Сравнение веб-краулеров Scrapy, Heritrix и Apache

Перевод статьи отсюда:

BLIKK BLOG http://blog.blikk.co/comparison-of-open-source-web-crawlers

Требования к веб-краулерам:

Надежность: Веб содержит серверы, которые создают ловушки для пауков, которые являются генераторами веб-страниц, вводящих сканеры в заблуждение, из-за которого они застревают в выборке бесконечное количество страниц в данном домене. Краулеры должны быть разработаны устойчивыми к таким ловушекам. Не все такие ловушки созданы специально, некоторые являются непреднамеренным побочным эффектом небрежной разработки веб-сайта.

Вежливость: веб-серверы имеют и явные и неявные политики, регулирующие частоту, с которой Искатель может посетить их. Эти политики вежливость должны соблюдаться.

Распределенность: Краулеры должны иметь возможность выполняться в распределенном режиме на нескольких машинах.

Масштабируемость: Архитектура краулеров должна позволять увеличение скорости сканирования путем добавления дополнительных машин и пропускной способности.

Производительность и эффективность: система сканирования должна обеспечивать эффективное использование различных системных ресурсов, включая процессор, память и полосу пропускания сети.

Качество: Учитывая, что значительная доля всех веб-страниц имеют плохие утилиты для обслуживания потребностей запросов пользователя, сканер должен быть нацелен в сторону первоочередного извлечения «полезных» страниц.

Свежесть: Во многих приложениях, сканер должен работать в непрерывном режиме: он должен получить свежие копии ранее исследованных страниц. Краулер поисковой системы, например, таким образом, может гарантировать, что индекс поисковой системы содержит достаточно свежее представление каждой индексированной веб-страницы. Для такого непрерывного обхода, краулеры должны быть в состоянии сканировать страницы с частотой, приблизительно равной скорости изменения этих страниц.

Расширяемость: Сканеры должны быть сконструированы, чтобы быть расширяемыми во многих отношениях - чтобы справиться с новыми форматами данных,

новые протоколами доступа, и так далее. Это требует использовать модульную архитектуру краулеров.

Ползать вширь или прицельно

Прежде чем перейти к сути сравнения сделаем шаг назад и рассмотрим на два разных случая применения веб-краулеров: фокусированные обходы и широкие обходы.

В фокусированном обходе вы заинтересованы в определенном наборе страниц (как правило, конкретной области). Например, вы можете сканировать все страницы продукта на amazon.com. В широком обходе набора страниц вы заинтересованы в очень большой или неограниченной зоне поиска. Это, как правило, то, что поисковые системы и делают. Это не различие между черным и белым. Это континуум. Фокусированный обход с большим количеством доменов (или нескольких целевых обходов, выполняемых одновременно) существенно приближаются к свойствам широкого сканирования.

При обходе одного домена (например, amazon.com), вы, по сути, ограничиваетесь своей политикой вежливости. Вы не хотите завалить сервер тысячами запросов в секунду, чтобы быть заблокированным. Таким образом, вы должны ввести искусственный предел запросов в секунду. Этот предел обычно устанавливается в зависимости от времени ответа сервера. В связи с этим искусственным ограничением большинство процессоров и сетевых ресурсов вашего сервера будет простаивать. Владение распределенным краулером, использующим тысячи машин не сделает сфокусированный обход быстрее, чем его работа на вашем ноутбуке.

В случае широкого обхода узким местом является производительность и масштабируемость краулера. Потому что если вам нужно запросить страницы из разных доменов, вы можете потенциально выполнить миллионы запросов в секунду, не перегружая определенный сервер. Вы ограничены количеством машин, которые у вас есть, их процессорами, пропускной способности сети и тем, насколько хорошо ваш краулер может использовать эти ресурсы.

Если все, что вы хотите – наскрести данные из нескольких доменов, то поиск краулеров веб-уровня может быть излишним. В этом случае обратите внимание на услуги, такие как kimono и import.io, которые оба являются огромными скребками особых данных с веб-страниц. SCRAPY (описано ниже) также является отличным выбором для ориентированного обхода.

SCRAPY

Веб-сайт: http://scrapy.org/

Язык: Python

SCRAPY это фремворк, написанный на языке Python для веб-скрайпинга. Она не имеет встроенных функций для работы в распределенной среде (на множестве компьютеров), так что для нее основной вариант использования - фокусированный обход. Это не означает, что SCRAPY не может быть использована для широкого обхода, но другие инструменты могут лучше подходят для этой цели, особенно при очень больших масштабах. В соответствии с документацией, наилучшей практикой для распределения обходов является вручную разделить URL-адреса, расположенные в домене.

Что выделяет SCRAPY является ее простота в использовании и отличная документацию. Если вы знакомы с Python, вы установите и будете работать всего через пару минут.

SCRAPY имеет несколько удобных встроенных форматов экспорта, таких как JSON, JSON lines, XML и CSV. SCRAPY былf построен для извлечения конкретной информации из веб-сайтов, не для получения полного дампа HTML и его индексации. Последнее требует ручной работы, чтобы избежать записи полного содержания всех HTML страниц в один огромный выходной файл. Вы должны нарезать файлы вручную.

Без способности работать в распределенной среде, динамической масштабируемости и работы в режиме непрерывного обхода, в SCRAPY отсутствуют некоторые из нужных ключевых особенностей. Тем не менее, если вам нужно легкий в использовании инструмент для извлечения конкретной информации из нескольких доменов, то SCRAPY почти идеальна. Я успешно пользуюсь в ряде проектов и был очень доволен ей.

Heritrix

Веб-сайт: https://webarchive.jira.com/wiki/display/Heritrix/Heritrix

Язык: Java

Heretrix разработан, поддерживается и используется The Internet Archive. Его архитектура описана в этой статье, и в значительной степени основана на научно-исследовательской работе Меркатора. Heretrix хорошо поддерживался с момента его выпуска в 2004 году и в настоящее время практически используется на различных других сайтах.

Heretrix работает в распределенной среде с помощью хеширования URL хостов в соответствующих машин. Как таковой, он является масштабируемым, но не динамически.

Это означает, что вы должны определить количество используемых машин, прежде чем начать обход. Если одна из машин падает во время обхода, вам не повезло.

Форматом вывода Heretrix являются WARC файлы, которые записываются в локальной файловой системе. WARC является эффективным форматом для записи нескольких ресурсов (например, HTML) и их метаданных в одном архивном файле. Запись данных в других хранилищах данных (или форматов) в настоящее время не поддерживается, и похоже, что это потребует довольно много изменений в исходном коде.

Непрерывное сканирование не поддерживается, но над этим работают. Однако, как и во многих проектах с открытым исходным кодом время разработки новых возможностей может быть довольно длительным, и я бы не стал рассчитывать на поддержку непрерывного обхода в ближайшее время.

Heretrix, вероятно, самый зрелый из open source gроектов, которые я рассматривал. Я обнаружил, что он проще в установке, настройке и использовании, чем Nutch. В то же время он более масштабируемой и быстрее, чем SCRAPY. Он поставляется вместе с веб-интерфейсом, который может быть использован для контроля и настройки обходов.

APACHE Nutch

Веб-сайт: http://nutch.apache.org/

Язык: Java

Вместо того чтобы строить свою собственную распределенную систему Nutch использует экосистему Hadoop и использует MapReduce для его обработки (подробнее). Если у вас уже есть существующий кластер Hadoop, можно просто указать Nutch на него. Если у вас нет существующего кластера Hadoop вы должны установить и настроить его. Nutch наследует преимущества (например, отказоустойчивость и масштабируемость), но и недостатки (медленный доступ диска между заданиями в связи с его пакетной природой) архитектуры Hadoop MapReduce.

Интересно отметить, что Nutch не стартует, как чистый веб краулер. Он начался, как поисковая система с открытым исходным кодом, которая обрабатывает как обход, так и индексацию веб-контента. Даже если Nutch с тех пор стал больше, чем поисковый робот, он все еще поставляется в глубокой интеграции с комплектом систем для индексации, таких, как Solr (по умолчанию) и ElasticSearch (через плагины). Более новая ветвь Nutch 2.x пытается отделить базовую программу для хранения данных от краулера с помощью Apache Gora, но все еще находится в довольно ранней стадии. В моих собственных экспериментов я обнаружил, что он довольно незрелый и багги. Это

означает, что если вы планируете использовать Nutch вы, вероятно, будет ограничены сочетая его с Solr и ElasticSearch, или написать свой собственный плагин для поддержки различных форматов бэкэнд или экспорта.

Несмотря на большое количество предварительного опыта работы с Hadoop и проектами на базе Hadoop я нашел Nutch довольно сложным в установке и настройке, в основном из-за отсутствия хорошей документации или реальных примеров.

Nutch (1.x), кажется, стабильная платформа, которая используется в производстве различными организации, CommonCrawl среди них. Он имеет гибкую систему плагинов, что позволяет расширять его функциональные возможности. В самом деле, это, кажется, необходимо в большинстве случаев применения. При использовании Nutch можно ожидать небольших затрат времени на написание собственных плагинов или подбора подходящего к вашему случаю исходного. Если у вас есть время и знания, чтобы сделать так, то Nutch кажется отличной базовой платформой.

Nutch настоящее время не поддерживает непрерывные обходы, но вы могли бы написать пару скриптов для эмуляции такой функциональности.