query language for XML* // Computer Networks (Amsterdam, Netherlands: 1999). — Т. **31(11–16)**, 1999, с. 1155–1169. ↑

[9] Clark J., DeRose S. XML path language (XPath), W3C. ↑3.2

[10] Abiteboul S., Vianu V. *Regular path queries with constraints* // Proc. of the Sixteenth ACM-SIGACT-SIGMOD-SIGART Sym. on Principles of Database Systems (PODS 97), 1997, с. 122–133. ↑3.2

[11] Afonin S., Shundeev A., Roganov V. *Semistructured data search using dynamic parallelisation technology* // Proceedings MIRPO 2003, 2003, с. 152–157. ↑3.4

Научно-исследовательский институт механики МГУ им. М.В. Ломоносова

V. A. Vasenin, S. A. Afonin, A. S. Kozitsin, A. S. Shundeev. *The search in large data warehouses and high performance cluster computing.* (in russian.)

Abstract. One of the most important challenges for current information technologies is the important problem of high relevant information search in the large volumes of heterogeneous data. In this paper a system of thematic monitoring and semi-structured data search is considered. The system was developed for high performance computing cluster "SKIF" (IPS RAS, Moscow State University) using dynamic parallelization software system «Open TS». The main results and conclusions are present.