17.Кирсанов М.Н. Расчет пространственной стержневой системы, допускающей мгновенную изменяемость //Строительная механика и расчет сооружений. 2012. № 3. С. 48-51.
18.Кирсанов М.Н. Аналитический расчет пространственной стержневой системы//Строительная механика инженерных конструкций и сооружений. 2012. № 1. С. 49-53.
19.Kirsanov M.N. Analysis of the buckling of spatial truss with cross lattice. Magazine of Civil Engineering. 2016. No. 4. Pp. 52–58. doi: 10.5862/MCE.64.

УДК 004(051+057.5)

Н.И. Лыгина
К.п.н., доцент, кафедра АСУ, НГТУ
г. Новосибирск, Российская федерация
А.С. Пудич
Магистрант, НГТУ
г. Новосибирск, Российская федерация

ИССЛЕДОВАНИЕ ПРАВИЛЬНОСТИ И ЭФФЕКТИВНОСТИ СРЕДСТВ ПАРСИНГА ИНФОРМАЦИИ НА ВЕБ-РЕСУРСАХ

Аннотация

В рамках исследования конкретизированы критерии качества средств парсинга в соответствии с ГОСТ Р ИСО/МЭК 9126-93. Для тестирования на правильность и эффективность выбраны и описаны двадцать стандартных средств парсинга и средств парсинга сторонних разработчиков, в том числе и средства парсинга на основе регулярных выражений и строковых функций языка, разработанные в рамках данного исследования. Приведена характеристика используемого в исследовании программного инструмента и условий тестирования средств парсинга. Проанализированы результаты тестирования средств парсинга и определены рекомендации по их выбору.

Ключевые слова

Качество программного обеспечения, эффективность, правильность, средства парсинга.

В настоящее время много областей бизнеса агрегируют информацию. Автоматизированный сбор больших объемов информации с максимально возможной скоростью выполняют специальные программы, которые относятся к так называемым программам-ботам.

Как правило, в процессе сбора информации бот выполняет следующие операции: получает адрес веб-страницы ресурса, получает доступ к коду веб-страницы по сети и загружает ее посредством HTTP клиента, читает, извлекает и систематизирует данные, отделяя необходимую информацию от программного кода страницы, сохраняет полученные данные в базе данных или предоставляет результаты по необходимости в определенном виде.

Информацию, которую извлекают боты, накапливают с целью выбора по определённым критериям или анализа динамики, резервного хранения на случай утраты доступа к первоисточникам, а также используют для периодического обновления.

Отделение нужной информации происходит посредством синтаксического и лексического анализа её содержимого. Этот процесс получил название парсинга. Парсинг является частью многих процессов обработки информации от перевода текста до трансляции кода языков высокого уровня в машинный код. Следует отметить, что чрезмерное копирование информации алгоритмами поисковых систем давно расценивается как повод для пессимизации или блокировки ресурса.

В данной статье рассматривается парсинг веб-страниц, особо востребованный в индустрии поискового продвижения сайтов (SEO). Существует большое количество средств парсинга веб-документов и их программных реализаций на различных языках. В связи с этим возникает задача выбора средства парсинга по определённым характеристикам.

Сравнительная оценка средств парсинга в соответствии с ГОСТ Р ИСО/МЭК 9126-93 «Оценка программной продукции». Характеристики качества и руководства по их применению» выполнялась по следующим характеристикам и их атрибутам:

- функциональные возможности определялись атрибутом правильность, а именно, распознавание кодировки страницы и представление всей извлеченной информации в доступном для дальнейшей обработки виде;
- эффективность определялась по времени и по объёму потребляемой оперативной памяти при обработке одного веб-документа;
- адаптируемость определялась как возможность переноса средств парсинга из одного конкретного окружения в другое (Windows, PHP 7; Linux, PHP 5.6; Linux, PHP 7).

В работе рассмотрены возможные методы извлечения информации с помощью языка программирования PHP 7 версии. Этот язык обладает высокой производительностью и имеет поддержку современного объектно-ориентированного программирования. На данный момент не существует подобного исследования для средств языка PHP, несмотря на долгое существование некоторых из них. Можно только указать на работу по анализу производительности анализаторов HTML документов на языках Erlang, Python, PyPy, Node.JS, C [3].

Для парсинга веб-документов HTML, XHTML, XML стандартными средствами PHP чаще всего применяются программные интерфейсы (API) DOM (Document Object Model), SAX (Simple API for XML), XMLReade, SimpleXML . Эти типы интерфейсов включены стандартно в дистрибутивы PHP, базируются на библиотеке libxml2 проекта GNOME.

По способу загрузки веб-документа API бывают либо с загрузкой в память целиком и построением объектной структуры, либо с потоковой загрузкой (последовательно по частям), в последнем случае различают передающий (push) анализатор (на основе событий) и принимающий (pull) анализатор (курсорный).

Сравнения DOM (Document Object Model) и SAX (Simple API for XML) описаны в [2]. Принцип работы SAX и пример реализации описаны в [7], XMLReader – в [5]. При разработке программной реализации были использованы примеры для DOM, SAX и XMLReader из [4], для SimpleXML – из [6].

Для сравнения выбраны 20 средств парсинга веб-документов форматов HTML, XHTML, XML. Множество оцениваемых средств ограничено средствами встроенного расширения libxml2, а также средствами, разработанными с помощью стандартных функций обработки строк и регулярных выражений, и зарекомендовавшими себя средствами сторонних разработчиков.

В данном исследовании проанализированы также неспециализированные средства парсинга, использующие функции работы со строками и специальный формальный язык, называемый языком регулярных выражений (RegExp). Идея создания средства парсинга с помощью функций обработки строк была взята из [1]. Регулярные выражения в PHP реализованы в библиотеке PCRE (Perl-совместимые регулярные выражения). Средства на их основе требуют реализации под конкретную задачу.

Средства парсинга выбирались по их востребованности, типу интерфейса (объектно-ориентированный, потоко-ориентированный), способу поддержки запросов (XPath выражения, CSS селекторы) и способу поддержки форматов (HTML, XHTML, XML, Автоопределение). Для выделения характеристик сравнения стандартных средств и средств сторонних разработчиков для обработки веб-документов был проанализирован исходный код этих средств.

Отметим некоторые особенности средств парсинга, которые послужили основанием для их включения в группу исследуемых.

Программные каркасы фреймворк Zend Framework, содержащий компонент Dom, и фреймворк Symfony 2, содержащий компоненты CssSelector и DomCrawler, являются на сегодняшний день наиболее

известными и поддерживаемыми. Они способны определять тип данных веб-документа самостоятельно, если он не указан, а также поддерживают селекторы XPath и CSS, которые конвертируются в XPath.

Средство fDOMDocument расширяет DOM API. Основная цель средства заключается в генерации исключений об ошибках вместо PHP предупреждений и уведомлений. Это полезно, если необходимо диагностировать место ошибки в документе.

Средство FluentDOM среди загружаемых форматов имеет также форматы JSON, CSV, JsonML, RabbitFish, PHP.

К средствам, допускающим цепочный стиль программирования (jQuery подобный), относятся средства DiDom, FluentDOM, phpQuery (реализует паттерн Singleton), QueryPath (используется как модуль в CMF Drupal, позволяет использовать альтернативы при выборке элементов, чего нет в других решениях), Simple HTML Dom, Advanced HTML DOM. Цепочный стиль удобен при разработке и популярен в среде программистов.

Средство Nokogiri отличается малым объёмом программного кода (300 строк) и использованием всего одного класса.

Следует особо выделить средства парсинга, разработанные без помощи стандартных расширений библиотеки *libxml*:

- средство *Simple HTML DOM* имеет цепочный интерфейс, поддерживает *CSS*селекторы;
- средство *Ganon* позволяет разбирать по частям *HTML/XML/RSS*; поддерживает *HTML5*, невалидный *HTML*, кодировку *UTF-8*, *CSS3* подобные запросы к элементам;
- средство HTML5lib имеет событийный (SAX-подобный) анализатор с возможностью выбора потока для обработки; сереализует HTML5; лексический анализатор разбирает и классифицирует данные методом рекурсивного спуска, строит DOM дерево в памяти и позволяет использовать XPath для обращения к элементам.

Наряду со средствами PHP в исследовании оценивается средство парсинга, разработанное на JavaScript API. Средство работает в сценарном веб-браузере PhantomJS, который запускается из PHP, таким образом, внешне работа с данным средством не отличается от работы с остальными.

Для оценки средств парсинга по выбранным характеристикам качества не существует специального инструмента, соответственно была разработана система тестирования.

В разработанной для исследования среде тестирования использовалось следующее программное обеспечение: HTML Tidy версии 5.3.12 и его настройка, сценарный веб-браузер PhantomJS версии 2.1.1, Composer версии 1.2.2 для загрузки средств парсинга сторонних разработчиков, PHPмодуль xDebug для сбора статистики работы скриптов (отладка и профилирования), Webgrind для просмотра статистики, собранной с помощью xDebug, веб-сервер Apache2, веб-сервер Nginx и его настройка.

В исследовании качества всех средств парсинга использовались одна и та же задача на языке PHP для веб-интерфейса и интерфейса командной строки (CLI), автозагрузчик разработанных классов, работающий как в операционной системе Linux, так и в Windows, и веб-интерфейс.

Инструментом тестирования являлась HTML форма с выбором обрабатываемого веб-документа, его формата и последующей обработкой веб-документа средствами парсинга на PHP посредством AJAX запросов, причем каждый в различном потоке. Результаты динамически передавались на страницу и добавлялись в результирующую таблицу. Ограничение времени выполнения одного AJAX запроса равнялось 300-ам сек. Ошибки выводились в консоль браузера, также использовался интерфейс командной строки (*CLI*).

Инструменты тестирования всех средств парсинга разработаны с помощью сценариев *bash* и *PHP* в операционной системе *Linux* и с помощью пакетных файлов *batch*, сценариев *PowerShell* и *PHP* в *Windows*. В автоматическом режиме обрабатываются *HTML*, *XHTML*, *XML* форматы веб-документов разного размера и в различных вариантах итераций. Времена работы каждого средства парсинга было равно 300 сек. Ошибки выводились в стандартный поток вывода (*stdout*), также использовался инструмент преобразования результатов в *CSV* таблицы.

В ходе тестирования соблюдены определённые условия:

- все средства парсинга выполняют одинаковую полезную работу, таким образом, находятся в равных условиях друг с другом. Для исследований выбрана ситуация сбора ссылок документа – содержание узла и атрибута для HTML документов, атрибут узла и содержание дочернего для XML документа;
- проверяется корректность извлечения ссылок на тестовой странице, содержащей всевозможные способы размещения элемента <a> на странице, в том числе в невалидном формате, вложенном в элемент <pre> и в комментарии;
- учитывается наличие накладных расходов (overhead) на работу с кодировками. Код примера, использующий средство парсинга, должен быть дополнен кодом, решающим проблему различных кодировок, если, опираясь на свои возможности, средство не может этого делать. Таким образом, все средства парсинга должны возвращать одинаковый результат;
- программа один раз загружает веб-документ в память (исключение – потоковые средства парсинга) и затем N раз последовательно обрабатывает, где N = {1, 5, 20, 50, 100}.

Выбранные для тестирования веб-документы и их характеристики (формат, размер, URL) представлены ниже:

- типовая страница блога *Wordpress, HTML,* 75 КБ, https://en.blog.wordpress.com/;
- поисковая выдача (*SERP*) *Google* по запросу «*parsing php*», *HTML,* 333 КБ, https://www.google.ru/search? hl=ru&q=parsing+php;
- страница перечня доменов верхнего уровня онлайн-энциклопедии *Wikipedia, HTML,* 934 КБ, https://en.wikipedia.org/wiki/ List_of_Internet_top-level_domains;
- страница статьи, содержащей более 900 комментариев на ресурсе «Хабрахабр», *HTML,* 2,45 МБ, https://habrahabr.ru/post/70330/;
- официальный перечень всех MIME-типов на ресурсе организации *IANA, XHTML,* 584 КБ, http://www.iana.org/assignments/media-types/media-types.xhtml;
- каталоги товаров магазина Ozon.ru различных размеров, XML, 164 КБ, 1.63 МБ, 23.1 МБ, 159 МБ, http://www.ozon.ru/context/ detail/id/2643202/.

В итоге для каждого средства парсинга проводилось тестирование с каждым из 9-ти входных веб-документов при пяти различных вариантах итераций их обработки.

Тестирование показало, что стандартные средства парсинга соответствуют норме качества «Правильность». Неспециализированные средства, реализованные с помощью функций обработки строк и регулярных выражений, требуют дополнительного преобразования извлекаемого текста в нужную кодировку.

Тестирование средств сторонних разработчиков в отличие от стандартных показало наличие недостатков в их работе:

- средство *Zend Dom Query* требует у *XML* документов отсутствия специальной *DOCTYPE* инструкции, что согласно стандарту, выходит за рамки валидности *XML* документов, но таким образом разработчики повышают безопасность, избегая *XEE* инъекций [8]; однако замечено, что средство работает с веб-документами, содержащими *DOCTYPE* инструкции, если они представлены в кодировке *UTF-8* с *BOM* меткой;
- средство *QueryPath* завершает обработку с ошибкой, если *XML* документ закодирован в *UTF-8* с *BOM*;
- средство *DiDom* требует указания кодировки обрабатываемого веб-документа, если обрабатывается *XML* документ, иначе данные преобразуются неправильно;
- средства *Ganon, XMLReader, Expat* (преобразование в структуру), средства на основе регулярных выражений и строковых функций требуют дополнительные преобразования кодировки извлекаемых данных, поскольку средства не обладают данной функцией;

• средство *HTML5lib* для правильной работы требует указания кодировки обрабатываемого веб-документа, но не имеет в документации информации о работе с различными кодировками, однако возможности для этого у средства есть;

• средство Simple HTML Dom не способно распознать указания кодировки из документов HTML 5 версии: при проверке распознавания ссылок в HTML документе у следующих средств не обнаружены ссылки: Simple HTML Dom не обнаружил ссылку внутри тега CODE;Expat (с созданием структуры) не обнаружил ссылку внутри тега TEXTAREA;HTML5lib и PhantomJS не обнаружили ссылку внутри тегов XMP и TEXTAREA.

Для анализа эффективности работы средств в различных окружениях использовался сервер со следующими техническими характеристиками: оперативная память объёмом 4 ГБ, процессор *Intel Core i7* с четырьмя ядрами.

Следует отметить, что для оценки эффективности средства *Zend Dom Query* пришлось исключать инструкции *DOCTYPE* из *XML* документов с помощью регулярных выражений.

Для исправления веб-документов с невалидной разметкой для средств парсинга *XMLReader*, *Expat* (преобразование в структуру) используется инструмент *HTML Tidy* версии 5.3.12 (http://www.html-tidy.org/).

На рис. 1в качестве примера показаны и подписаны в порядке уменьшения времени обработки средства парсинга в окружении *Windows* с *PHP* 7. Не показаны средства *DOM*, *PhantomJS*, *phpQuery*, т.к. они не справились с обработкой большого *XML* документа в течение 300 секунд при одной итерации. Средство *Advanced HTML Dom* не показано по причине неадаптируемости к окружению.

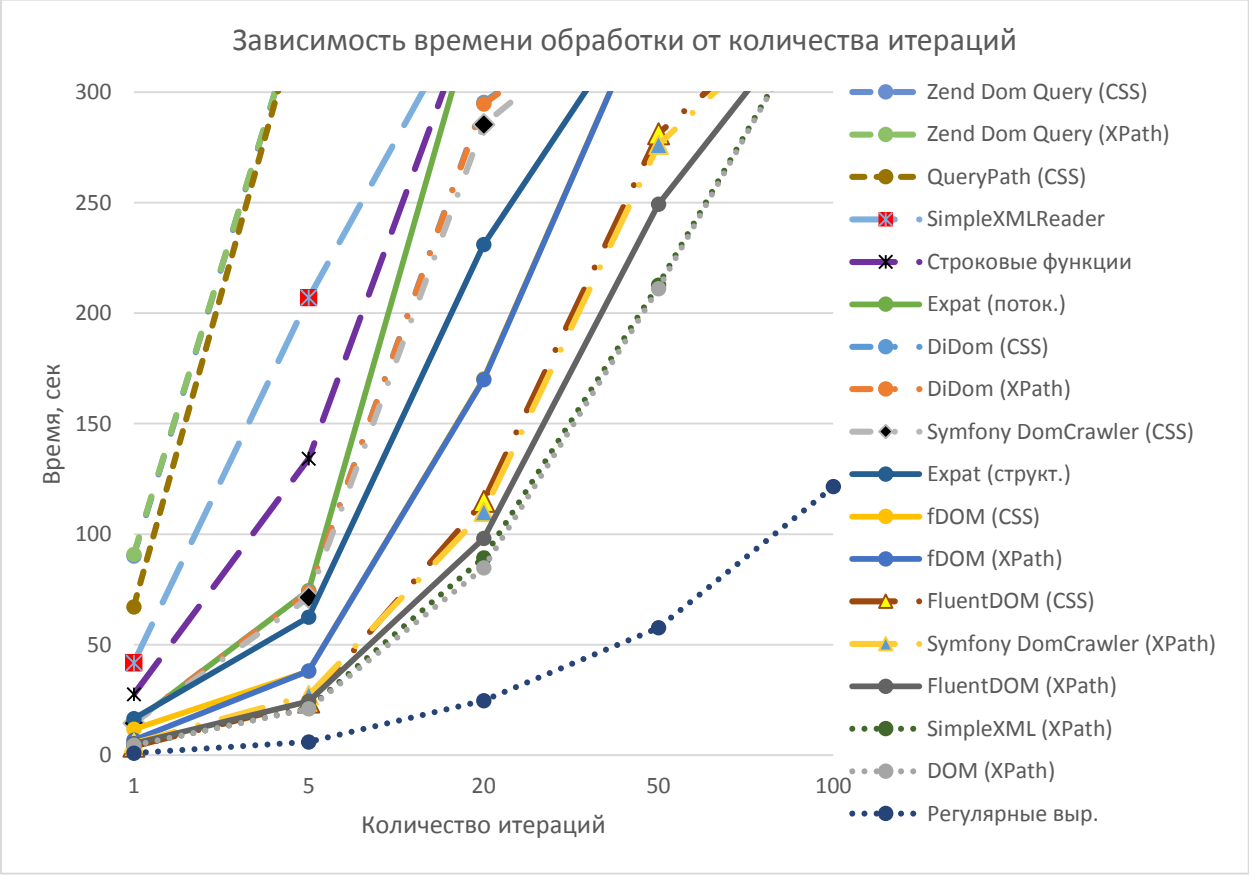

Рисунок 1 – Зависимость времени обработки XML документа каталога «Ozon.ru» размером 159 МБ в окружении Windows с PHP 7 версии от количества итераций

На рис. 2 показана разница времени обработки *XML* документа объёмом в 23.1 МБ между смежными средствами парсинга – потоковым *Expat* и *Expat* с преобразованием в структуру, стандартным *SimpleXML* и

SimpleXMLReader, *phpQuery* и *QueryPath*, а также сравнение с парсингом с помощью консольного веб-браузера *PhantomJS* и стандартным средством *DOM API*.

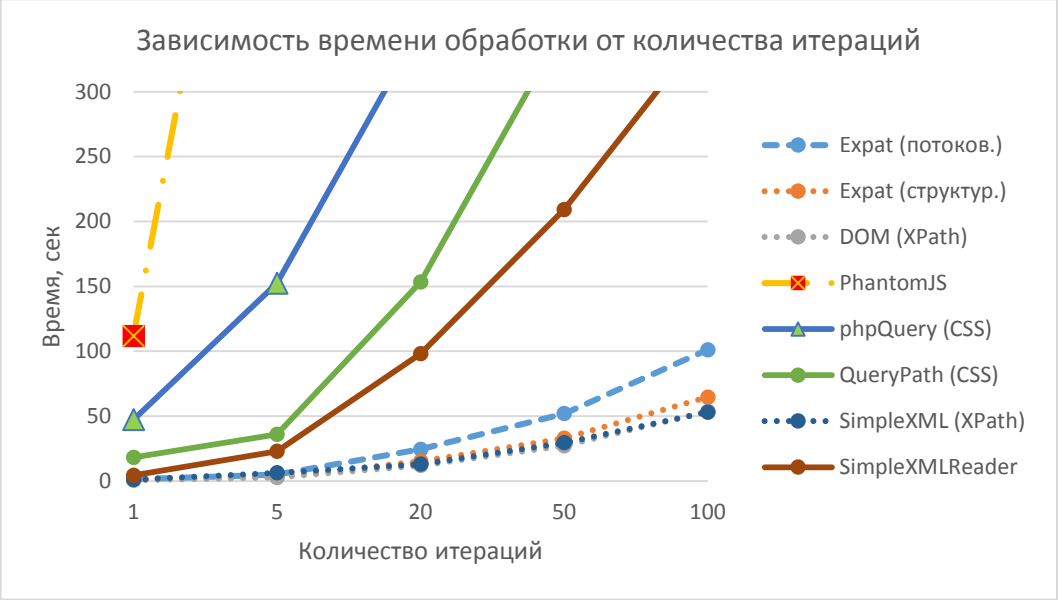

Рисунок 2 – Зависимость времени обработки XML документа каталога «Ozon.ru» размером 23.1 МБ в окружении Linux с PHP 7 версии от количества итераций

Анализ эффективности работы средств парсинга в трёх окружениях позволяет говорить также об их адаптируемости. На графиках приведены примеры замеров только для некоторых из окружений, а ниже представлено сравнение результатов тестирования средств парсинга по всем выбранным для исследования окружениям:

- средство *Ganon* показывает наибольшее время обработки среди всех средств;
- средство *Advanced HTML Dom* не приспособлено к работе на *Linux* с *PHP* 7, завершает работу ошибкой выделения памяти;
- сценарный веб-браузер *PhantomJS* действительно не является конкурентом *PHP* средствам по скорости, но это объясняется затратами времени на запуск бинарного файла из *PHP* скрипта, который также учитывался в замерах, однако запуск в *Linux* происходит в два раза быстрее, чем в *Windows*;
- средства *HTML5lib* и *Simple HTML Dom* при *PHP* 5.6 в *Linux* отрабатывают дольше, чем при *PHP* 7 в *Linux* (в 2-3 раза) и *Windows* (в 1.5-2 раза), имеют наибольшее количество вызовов функций, также *Simple HTML Dom* в принципе не выдаёт результата при обработке большого документа (статья ресурса «Хабрахабр»);
- средство *Expat* (с преобразованием в структуру) в *Windows* при *PHP* 7 работает в 2-3 раза дольше, чем в *Linux* с большими файлами, и до 10 раз дольше при работе с небольшими файлами;
- средство *Zend Dom Query* при работе с большими файлами (статья «Хабрахабр», страница *Wikipedia*) в *Linux* при *PHP* 5 работает в 10-12 раз медленнее, чем при *PHP* 7 в *Linux* и *Windows* (рис. 1);
- средство *Symfony Dom Crawler* при использовании *CSS* селекторов работает заметно медленнее, чем при *XPath*, особенно при обработке *XML* документов (рис. 1);
- средство *phpQuery* в *Linux* при *PHP* 5 с ростом итераций работает дольше (~1.5 раза), чем при *PHP* 7 в *Windows*, и еще дольше при *PHP* 7 в *Linux*;
- средство *QueryPath* на больших файлах в 1.5-2 раза медленнее работает при *PHP* 5 по сравнению с *PHP* 7;
- средство phpQuery быстрее QueryPath при обработке небольших XML документов, но в обработке более объёмных XMLи HTML документов медленнее (рис. 2); как утверждается в [9], средство phpQuery

быстрее работает при операциях чтения, а QueryPath – при операциях записи, но по результатам данного тестирования можно предположить, что в [9] не тестировались большие файлы;

- средство *SimpleXMLReader* в 4-8 раз проигрывает по скорости *SimpleXML* с *XPath* (рис. 1, рис. 2);
- средство *Expat* с потоковой обработкой в 1.6 раз медленнее, чем *Expat* с преобразованием в структуру (рис. 1, рис. 2);
- средство на основе строковых функций при обработке *XML* (или *HTML*) документов в *Linux* с *PHP* 7, как правило, в 5 (или 3) раз быстрее, чем с *PHP* 5, и в 15 (или 4,5) раз быстрее, чем в *Windows* с *PHP* 7;
- средство на основе регулярных выражений при *PHP* 5 с ростом количества итераций работает с большими *XML* документами в 4 раза медленнее, чем при *PHP* 7; по сравнению с обработкой строковыми функциями обработка регулярными выражениями значительно выигрывает по скорости.

При росте числа итераций в случае обработки больших XML документов наиболее медленными оказались средства парсинга *phpQuery*, *DOM*, *SimpleXMLReader, Zend Dom Query* (особенно при *PHP* 5) (рис. 1), а при обработке *HTML* и *XHTML* документов – *Ganon* (самый медленный), *Simple HTML Dom*, *PhantomJS*, *HTML5lib*. В свою очередь наиболее быстрыми являются средство парсинга регулярными выражениями (самый быстрый результат был получен при *PHP* 7), *DOM (Xpath)*, *SimpleXML*, *FluentDom* (*CSS* и *XPath*), *Symfony Dom Crawler* (*XPath*), средство парсинга на основе строковых функций (только в *Linux* при *PHP* 7) (рис.1).

Использование оперативной памяти средствами парсинга показано на рис. 3. Если потребление памяти растет с увеличением числа итераций, то это указывает на утечки памяти или проблемы со сборщиком мусора.

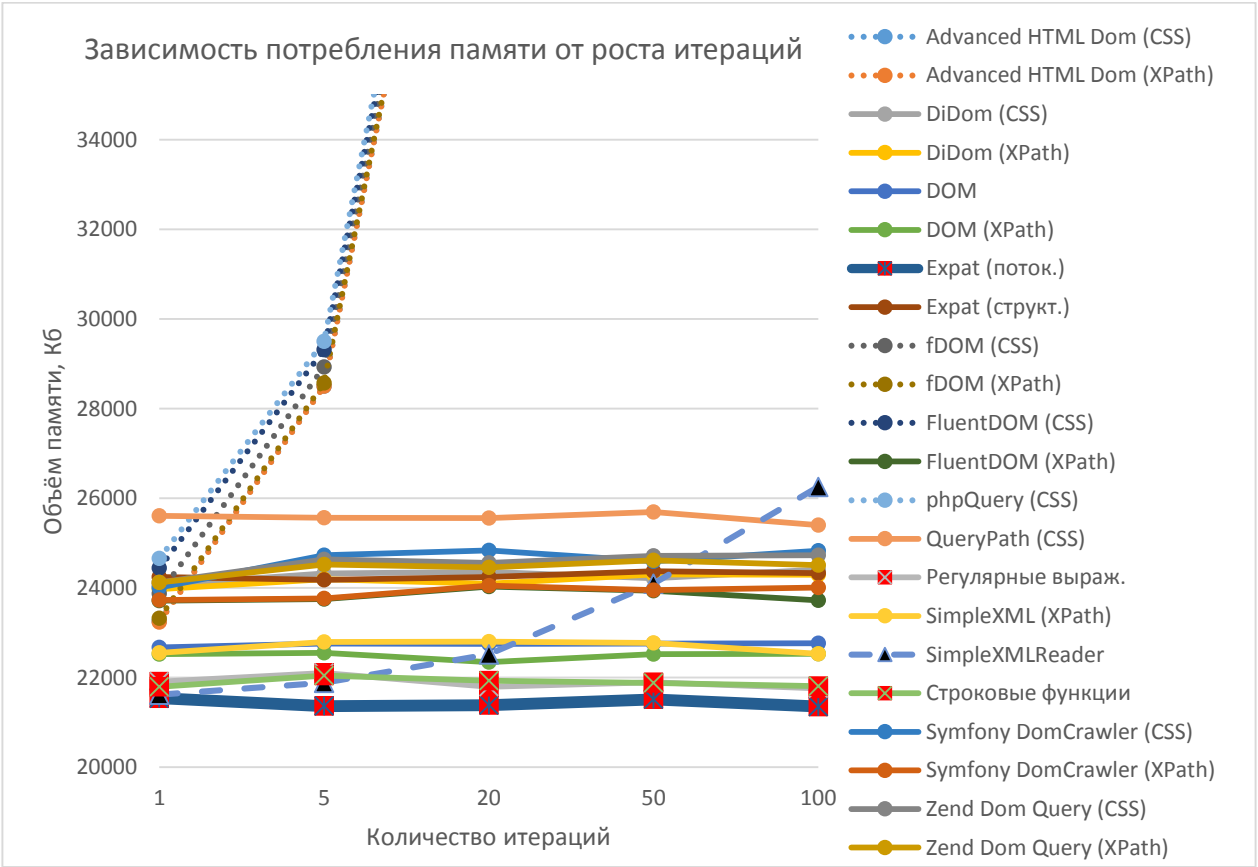

Рисунок 3 – Зависимость потребления памяти при обработке XML документа в окружении Linux с PHP 5 версии от числа итераций

При однократных замерах некоторые средства парсинга при обработке *XML* документа размером 159 МБ потребляли до 2.7 ГБ оперативной памяти (рис. 4).

Рисунок 4 – Потребляемая память при обработке XML документа размером 159 МБ

При сравнении потребляемой памяти средствами парсинга выявлено, что:

- при разовом измерении максимально потребляемой памяти при обработке HTML документов выигрывают средство *Expat* с преобразованием в структуру и средства парсинга с помощью регулярных выражений и строковых функций, а также *XPath* версии средств *fDOMDocument*, *DOM*, *Fluent DOM* (рис. 4), однако *Expat* с созданием структуры не эффективен для обработки *XML* документов;
- в динамике роста итераций наблюдаются утечки памяти у *Advanced HTML DOM*, *phpQuery*, *fDOMDocument* (*CSS* и *XPath*), *Fluent DOM* (*CSS*), а также у *Ganon* и *Simple HTML DOM* (рис. 1);
- потоковые средства *Expat* и *SimpleXMLReader* выигрывают в обработке *XML* документов у остальных средств, однако у *SimpleXMLReader* был отмечен рост потребления ресурсов с ростом количества итераций (рис. 3).

Выводы

Средства парсинга, у которых каждый анализируемый веб-документ хранится в памяти в виде иерархии его элементов (узлов), при повторном обращении к различным частям документа имеют меньшее время обработки веб-документа, поскольку не нужно заново загружать и строить дерево. Такие средства предоставляют наиболее удобный способ доступа к элементам веб-документа с помощью запросов (*CSS* и *XPath*) и подходят для любых задач многократной выборки информации из веб-документа.

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*).

Средство парсинга регулярными выражениями применимо для задач, где необходимо извлечь конкретную часть документа с заранее известной структурой. Быстрый поиск соответствий с шаблонами, написанными с помощью регулярных выражений, гарантирует эффективный результат при условии надёжно составленных выражений. Синтаксис регулярных выражений гораздо сложнее, чем у *CSS* селекторов и даже *XPath* запросов, а также чувствителен к небольшим изменениям в разметке. Такое средство парсинга целесообразнее использовать там, где средства, использующие стандартные расширения библиотеки *libxml*, не могут помочь, например, в извлечении комментариев в коде или фрагментов кода перед разбором любыми другими средствами.

На основе анализа результатов тестирования сформулированы рекомендации по выбору средств парсинга:

- средства парсинга *Fluent DOM* (*XPath*), *DOM* (*XPath*), *Symfony Dom Crawler* (*XPath*) можно использовать для обработки документов различного формата и размера с максимальной скоростью при минимальном потреблении памяти;
- средство парсинга *Zend Dom Query* также можно использовать для обработки документов на *PHP* 7, исключая *XML* документы большого размера; при обработке *XML* документов нужно исключать инструкцию *DOCTYPE*;
- средства *DiDom*, *nokogiri* стабильно работают, но обладают меньшим функционалом и не столь универсальны, как вышеназванные средства;
- средство парсинга регулярными выражениями можно использовать с максимальной скоростью из всех средств в несложных задачах, таких как извлечение комментариев в коде или фрагментов кода перед разбором стандартными средствами парсинга, однако стоит использовать *PHP* 7 вместо *PHP* 5.6;
- средство парсинга с обработкой строковыми функциями показывает наилучшую эффективность только в окружении *Linux* с *PHP* 7;
- средство *Expat* с потоковой обработкой подтвердило свою эффективность по всем показателям при разборе только *XML* документов; использование *Expat* с преобразованием в структуру не эффективно для *XML*, а результаты предоставляются в неудобном для дальнейшей обработки виде;
- сценарный веб-браузер *PhantomJS* с достаточной эффективностью обрабатывает только небольшие документы; при обработке пакета документов эффективность обработки снижается по сравнению с другими средствами парсинга;
- окружение *Linux* с *PHP* 7 версии обеспечивает наибольшую эффективность обработки любыми средствами парсинга по сравнению с *Linux* с *PHP* 5.6, *Windows* с *PHP* 7.

Следует отметить, если кодировкой обрабатываемых веб-документов является *UTF-8* без *BOM* метки, то дополнительные действия по преобразованию не нужны.

Список использованной литературы:

1. Schrenk M. Webbots, Spiders, and Screen Scrapers 2nd edition: A Guide to Developing Internet Agents with PHP/CURL. No Starch Press Inc. 2012. P. 362.
2. Чеботарев А. XML: свобода, ограниченная только фантазией [Электронный ресурс] // ЦИТ Форум. 2004. URL: http://citforum.ru/internet/xml/xml_fant/ (дата обращения: 24.11.2016).
3. Бенчмарк HTML парсеров [Электронный ресурс] // Сайт «Хабрахабр» – Разработка. 2012. 26 декабря. URL: https://habrahabr.ru/post/163979/ (дата обращения: 19.11.2016).
4. Морган К. [Morgan C.] XML для PHP-разработчиков: Часть 2. Расширенные методы парсинга XML: пер. с англ. [Электронный ресурс] // Сообщество developerWorks. 2010. 15 апреля. URL: http://www.ibm.com/developerworks/ru/library/x-xmlphp2/index.html (дата обращения: 10.12.2016).
5. Расти Хэролд Э. [Rusty Harold E.] Синтаксический анализ XML в PHP: пер. с англ. [Электронный ресурс] // Сообщество developerWorks. 2007. 11 октября. URL: http://www.ibm.com/developerworks/ru/library/x-pullparsingphp/index.html (дата обращения: 10.12.2016).
6. Gervasio A. Introducing SimpleXML in PHP 5 [Электронный ресурс] // Dev Shed. 2006. 12 июня. URL: http://www.devshed.com/c/a/PHP/Introducing-SimpleXML-in-PHP-5/ (дата обращения: 11.12.2016).
7. Grant J. PHP and XML: Using the expat functions [Электронный ресурс] // 2000. URL: http://www.phpbuilder.com/columns/justin20000428.php3 (дата обращения: 10.12.2016).
8. ZF2012-02: Denial of Service vector via XEE injection [Электронный ресурс] // Zend Framework – Security. 2012.20 августа. URL: https://framework.zend.com/security/advisory/ZF2012-02 (дата обращения: 18.01.2017).