УДК 004.75

МОДЕЛИ И АЛГОРИТМЫ РАСПРЕДЕЛЕННОГО ПОИСКОВОГО РОБОТА

Пранскевичус В. А., Привалов М. В.
Донецкий национальный технический университет
кафедра автоматизированных систем управления
E-mail: vlprans@gmail.com, evilmax@gmail.com

Аннотация

Пранскевичус В. А., Привалов М. В. Модели и алгоритмы распределенного поискового робота. *В данной статье проанализированы основные методы и модели, применяющиеся в параллельных поисковых роботах. Рассмотрены архитектуры поисковых роботов, политики параллелизации и метрики, применяемые для оценки поисковых роботов. Выделены основные пути исследований политик параллелизации и повышения масштабируемости.*

Общая постановка проблемы

В связи с бурным развитием Всемирной паутины, с каждым днем все более актуальной становится проблема автоматизированного сбора и анализа информации, размещаемой на различных веб-ресурсах. Еще в начале 90-х годов прошлого столетия Всемирная паутина представляла собой огромное количество слабо структурированной информации, производить поиск в которой стало для человека практически непосильной задачей. Именно в это время стали появляться первые разработки в сфере автоматизированных агентов, облегчающих задачу поиска необходимой информации в паутине. Основной частью таких систем является поисковый робот — программный комплекс, осуществляющий навигацию по веб-ресурсам и сбор информации для базы данных приложения-агента. В общем случае, собираемая роботом информация состоит из веб-страниц и ссылочной структуры веба.

Задача построения эффективного поискового робота является довольно нетривиальной, по нескольки причинам:

- Всемирная паутина является гетерогенной хаотично развивающейся средой, большая часть ресурсов которой содержит нарушения принятых стандартов веб-разработки;
- в связи с практически экспоненциальным ростом объемов информации во Всемирной паутине, поисковый робот должен позволять эффективно обрабатывать большое количество веб-ресурсов за конечное время и иметь архитектуру, пригодную для масштабирования;
- поисковый робот должен быть достаточно универсальным для того, чтобы гибко подстраиваться под нужды использующего его приложения.

Благодаря огромным объемам и гетерогенности Всемирной паутины, архитектура поискового робота и в частности политики параллелизма, заложенные в ней, являются интересным объектом исследований. На сегодняшний день ключевым моментом, определяющим производительность поискового робота является горизонтальная масштабируемость его архитектуры, т.е. свойство системы увеличивать производительность при добавлении новых узлов (компьютеров). Достаточно интересное исследование и попытка классификации различных архитектур параллельных поисковых роботов представлены в [1].

Однако, существующие архитектуры поисковых роботов достаточно сложны, и их масштабируемость зачастую далека от линейной. Также, при масштабировании возможно ухудшение значений метрик качества собираемой роботом информации. Можно сделать

вывод, что дополнительные исследования политик параллелизации поисковых роботов могут улучшить масштабируемость и производительность робота, а также повысить качество собираемых им данных.

Исследования

Учитывая обозначенные выше сложности, связанные с обходом Всемирной паутины, архитектуры и политики параллелизации поисковых роботов общего назначения изначально разрабатывались таким образом, чтобы обеспечить максимально быстрое получение данных и простоту масштабирования. Можно выделить такие два больших класса архитектур параллельных поисковых роботов:

- Централизованные архитектуры — лежат в основе большинства используемых на сегодняшний день поисковых роботов. Эти архитектуры состоят из нескольких потенциально распределенных конкурентных компонентов, имеющих центральный пункт синхронизации(например, очередь задач или специальный компонент-координатор). Ярким примером поискового робота, имеющего централизованную архитектуру является Googlebot [2]. Обычно при использовании централизованной архитектуры применяется внутридоменная (intra-site) модель распределения (согласно классификации данной в [1]), при которой компоненты распределяются внутри одного локального окружения. Структурная схема поискового робота с централизованной архитектурой представлена на рис. 1.

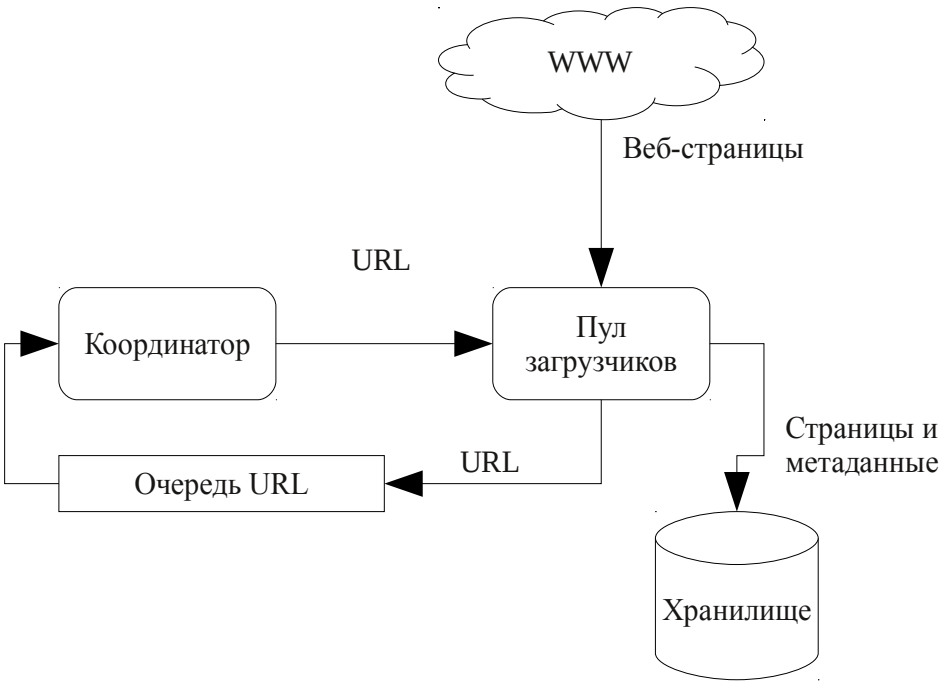

Рисунок 1– Централизованная архитектура поискового робота

Децентрализованные или мультиагентные архитектуры — отличаются полной (или приближенной к полной) децентрализацией компонентов. Построение подобных систем требует несколько иных алгоритмов и методов, нежели тех, которые используются в централизованных архитектурах, поэтому их можно вынести в отдельный класс. Поисковые роботы, построенные на основе децентрализованной архитектуры состоят из автономных агентов, выполняющих задачи обхода Всемирной паутины. Задачи распределяются между агентами при помощи алгоритмов наподобие DHT (distributed hash table, распределенная хеш-таблица) или Consistent Hashing. Такие методы могут равномерно распределять задачи

между агентами и позволяют простое добавление и удаление агентов без нарушения работоспособности системы. Пример использования децентрализованной архитектуры приведен в [3].

На более низком уровне, архитектуры поисковых роботов могут отличаться методами распределения задач между компонентами, способами обмена URL и алгоритмом их распределения. Конфигурация конкретного поискового робота может варьироваться в зависимости от нужд использующего его приложения. Например, для минимизации объема загружаемых данных и оптимизации производительности веб-краулера могут использоваться отложенные вычисления в форме абстракций futures [5].

Для оценки поисковых роботов, можно использовать следующие метрики, предложенные в [1]:

- дублирование (overlap) — оценивает объем избыточно загруженных страниц и определяется по формуле:

$$O = \frac{N - I}{I}, \quad (1)$$

где N — общее количество загруженных страниц, I — количество уникальных страниц.

- покрытие (coverage) — определяет объем страниц, которые должны были быть загружены, но благодаря специфике функций распределения адресов между агентами не были. Покрытие можно определить следующим соотношением:

$$C = \frac{I}{U}, \quad (2)$$

где I — количество уникальных страниц, U — общее количество страниц.

- качество (quality) — для определения качества, предположим существование гипотетического предсказывающего робота (oracle crawler), которому заранее известно значение важности страницы по какой-либо метрике (для определения важности страницы могут использоваться различные метрики, например количество обратных ссылок или PageRank [7]). Обозначим множество “интересных” страниц мощности N, загруженных предсказывающим роботом как P_N , а аналогичное множество страниц, загруженных реальным роботом как A_N. Исходя из этого, можно определить качество как:

$$Q = \frac{|A_N \cap P_N|}{|P_N|} \quad (3)$$

Перечисленные выше метрики находятся в прямой зависимости от политик параллелизации, заложенных в архитектуре поискового робота, и имеют тенденцию ухудшаться при распределении конкурентных компонентов или увеличении их числа [1]. Этот факт явно показывает необходимость дальнейших исследований политик параллелизации.

Параллельный поисковый робот, по своей сути представляет собой образец системы с конкурентными вычислениями, и следовательно к нему могут быть применены математические методы и модели конкурентных вычислений. Одной из наиболее выразительных моделей конкурентных вычислений является π-исчисление [4]. П-исчисление является Тьюринг-полной моделью вычислений и является мощным инструментов для формализации и исследования конкурентных процессов с комплексными взаимодействиями.

Базовое π-исчисление предлагает следующие примитивы для описания процессов:

- конкурентность, обозначается как $P \mid Q$, где P, Q — процессы;

- входной префикс, *c(x).P* определяет процесс *P*, ожидающий сообщения *x*, которое должно быть доставлено по каналу *c*;
- выходной префикс *c(y).P,* определяющий процесс *P*, осуществляющий отправку имени *y* по каналу *c*;
- репликация, обозначается как *!P*, может быть рассмотрена, как процесс, который всегда может создавать новую копию *P*.
- создание нового имени, *(υx)P* создает локальное имя *x* в контексте процесса *P*.

В качестве примера определения процесса в терминах π-исчисления можно рассмотреть процесс *Fetch*, выполняющий непосредственную загрузку страницы по данному URL:

$$Fetch[f,store]=!\,f(s,p).(vc)\left(\bar{s}\langle p,c\rangle.c(data).\bar{store}\langle s,p,data\rangle\right) \qquad (4)$$

Данный процесс представляет собой абстракцию π-исчисления (т.е. обладает параметрами); URL в виде (s, p), где s — сервер, p — страница, принимается по каналу f, затем происходит запрос страницы с сервера с последующим ее получением по каналу с параметром data. Затем data передается по каналу store в хранилище. К процессу применяется оператор репликации, что позволяет запускать параллельно несколько загрузок.

Таким образом, при помощи π-исчисления можно промоделировать другие компоненты системы, что позволит более формально подойти к исследованию архитектур веб-краулеров и политик параллелизации. π-исчисление позволяет моделировать сложные взаимодействия между компонентами системы и проводить структурные преобразования при помощи механизмов бисимуляции и структурной конгруэнтности процессов.

В теории конкурентных вычислений принято выделять два основных механизма взаимодействия: взаимодействие при помощи общей памяти и взаимодействие путем передачи сообщений. Это разделение также находит отражение в реализациях языков программирования и фреймворков. В семантике π-исчисления заложено межпроцессное взаимодействие при помощи передачи сообщений, что необходимо учитывать при выборе средств реализации поискового робота.

Основными требованиями, выдвигаемыми к языку реализации являются:

- наличие высокоуровневых конструкций и средств для функциональной декомпозиции системы и создания необходимых абстракций;
- выразительность, близкая к формальному математическому описанию системы;
- наличие встроенных средств для описания конкурентных вычислений, позволяющих легко реализовать процессы системы и их взаимодействия;
- наличие качественных, поддерживаемых сообществом разработчиков библиотек, например для сетевых взаимодействий, протоколов обмена сообщениями, разбора HTML страниц, работы с СУБД и т. п.;
- открытость и общедоступность реализации языка и инструментов разработки.

С учетом выдвинутых требований, в качестве языка реализации был выбран язык программирования Haskell (http://www.haskell.org). Haskell относится к семейству языков ML и представляет собой чистый функциональный язык с отложенными вычислениями. Haskell обладает строгой статической типизацией с автоматическим выводом типов по модели Хиндли-Милнера, поддержкой алгебраических и рекурсивных типов данных, вызовом функции по образцу.

Определяющим фактором в выборе именно этого языка программирования стала его продвинутая реализация примитивов конкурентных вычислений (Concurrent Haskell), близкая по семантике к π-исчислению [6]. Примитивы Concurrent Haskell реализуют все основные операции π-исчисления, кроме недетерминированного выбора, который тем не менее может быть реализован в случае необходимости при помощи других примитивов.

Выводы

Специфика задачи обхода Всемирной паутины приводит к неминуемому использованию конкурентных вычислений при разработке поисковых роботов. Для эффективной реализации конкурентности и хорошей масштабируемости поискового робота, необходимо уделять немалое внимание разработке архитектуры системы и политик параллелизации. Для оценки эффективности архитектуры поискового робота может использоваться ряд специально разработанных метрик.

Проанализировав существующие архитектуры, политики параллелизации и методы построения поисковых роботов, для разрабатываемой системы была выбрана децентрализованная архитектура. Также было предложено использование π-исчисления как механизма формализации и дальнейшего исследования политик параллелизации поисковых роботов. Для самой реализации поискового робота был выбран язык программирования Haskell и его расширение Concurrent Haskell как наиболее близкий к семантике π-исчисления способ описания конкурентных вычислений.

Литература

- Junghoo Cho and Hector Garcia-Molina. Parallel crawlers // In Proc. of the 11th International World–Wide Web Conference – 2002.
- Sergey Brin, Lawrence Page. The Anatomy of a Large-Scale Hypertextual Web Search Engine // Computer Science Department, Stanford University, Stanford — 1998. - С. 107-117.
- Paolo Boldi, Bruno Codenotti, Massimo Santini, Sebastiano Vigna. UbiCrawler: a scalable fully distributed Web crawler // Software: Practice and Experience – 2004. - С. 711-726.
- Robin Milner. Communicating and Mobile Systems: the Pi-Calculus // Cambridge, UK: Cambridge University Press – 1999. - 162 С.
- Пранскевичус В. А., Привалов М. В. Построение масштабируемого сфокусированного поискового робота с использованием принципа отложенных вычислений // Вестник ЛНТУ им. Шевченко – 2010. - С. 189-194.
- Simon Peyton Jones , Andrew Gordon , Sigbjorn Finne. Concurrent Haskell // Proceedings of the 23rd ACM SIGPLAN-SIGACT symposium on Principles of programming languages – 1996. - С. 295-308.
- Larry Page, Sergey Brin, R. Motwani, T. Winograd. The PageRank Citation Ranking: Bringing Order to the Web // Technical Report, Stanford InfoLab – 2001.