Московский Государственный Университет имени М.В.Ломоносова
Факультет Вычислительной Математики и Кибернетики
Кафедра Системного Программирования

Курсовая работа

Разработка и реализация прототипа системы потоковой обработки данных

Автор:
гр. 427
Овчинников Дмитрий

Научный руководитель:
д.т.н профессор Кузнецов Сергей Дмитриевич

Москва, 2015

Содержание

1 Постановка задачи

1.1 Актуальность

Поскольку рост информации коллосален в последнее время, ее нужно как-то быстро обрабатывать, и чем быстрее, тем лучше. Существуют такие технологии, как Hadoop, MapReduce, которые безусловно полезные, но довольно-таки медленные и непригодные для "Big data in realtime". Это сфера, где нужно обрабатывать много (потенциально бесконечно) данных за очень короткий промежуток времени, к таким относятся, например:

1. Системы анализа и мониторинга
2. Задачи оптимизации работы сетей и отдельно каждого устройства в сети
3. Обнаружение подозрительных действий на охраняемых территориях
4. Телефония, обслуживание АТС
5. Финансовый трейдинг

Это сфера ИТ еще нова и только набирает популярность, и до некоторого момента проблема быстрых вычислений решалась путем закупки дорогостоящего оборудования. Однако сейчас решение проблемы все больше и больше ложится на плечи программистов, которые меняют подходы к обработке информации.
Существует тенденция, когда все вычисления (которых слишком много) ставят "на поток т.е. выделяют отдельно систему, которая обрабатывает непрерывные данные, приходящие к ней и передает дальше, не задерживая. О разработке такой системы и пойдет речь в данной курсовой работе.

1.2 Общие требования

Планируется реализовать систему, ввиде отдельной библиотеки, которая представляла бы собой один кластер распределенной системы потоковой обработки данных. Для начала разработки необходимо обозначить набор требований, которым должна удовлетворять система:

1. Система не должна использовать работу с жестким диском, чтобы уменьшить время выполнения.

2. Система должна предоставлять инструменты для создания графов(топологий) вычислений.

3. Система должна иметь возможность пользоваться всеми ресурсами процессора, т.е. должна иметь возможно распараллеливания.

4. Система желательно быть кросс-платформенной

5. Система должна гарантировать полный проход графа вычислений.

6. Система не должна заботиться о том, как принимать данные и куда их потом передавать.
   Об этом должен позаботиться пользователь-программист, испольузющий эту библиотеку.

7. Система должна быть достаточно гибкой.
   Это означает, что пользователь, имел бы возможность настраивать множество параметров, таких как коэфициент распараллеливания, время жизни топологии, максимальную память, которые занимают данные в системе.

Так выглядят требования в общем виде. В процессе разработке они будут уточняться и поясняться. Сразу следует отметить то, что данная реализация - это прототип системы потоковой обработки данных, а значит в этой системе не будет реализовано то, как кластеры будут взаимодействовать между собой. Будут лишь некоторые замечания и предположения о том, как это будет проходить. А во всем остальном - должен получиться полноценный распараллеливаемый кластер паралельной обработки данных.

1.3 Выбор языка программирования

Есть множество замечательных языков программирования, у каждого есть свои преимущества и недостатки. Автор же остановился на языке Java, поскольку он кросс-платформенный и довольно-таки распространенный в тех сферах, где планируется использовать данную систему (Постановка задачи). Однако, у Java есть один большой

недостаток - программы на Java чуть ли не самые медленные среди всех языков программирования. Для такой системы закономерно выбирать какой-нибудь функциональный язык из семейства Lisp, поскольку обладают лаконичностью, компактностью и выразительностью. Хорошо подходят для этого Clojure, Scala или Groovy, которые реализованы на java-машине, т.е. обладают полной совместимостью с Java и обладают той же кросс-платформенностью. Но в силу ограниченности по времени написания данной работы, не было возможности выучить один из этих языков. Поэтому, жертвуя производительностью в пользу удобства написания, данная система будет полностью написана на Java.

2 Разработка прототипа

2.1 Общие сведения о схеме

Здесь, как говорится не будем изобретать велосипед заново (хотя, возможно, стоило бы и попытаться сделать все по-другому) и возьмем уже привычную схему для потоковой системы.

Основное место занимает топология(Topology), которая определяет, как именно данные перемещаются внутри программы. Элементами топологии являются воронки(spouts) и сита (bolts). Роль воронок заключается в том, чтобы извлекать данные извне, например, из БД, http, xml или из файлов. В воронках нет никаких вычислительных элементов, они всего лишь занимаются извлечением данных. Говоря терминами функционального программирования воронки - это “грязные”функции, т.е. они взаимодействуют с внешним миром, и их результат не всегда зависит от входных данных.

Смысл же сит(bolts) заключается в том, чтобы обрабатывать кортежи данных, которые им предназначены и передавать их дальше. Сита также могут взимодействовать с внешним миром - передавать результат работы топологии дальше.

На рис. Схема топологии показана примерная схема топологии (в зависимости от реализации она может различаться).

Итак все в этом приложении должно строиться вокруг топологии и для топологии, чтобы иметь возможность вычислить кортежи, которые будут подаваться на вход. С точки зрения пользователя этой информации будет достаточно для понимания, как работать с данной системой в целом.

2.2 Взаимодествие с пользователем

Один из первых вопросов, которые возникают при разработке любого приложения - как данные от пользователя буду попадать в программу? Каким образом пользователь будет взаимодействовать с программой? Здесь мы воспользуемся опытом уже известной и достаточно популярной системой потоковой обработки данных - Apache Storm. Кластер системы будет предоставляться как библиотека на языке java, который программист-пользователь, импортируя в свою программу, получит набор интерфейсов

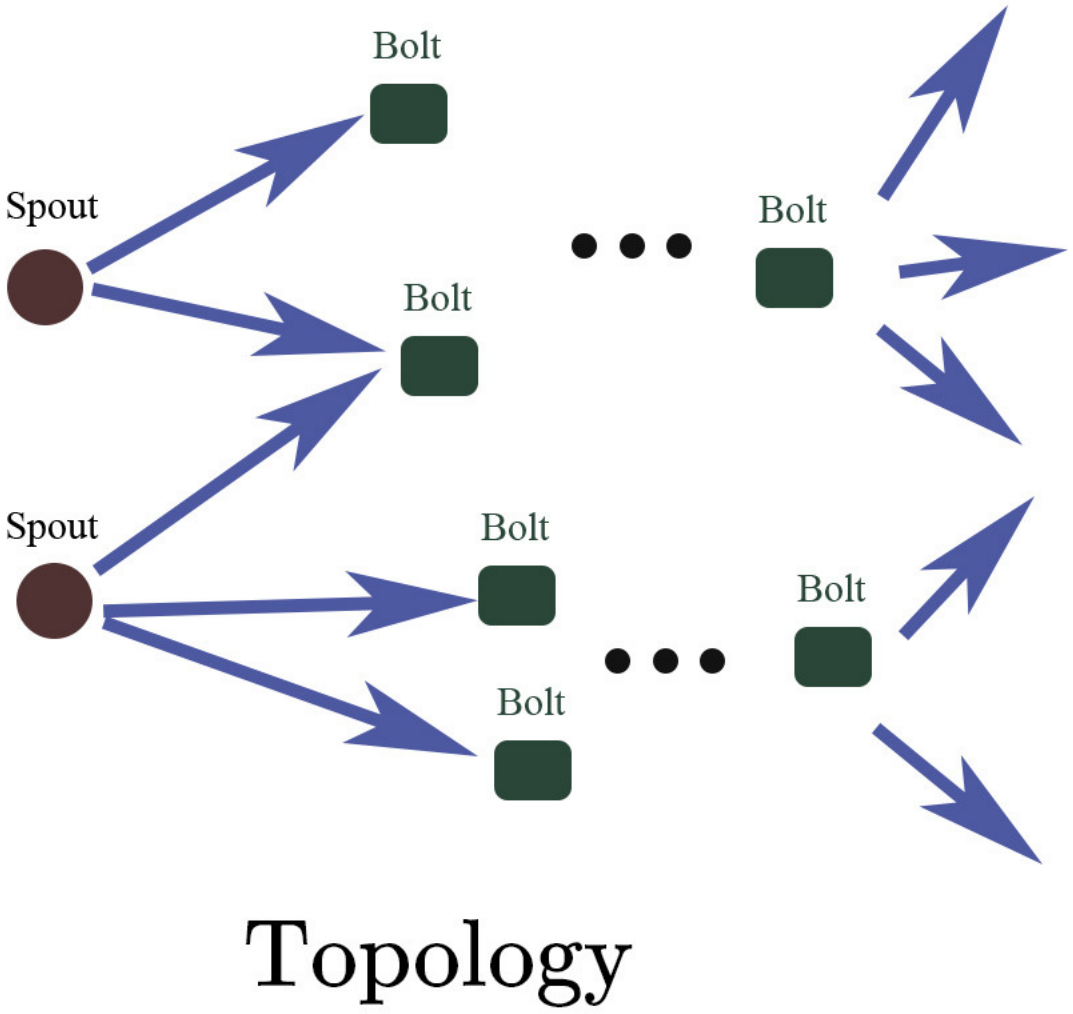

Рис. 1: Схема топологии

для коммуникации с кластером.

Основные возможности, которые доступны пользователю - это создание воронок и сит, наследующих определенный интерфейс, описанный автором. Предполагается, что пользователь будет добавлять сита и воронки в объект класса Topology - основной объект, которым будет манипулировать система при вычислении.

Пользователю также должна предоставляться возможно определять кортежи. Единственное требование на данном этапе, которое должно удовлетворяться кортежами - оно должно наследовать стандартный интерфейс кортежа, описанный в системе. На данном этапе этот интерфейс не имеет определенных методов, только лишь расширяет java-интерфейс Serializable. Это необходимо в случае распределения системы на несколько кластеров.

2.3 Логика приложения в общих чертах

За неимением опыта разработки больших и сложных систем, автор испытывает небольшую сложность в формальном описании системы и приносит свои извинения.

Конечно, выглядит не все так просто, как было написано в схемах общего вида, давайте же разберемся в этом. Первое, что нужно сказать, что топология(читать "граф") представлена стандартным образом, в виде списка воронок, сит и связей между ними.

3 Реализация прототипа

Заключение

Список литературы

Приложение. Код программы