для проектирования перспективных изделий запрос-ответной аппаратуры / Вестник Технологического университета. 2018. Т. 21. № 2. С. 155-162.

2. Мокшин В.В., Якимов И.М. Метод формирования модели анализа сложной системы / Информационные технологии. 2011. № 5. С. 46-51.

3. Мокшин В.В., Якимов И.М., Кирпичников А.П., Шарнин Л.М. Разработка системы мониторинга состояния грузоподъемных механизмов / Вестник Технологического университета. 2017. Т. 20. № 19. С. 75-81.

4. Родина Р.В. Имитационное моделирование как средство оптимизации процессов производства // Научные достижения и открытия современной молодежи: сборник статей Международной научно-практической конференции в 2 ч. 2017. Ч.1. С. 75-77.

5. Якимов И.М., Абзалова Л.Р., Кирпичников А.П., Мокшин В.В. Краткий обзор графических редакторов структурных моделей сложных систем / Вестник Казанского технологического университета. 2014. Т. 17. № 17. С. 213-221.

В.С. Кочетков

ИССЛЕДОВАНИЕ ПРИМЕНИЯ МЕТОДОВ ПАРСИНГА DOM И SAX ДЛЯ ПОЛУЧЕНИЯ ДАННЫХ О ГОСУДАРСТВЕННЫХ ЗАКУПКАХ

(Самарский университет)

Формат XML очень широко используется в информационных технология, рекомендован Консорциумом Всемирной паутины(W3C). Отличным примером применения технологии XML могут служить открытые данные информационной системы zakupki.gov.ru, доступные на соответствующем ftp: объём ежедневных обновлений – десятки и сотни тысяч XML файлов объемом в гигабайты или десятки гигабайт, в среднем раз в несколько недель выходит новая версия схемы данных.

Актуальность исследования методов синтаксического анализа XML-файлов обусловлена существенной экономией времени при обработке больших массивов файлов, когда количество файлов исчисляется десятками тысяч, даже если при обработке одного файла экономия времени незначительна.

В случае с ftp-сервером государственных закупок структура данных следует за требованиями законодательства. В связи с этим информация об извещениях о проведении государственных закупок представлена более чем десятком типов документов fcsNotification* в зависимости от типа закупки (электронный аукцион fcsNotificationEF, запрос котировок fcsNotificationZK, закупка у единственного поставщика fcsNotificationEP и т.п.). Все эти документы основаны на одном базовом типе извещения, но отличаются в деталях, и это требуется учитывать при импорте и анализе данных. [2]

Также существую проблемы с получением данных о государственных закупках средствами сайта: существует ограничение в количестве результатов,

поиск выдает не более, чем 500 строк, и скачивание данных о конкурсах (в виде таблицы .csv формата) средствами сайта невозможно. При этом потребность иметь регулярно обновляемые сведения о проходящих конкурсах, попадающих под определенные критерии, возникает у каждой организации, участвующей в государственных закупках.[3]

Каждый XML файл содержит всю информацию о закупке, которая зачастую является излишней при решении прикладной задачи. Необходимость парсинга XML-файлов обусловлена избыточностью хранящихся на ftp данных и постоянным изменением их структуры. Для решения этой задачи проведено исследование времени получения атрибутов ограниченного набора тегов документа путем обработки файлов с помощью основных методов синтаксического анализа XML: DOM и SAX.

DOM (Document Object Model - объектная модель документов) - платформенно-независимый программный интерфейс. Модель DOM не накладывает ограничений на структуру документа. Любой документ известной структуры с помощью DOM может быть представлен в виде дерева узлов, каждый узел которого содержит элемент, атрибут, текстовый, графический или любой другой объект. Узлы связаны между собой отношениями родитель-потомок.

Способ доступа через DOM удобен, когда структура документа должна (или может) быть доступна в целом и легко модифицироваться.

Для анализа документов методом DOM путем парсинга XML-файла создается DocumentBuilderFactory, с помощью которой реализуется DocumentBuilder, который, в свою очередь, выполняет разбор XML документа для создания объекта Document и последующего извлечения информации путем прохода по всем узлам(nodes) объекта для нахождения искомых атрибутов.

Поскольку DOM-парсер реализует объектную модель документа, его полезно использовать там, где требуется работать с документом целиком, как с деревом. Представление всего документа будет занимать в памяти значительный объем, поэтому DOM резонно считается моделью, очень требовательной к ресурсам.

SAX (Simple API for XML) базируется на модели последовательной одноразовой обработки и не создает внутренних деревьев. При прохождении по XML вызывает соответствующие методы у классов, реализующих интерфейсы, предоставляемые SAX-парсером.

SAX-разбор происходит асинхронно, то есть программа не должна ждать завершающего тега для отработки уже полученных. Последнее может пригодиться, например, для отображения записей SQL-запроса еще до того, как все они будут получены. Другое применение - обрыв сеанса, допустим при нахождении одной нужной записи из миллиона.

Недостаток SAX - необходимость хранения состояния в процессе разбора XML-потока, то есть в случае со сложной и/или недетерминированной структурой придется частично реконструировать узловую модель для отслеживания уровней вложения.

Средства парсинга с потоковой обработкой применимы в задачах, где повторное обращение не нужно. В таком случае происходит последовательный перебор узлов до нужного и поочередная загрузка в память только текущего узла. Данные средства могут анализировать очень большие документы и подходят для таких задач, как индексация или преобразование в другие форматы (например, замена дескрипторов XML на дескрипторы HTML). [4]

SAX-парсер реализует интерфейс DefaultHandler, который использует функции обратного вызова, для извлечение данных по набору условий вида:

В первую очередь обрабатывается событие startDocument.Когда синтаксический анализатор SAX разбирает XML и встречает запуск тега (например,<something>), он запускает событие tagStarted(фактическое название события может отличаться). Аналогично, когда конец тега встречается во время разбора (</something>), он запускает tagEnded. Использование анализатора SAX подразумевает, что вам необходимо обрабатывать эти события и понимать данные, возвращаемые с каждым событием. Если в элементе есть содержимое, будут вызываться такие события, как characters для дополнительного текста, startElement и endElement для дочерних элементов и т.п. Заканчивается работа парсера событием endDocument.

SAX и DOM подходы иногда сочетают между собой. Так, веб-браузер отображает элементы в процессе получения методом SAX, но после получения документа строит его DOM-модель для DHTML-доступа в терминах иерархии объектов.[5]

В рамках работы проведены исследования реализации методов синтаксического анализа информации, влияния количества документов и набора извлекаемых тегов на время выполнения анализа.

Для сравнения времени, затрачиваемого на извлечение и обработку информации из XML-файлов с ftp-сервера zakupki.gov.ru были выгружена и размещена на локальном компьютере контрольная выборка документов, чтобы минимизировать влияние на время обработки файлов факторов, не относящихся к теме исследования. В качестве средства реализации был выбран язык Java, на котором были написаны классы, реализующие описанные ранее методы обработки слабоструктурированной информации, а также классы сущностей Notification с различными наборами атрибутов.

Эксперименты проводились для контрольных выборок документов размера $N = 1000, N = 5000, N = 10000$. Набор тегов, прописанных в условиях парсера, по которым производился поиск данных также варьировался от 4 до 16. Целью исследования контрольных выборок документов было обнаружение корреляции между временем выполнения парсинга файлов и количеством извлекаемой информации.

И в случае работы SAX-парсера, и в случае работы DOM-парсера, набор тегов передается конструктору объекта Notification для последующей записи объекта в репозиторий.

Для исключения влияния на результаты исследования случайных факторов было проведено 10 итераций извлечения данных по контрольным выборкам

с различными наборами тегов, данные были занесены в таблицы на основании которых была составлена диаграмма времени обработки контрольных выборок документов (рисунок 1).

Следует отметить, что на время выполнения обработки существенное влияние оказывает набор извлекаемой информации. В случае использования метода SAX, и в случае использования метода DOM добавление условий для извлечения дополнительных наборов тегов значительно влияет на время обработки файлов. На рисунке 3 представлены диаграммы времени обработки контрольных наборов файлов методами DOM и SAX.

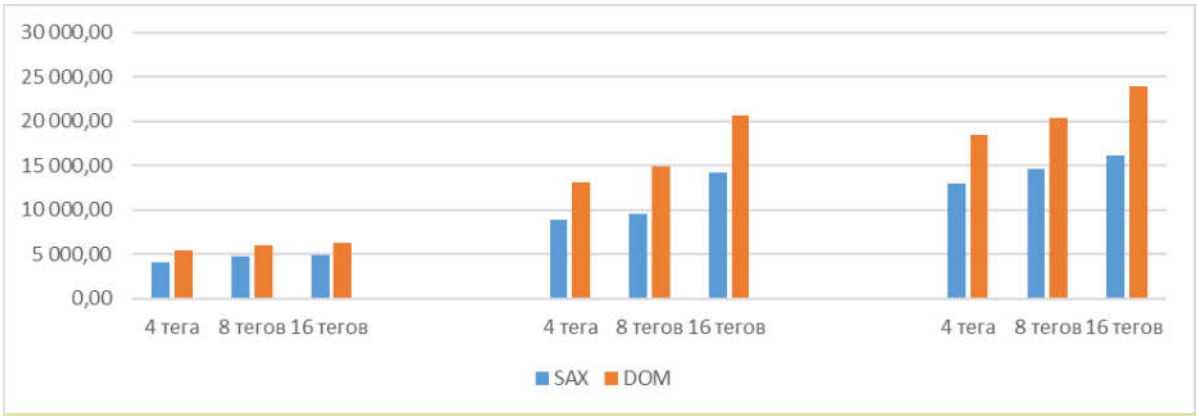

Рис. 1. Диаграмма времени обработки контрольных выборок документов методом DOMи SAXпри различном наборе извлекаемых тегов(на вертикальной оси показано время в миллисекундах, на горизонтальной три группы столбцов для выборок размером N=1000, 5000, 10000 соответственно)

Использование метода SAX позволяет в среднем до 1,5 раз уменьшить время, затрачиваемое на обработку XML-файлов с целью извлечения данных по сравнению с аналогичными операциями, производимыми над тем же набором файлов, но при использовании метода DOM.

В случае SAX-парсера увеличение количества тегов в 4 раза дало прирост времени обработки файлов в 1,33 раза.

В результате работы было выявлено, что для целей получения данных с ftp информационной системы zakupki.gov.ru рационально использовать метод SAX, поскольку в общем случае время, затрачиваемое на обработку файла методом DOM может превышать в 1,5 раза время обработки файла методом SAXдля получения содержимого аналогичного набора тегов.

Стоит учитывать, что метод SAX подходит только для извлечения информации, и в случае необходимости модификации данных неприменим, в отличии от DOM. Однако в случае решения задачи получения данных, этой особенностью можно пренебречь.

На основании полученных данных с ftp возможно проводить дальнейшие исследования, позволяющие формировать рекомендации для поставщиков и заказчиков. Например, выявление оптимального расположения оптовых баз, или оптимальной начальной цены контракта для обеспечения конкуренции.

Литература

1. Бретт Маклахлин. Java и XML, 2-е издание, Символ-Плюс, 2016. – 544 страницы.
2. Интеграция XML данных — другой путь // [habrahabr.ru] –2017 – URL: https://habr.com/ru/post/325186/
3. Выгрузка условий конкурсов с zakupki.gov.ru // [habrahabr.ru] –2015 – URL: https://habr.com/post/253201/
4. Чеботарев,А. XML: свобода, ограниченная только фантазией// Компьютеры+Программы. - 2003. - № 5. - С. 52-55
5. Лыгина,Н.И., Пудич,А.С.. Исследование правильности и эффективности средств парсинга информации на веб-ресурсах// Инновационная наука, №. 3-1, 2017, С. 59-67.

М.А. Кузин[1], Д.Л. Головашкин[2]

ЭКСПЕРИМЕНТАЛЬНОЕ ИССЛЕДОВАНИЕ БЛОЧНО-ПАРАЛЛЕЛЬНОГО АЛГОРИТМА РАЗНОСТНОГО РЕШЕНИЯ ВОЛНОВОГО УРАВНЕНИЯ

([1] Самарский национальный исследовательский университет имени академика С.П. Королева,
[2] Институт систем обработки изображений РАН – филиал ФНИЦ «Кристаллография и фотоника» РАН)

Введение

Актуальность разностного решения волнового уравнения связана с его использованием в современной вычислительной оптике [1,2]. Основной проблемой при этом признается [2] значительная длительность расчетов при реализации такого решения, обусловленная высокой вычислительной сложностью. Популярным в последнее время инструментом решения указанной проблемы, в англоязычной литературе именуемым "tiling" [3], является построение блочных алгоритмов. Классическим подходом к сокращению длительности вычислений при разностном решении волнового уравнения по-прежнему остается разработка параллельных алгоритмов [4]. Объединение перечисленных методик представляется авторам настоящей работы весьма перспективным в силу успешности такого приема при организации вычислений по FDTD (Finite-Difference Time-Domain) методу [5] и несомненной связи обоих численных методов [2,6].

1. Исследование параллельного и блочного алгоритмов

В качестве аппаратной базы для проведения экспериментов была выбрана ЭВМ, оснащенная процессором AMD Ryzen 5 2600, 3,4 Ггц, материнской платой MSI B450 Tomahawk (MS-7C02) с частотой системной шины 8000 МГц и оперативной памятью DDR4-3200Мгц объемом 16 Гб. Работающая под управлением операционной системы Ubuntu 18.04.2. Для компиляции кода использо-