Разработка алгоритма для выделения частовстречающихся шаблонов ошибок из log-файлов программ

Оглавление

Введение **3**

1 Анализ предметной области **5**
1.1 Представление задачи в терминах MapReduce 5
1.2 Выбор языка программирования и средств разработки 5
1.3 Структура данных 6
1.4 Стадии выполнения задачи 6
1.4.1 Написание программного кода 6
1.4.2 Написание функциональных тестов 6

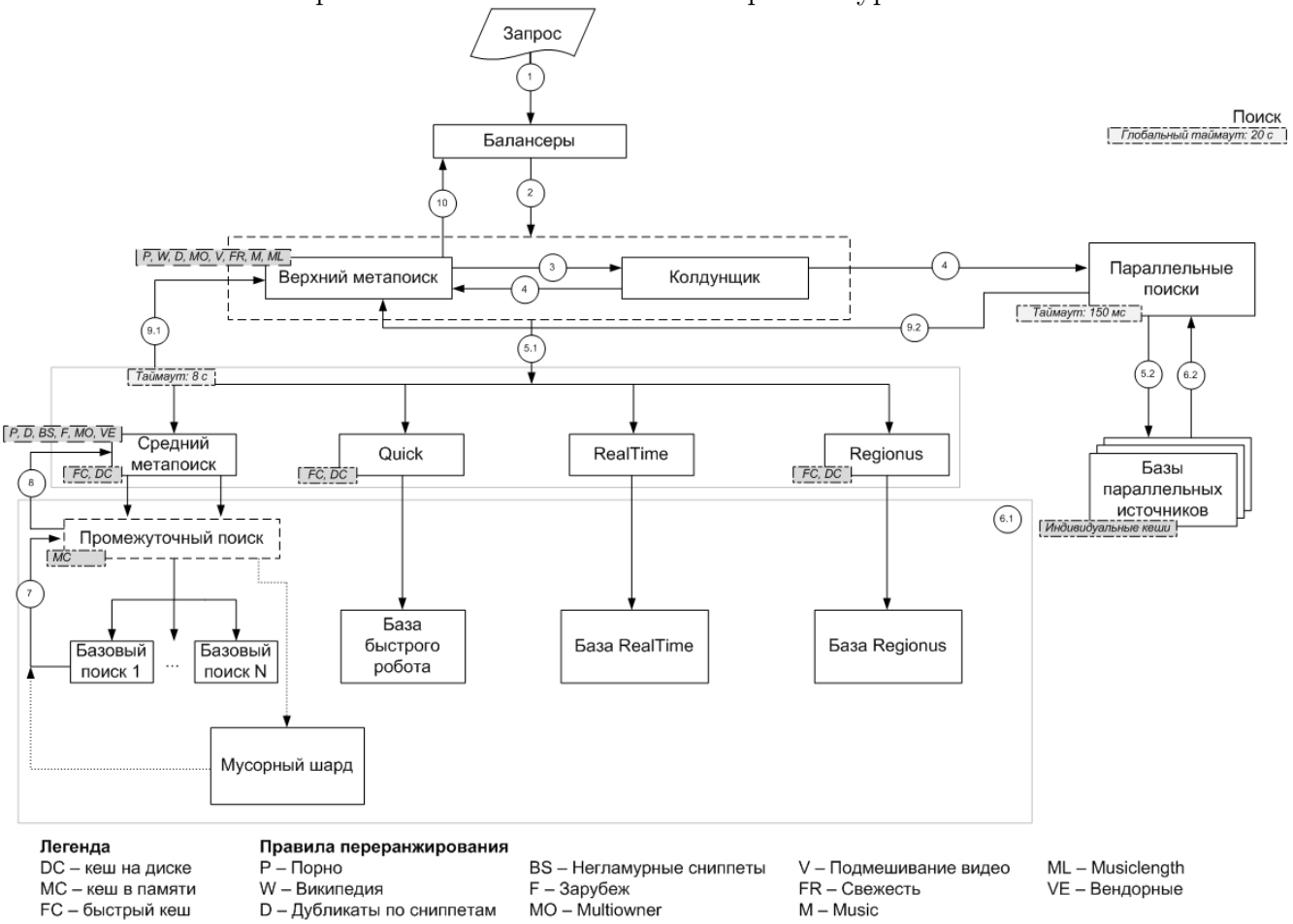
2 Имплементация задачи **7**
2.1 Диаграмма компонентов 7
2.2 Описание способов запуска 7
2.3 Анализ полученных результатов 7

3 Дальнейшее развитие **8**

Заключение **9**

Введение

Большинство программных систем, имеющих сложную структуру и состоящих из нескольких сотен различных компонент, обладают рядом схожих проблем. Например веб-поиск содержит следующие компоненты: балансеры, верхние, средние метапоиски, промежуточные и базовые поиски, колдунщики, антироботы, свежесть, региональные поиск, несколько десятков параллельных поисков. Краткое схематичное описание архитектуры:



Всего единовременно запущено несколько ******** тысяч инстансов [1] приложений. Каждый инстанс генерирует множество ошибок и записывает каждую из них в log-файл.

Некоторые приложения, близкие по функционалу, пишут в один и тот же файл. Log-файлы ротируются согласно определённому алгоритму. Тем не менее объем log-файла для одного инстанса может достигать нескольких сотен мегабайт, что препятствует быстрому ручному анализу в случае инцидента и инженеры вынуждены тратить ценные секунды на просмотр сотен тысяч строк файла в поисках сообщения с описанием элемента, вызвавшего сбой работы системы.

Начальным требованием к системе для эффективного использования алгоритма является наличие сопоставимого с количеством поисковых приложений количества нод на которых может быть запущена программа, реализующая алгоритм.

В организации, в которой выполнялась учебно-исследовательская работа, развёрнута боль-

[1] инстанс — приложение, запущенное в контейнере и описываемое парой host:port

шая поисковая инфраструктура, которая не лишена недостатков и существует вероятность поломки некоторой её части. Существует множество средств мониторинга состояния веб-поиска и противодействия инцидентам, но в некоторых случаях инженерам их недостаточно и приходится вручную анализировать log-файлы отдельных инстансов приложений на отдельных host'ах, что, в свою очередь, замедляет скорость реакции на непредвиденную ситуацию. Но даже автоматизация процесса анализа log-файла на одного инстанса не решает проблему полностью, поэтому необходима возможность быстрого анализа log-файлов сразу множества инстансов.

Таким образом, целью этой учебно-исследовательской работы является разработка алгоритма, позволяющего собирать статистику по ошибкам, встречающимся в log-файлах инстансов поисковых приложений на основе существующих шаблонов и выделять новые шаблоны.

Глава 1

Анализ предметной области

1.1 Представление задачи в терминах MapReduce

Для распределённого запуска приложений была выбрана модель распределённых вычислений MapReduce[1] по ряду причин.

Как оказалось задача без особых сложностей выражается в терминах чистых функций flat map и flat reduce, так как основная структура, используемая в алгоритме — это пара(паттерн[2], количество совпадений) и благодаря этому однопоточный код легко запускается на множестве нод MapReduce-кластера. Это обстоятельство освобождает от реализации сложного механизма сетевого взаимодействия.

В качестве реализации модели MapReduce была выбрана реализация Yet Another Map Reduce. Код программы, реализующий алгоритм легко переносится на другие реализации данной модели распределённых вычислений.

1.2 Выбор языка программирования и средств разработки

В качестве среды разработки было решено использовать удалённую виртуальную машину с конфигурацией и окружением идентичными конфигурации и окружению MapReduce ноды. Были выбраны следующие программные продукты:

- В качестве операционной системы использовался дистрибутив GNU/Linux Ubuntu 12.04 LTS с модифицированным ядром и дополнительными пакетами.
- Vim — один из немногих несуществующих настоящих текстовых редакторов, обладающих массой встроенных возможностей и практически безграничным потенциалом к расширению за счёт встроенного интерпретируемого языка программирования VimL и поддержкой возможности написания расширений на таких языках, как python, ruby, perl.
- Сохранение состояние рабочего окружения осуществлялось с помощью терминального мультиплексора tmux[3], имеющего клиент-серверную архитектуру и позволяющего отсоединятся от текущей сессии, оставляя её работать в фоновом режиме с последующей возможностью переподключения.

[1] MapReduce — модель распределённых вычислений, представленная компанией Google, используемая для параллельных вычислений над очень большими, несколько петабайт, наборами данных в компьютерных кластерах.

[2] паттерн — регулярное выражение

[3] tmux — свободная консольная утилита-мультиплексор, предоставляющая пользователю доступ к нескольким терминалам в рамках одного экрана. tmux может быть отключён от экрана: в этом случае он продолжит

- Удалённое подключение осуществлялось средствами защищённого протокола ssh[4] (беспарольная аутентификация с использованием ключа) и технологии cauth.

1.3 Структура данных

1.4 Стадии выполнения задачи

1.4.1 Написание программного кода

1.4.2 Написание функциональных тестов

исполняться в фоновом режиме; имеется возможность вновь подключиться к tmux, находящемуся в фоне. tmux является штатным мультиплексором терминалов операционной системы OpenBSD. Программа tmux задумывалась как замена программы GNU Screen.

[4]SSH (англ. Secure Shell — «безопасная оболочка»]) — сетевой протокол прикладного уровня, позволяющий производить удалённое управление операционной системой и туннелирование TCP-соединений (например, для передачи файлов). Схож по функциональности с протоколами Telnet и rlogin, но, в отличие от них, шифрует весь трафик, включая и передаваемые пароли. SSH допускает выбор различных алгоритмов шифрования. SSH-клиенты и SSH-серверы доступны для большинства сетевых операционных систем.

Глава 2

Имплементация задачи

2.1 Диаграмма компонентов

2.2 Описание способов запуска

2.3 Анализ полученных результатов

Глава 3

Дальнейшее развитие

Заключение